

Digital Wayfinding im Rahmen von Digital Signage

Cristopher Hornsey | Reuben Meixner | Emrah Gudic

06.April.2016

Name	Aufgabenstellung
Cristopher Hornsey	Digital Wayfinding Editor
Emrah Gudic	Digital Wayfinding Client
Reuben Meixner	Qualitätssicherung

Eidesstattliche Erklärung

Wir erklären hiermit an Eides statt, dass wir die vorliegende Diplomarbeit selbstständig angefertigt haben. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Cristopher Hornsey

Reuben Meixner

Emrah Gudic

April, 2016

Abstract

Deutsch

Im Zuge der Diplomarbeit von Cristopher Hornsey, Reuben Meixner und Emrah Gudic ein Wegfindungseditor und ein Wegfindungs-Client für den Auftraggeber Herrn Mag. Ing. Prof. Hans-Christian Hammer entwickelt. Das Finden eines Raumes in Gebäuden gestaltet sich oft äußerst schwierig und gelingt auch meist ohne Nachfrage nicht. Der Wegfindungs-Client soll daher eine gut bedienbare Navigation in Gebäuden darbieten, um dem Benutzer eine Zeitersparnis zu bringen und sie über den schnellsten und kürzesten Weg zum Ziel führen.

Englisch

In the course of the mandatory thesis of Cristopher Hornsey, Reuben Meixner and Emrah Gudic a wayfinding editor and a wayfinding client is being developed for the customer Mag. Ing. Prof. Hans-Christian Hammer. Rooms can be hard to find, especially in larger buildings and finding a person who knows their way around the building, can also be tricky. The wayfinding editor and client serve to help visitors easily find the shortest route to their desired destination.

Danksagung

Wir möchten uns herzlichst bei allen bedanken, die uns bei unserem Projekt unterstützt haben und zu seinem Gelingen beigetragen haben:

- Bei unseren Lehrern, die uns während des Projekts zur Hilfe bereit standen und uns bei Fragen behilflich waren.
- Bei unserem Betreuungslehrer Mag. Dr. Prof. Thomas Stütz, der dem Diplomarbeitsteam mit Rat und Tat zur Seite stand.
- Bei allen Korrekturlesern der Dokumente, um diese fehlerfrei und ordentlich einzureichen.
- Bei unseren Eltern, die uns während den Arbeiten am Projekt durch Verpflegung unterstützten und Ihre objektive Meinung zu Gestaltungen erbrachten.
- Bei unseren hilfsbereiten Mitschülern, die uns konstruktive Kritik und wichtiges Feedback gegeben haben.
- Und natürlich bei unserem Auftraggeber Mag. Ing. Prof. Hans-Christian Hammer, der diese Diplomarbeit erst möglich gemacht hat und uns durch Engagement und Interesse den Weg erleichterte.

Inhaltsverzeichnis

1	Pflichtenheft	9
1.1	Motivation	9
1.2	Ausgangssituation und Zielsetzung	9
1.2.1	Ausgangssituation	9
1.2.2	Überblick über die Geschäftsprozesse	9
1.2.3	Zielfdefinition	10
1.3	Funktionale Anforderungen	10
1.3.1	An den Wayfinding Editor	10
1.3.2	An die Website	10
1.3.3	Use Case 1	11
1.3.4	Use Case 2	12
1.4	Nicht-funktionale Anforderungen	13
1.4.1	An den Wayfinding Editor	13
1.4.2	An die Website	14
1.5	Mengengerüst	15
1.6	Akzeptanzkriterien	16
2	Kriterienkatalog	17
2.1	Server	18
2.1.1	Art des Servers	18
2.1.2	Operating System des Servers	19
2.2	Player	20
2.2.1	Operating System des Players	20
2.2.2	Tauglichkeit für Einplatinencomputer	21
2.2.3	Stromverbrauch	22

2.3	Einbindung externer Quellen	23
2.4	Usability	24
2.4.1	Usability in Bezug auf die Bedienung des Backends . .	24
2.4.2	Usability in Bezug auf die Bedienung des Frontends . .	24
2.5	Multi-Monitor	25
2.6	Medienformate	26
2.6.1	Arten von Videoformaten	27
2.6.2	Arten von Bildformaten	28
2.6.3	Andere Dateiformate	30
2.7	Touchfunktion	33
3	Verwendete Technologien	35
3.1	Systemarchitektur	35
3.2	Hypertext Markup Language	35
3.2.1	Nutzung	36
3.2.2	Anwendung	36
3.3	JavaScript	37
3.3.1	Nutzung	37
3.3.2	Anwendung	37
3.4	Cascading Stylesheet	38
3.4.1	Nutzung	38
3.4.2	Anwendung	39
3.5	JQuery	39
3.5.1	Implementierung	40
3.5.2	Anwendung	41
3.5.3	Benutzte Funktionen	41
3.5.4	JCanvas	42

3.6	Bootstrap	43
3.6.1	Implementierung	43
3.6.2	Anwendung	44
3.6.3	Verwendete Klassen	44
3.7	Afterburner - FX	45
3.7.1	Was ist Afterburner - FX?	45
3.7.2	Konvention	45
3.7.3	Implementierung	46
3.7.4	Property Injection	49
3.8	GSON	50
3.8.1	Was ist GSON?	50
3.8.2	Implementierung	50
3.8.3	Anwendung	51
3.9	Neo4j	52
3.9.1	Was ist Neo4j?	52
3.10	Neo4j - OGM	55
3.10.1	Was ist Neo4j - OGM ?	55
3.10.2	Implementierung	55
4	Umsetzung	61
4.1	Überblick - Editor	61
4.2	GUI	61
4.2.1	Canvas	63
4.2.2	Menubar	64
4.2.3	Knoten - Choicebox	65
4.2.4	Stockwerkliste	66
4.2.5	Editor Einstellungen	67

4.2.6	Properties	68
4.3	Ungewollte Knotenüberschreibung	73
4.3.1	Problem	73
4.3.2	Ursache	74
4.3.3	Behebung	74
4.4	Realisierung im Client	76
4.4.1	Dijkstras-Algorithmus	76
4.4.2	Dijkstras.js	77
4.4.3	Aufbau der JSON-Datei	78
4.4.4	Koordinaten anpassen	80
4.4.5	Zeichnen der Knoten und Kanten	80
4.4.6	Darstellung	81
4.5	Überblick-Client	82
4.6	Probleme bei der Umsetzung des Clients	82
4.6.1	Erstellung des Arrays	83
4.6.2	Zeichnen des Pfades	83
4.6.3	Weiter & Zurück-Button	84
4.7	GUI-Client	85
4.7.1	Auswahl der Station	86
4.7.2	Auswahl der Unterstation	86
4.8	Auswahl des Raumes	87
4.9	Canvas	87
5	Anhang	89

1 Pflichtenheft

Projektname	Digital Wayfinding
Teamleiter	Cristopher Hornsey
Teammitglieder	Cristopher Hornsey, Emrah Gudic, Reuben Meixner
Erstellt am	25.02.2016
Zuletzt geändert am	31.03.2016

1.1 Motivation

Als Schulabschlussarbeit muss eine Diplomarbeit verfasst werden. Dazu gehört eine praktische Programmierarbeit. Die Idee ein Digital Wayfinding System im Rahmen von Digital Signage zu erstellen wurde uns von Herr Prof. Stütz und Herr Prof. Hammer näher gebracht.

1.2 Ausgangssituation und Zielsetzung

1.2.1 Ausgangssituation

Besucher und Patienten können sich im Krankenhaus nur schwer zurechtfinden und müssen entweder anhand von statischen Karten den Weg finden oder einen Mitarbeiter der mit dem Gebäude vertraut ist, nach Hilfe fragen.

1.2.2 Überblick über die Geschäftsprozesse

- Wayfinding Editor zum Editieren der Gebäudepläne und deren Knoten
- Datenbank zum Speichern der Gebäudpläne
- Export der Gebäudepläne
- Website für die Wegfindung

1.2.3 Zielfdefinition

Es gibt einen Bildschirm mit Touchscreen Funktion auf dem ein Digital Signage System rennt, welches Werbung, Newsticker, etc. einblendet. Per Klick eines Benutzers wird das Digital Wayfinding eingeschaltet. Der Benutzer wählt sein gewünschtes Ziel aus und kann zusätzlich angeben ob er einen Weg für Personen mit eingeschränkter Mobilität benötigt. Anschließend wird der kürzeste Weg zum Ziel berechnet und angezeigt. Ebenfalls gibt es ein Programm, welches das Editieren der Wegfindung in mehreren Gebäuden ermöglicht und diese Änderung. Diese Änderungen können dann exportiert werden und an die Digital Wayfinding Systeme verteilt werden.

1.3 Funktionale Anforderungen

1.3.1 An den Wayfinding Editor

Es soll möglich sein mehrere Gebäude im Editor hinzuzufügen und die einzelnen Stockwerke des jeweiligen Gebäudes zu bearbeiten.
Es muss möglich sein Knoten verschiedener Arten (Stiegen, Eckpunkte, Lifte, Zimmer) auf einem Stockwerk zu setzen und diese zu verbinden.
Man soll die Möglichkeit haben, zwischen barrierefreien Knoten und nicht barrierefreien Knoten zu unterscheiden, dabei sind Stiegen standardmäßig nicht barrierefrei und Lifte barrierefrei. Ebenso soll man festlegen können, ob Knoten zurzeit generell nicht benutzbar sind.
Es soll eine Export Funktion im Editor geben, welche den gesamten Gebäudeplan samt den gesetzten Knoten in einen String umwandelt, der an die Digital Wayfinding Systeme weitergeleitet werden kann.
Die Relationen zwischen den Knoten sollen eine Gewichtung haben, die die Distanz widerspiegelt. Mithilfe dieser Gewichtung kann dann der kürzeste Pfad berechnet werden.

1.3.2 An die Website

Die Website soll den kürzesten Pfad, mit Rücksicht auf Personen mit eingeschränkter Mobilität, vom jetzigen Standpunkt aus zum gewünschten Zielort berechnen können.

Bei der Eingabe soll keine Liste mit allen möglichen Zielpunkten angezeigt werden, sondern man schränkt den möglichen Zielort ein.
Es soll eine Druckfunktion eingebaut werden die das aktuelle Stockwerk bei einem zur Verfügung gestellten Drucker druckt.

1.3.3 Use Case 1

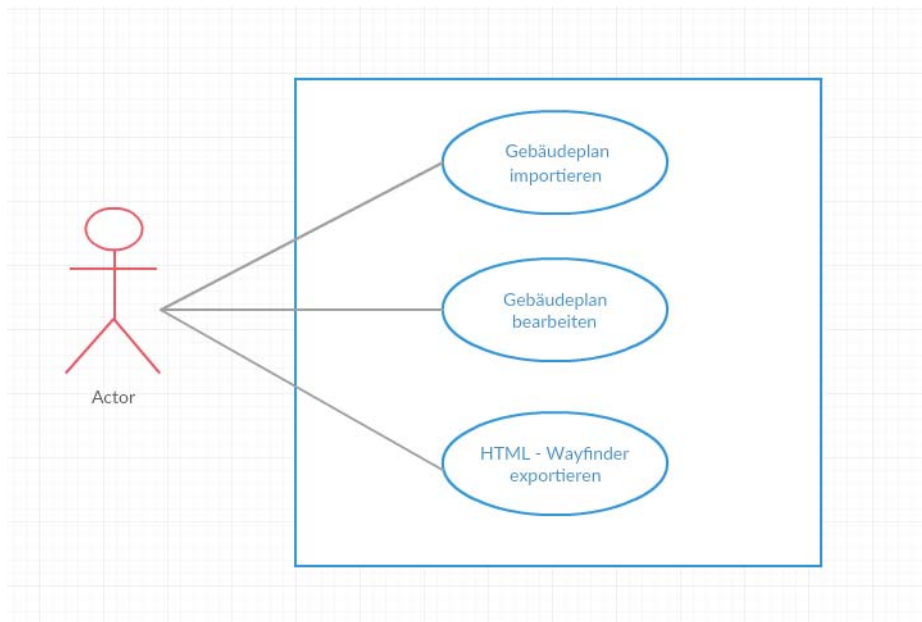


Figure 1: Use Case Editor

Beschreibung

Der Benutzer des Wayfinding Editors kann verschiedene Gebäudepläne importieren, diese dann bearbeiten und zum Schluss exportieren und an die Wayfinding Systeme verteilen.

Charakterisierende Information

Ziel	Durch setzen der Knoten in einem Gebäudeplan, wird die Ermittlung des kürzesten Pfades ermöglicht und es kann auf Sonderfälle Rücksicht genommen werden (Liftausfall, Rollstuhl, etc.).
Vorbedingung	Es muss ein genauer Gebäudeplan zur Verfügung stehen, so wie das Wissen wo sich was im Gebäude befindet.
Nachbedingung	Ist er einmal erzeugt, kann man den Gebäudeplan jederzeit ändern und die neue Version an die Wayfinding Systeme verteilen.
Involvierte Benutzer	Der Benutzer des Wayfinding Editors.

1.3.4 Use Case 2

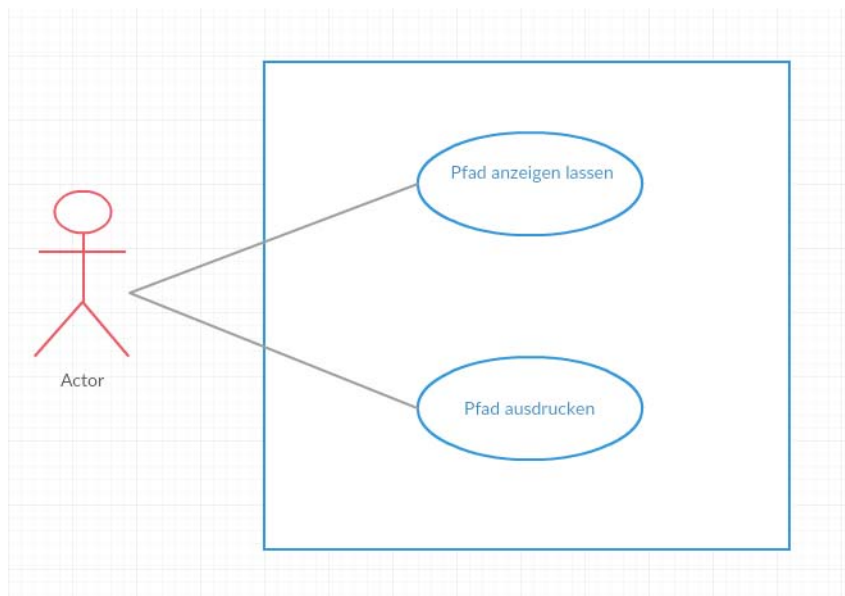


Figure 2: Use Case Client

Beschreibung

Der Benutzer des Wayfinding Systems kann seinen gewünschten Zielort eingeben und erhält dann den kürzesten für ihn geeigneten Weg. Falls der Benutzer es wünscht kann er die gewünschten Stockwerke ausdrucken.

Charakterisierende Information

Ziel	Ermittlung des kürzesten Pfades und dessen Anzeige.
Vorbedingung	Die Website benötigt den Export des Wayfinding Editors.
Nachbedingung	Die Benutzer haben einen für sie geeigneten Weg zu ihrem gewünschten Ziel.
Involvierte Benutzer	Besucher und Mitarbeiter des Krankenhauses.

1.4 Nicht-funktionale Anforderungen

1.4.1 An den Wayfinding Editor

ID	SG
Name	Skalierbarkeit des Gebäudeplans
Type	EFFICIENCY
Description	Es soll möglich sein einen Skalar für den Gebäudeplan angeben zu können, und die Option die Distanz (Gewichtung) zwischen den Knoten anhand dieses Skalars automatisch zu berechnen.
Assigned Use Cases	1

ID	AG
Name	Anzeige der Gewichtung
Type	USE
Description	Die Anzeige der Gewichtung soll ein- und ausschaltbar sein. Die Größe der Anzeige soll ebenfalls veränderbar sein.
Assigned Use Cases	1

ID	AK
Name	Anzeige der Knoten
Type	USE
Description	Die Knoten sollen standardmäßig anhand ihrer Farben eindeutig voneinander unterscheidbar sein. Der Benutzer soll jedoch die Möglichkeit haben die Farben für bestimmte Knotenarten selbst zu wählen. Ebenso soll es möglich sein die Größe der Knoten zu ändern.
Assigned Use Cases	1

1.4.2 An die Website

ID	KB
Name	Kurze Berechnungszeit
Type	EFFICIENCY
Description	Das Berechnen des effizientesten Weges darf nicht länger als 2 Sekunden dauern.
Assigned Use Cases	2

ID	EZ
Name	Einfache Eingabe des Zielorts
Type	USE
Description	Dem Benutzer soll es so einfach als möglich gemacht, seinen gewünschten Zielort zu finden. Das Design der Liste betont dabei die Eingabe.
Assigned Use Cases	2

ID	EW
Name	Eindeutige Erkennbarkeit des Weges
Type	USE
Description	Die Farbe und Platzierung des gezeichneten Weges sollen für den Benutzer gut leserlich und vom Rest des Gebäudeplans hervorgehoben.
Assigned Use Cases	2

ID	SW
Name	Stockübergreifende Wege
Type	USE
Description	Geht ein Weg über mehrere Stockwerke, soll man zwischen den Stockwerken einfach weiter- und wieder zurückschalten können.
Assigned Use Cases	2

ID	PF
Name	Passende Frontpage
Type	USE
Description	Der Background der Website soll passend für ein Krankenhaus sein und soll Menschen in sichtbarer guter Stimmung darstellen, um den Benutzer ein positives Gefühl zu übermitteln.
Assigned Use Cases	2

1.5 Mengengerüst

Für jedes zugängliche Zimmer, und für jede Ecke, jede Stiege und jeden Lift muss eine Koordinate (Knoten) abgespeichert werden. Neben den Koordinaten werden Angaben über die Barrierefreiheit, die Möglichkeit der Betretung der Relationen (Datum oder Zeitspanne), die Art des Knotens sowie der Name gespeichert. Jeder Knoten besitzt mindestens 1 Relation, die sie mit einem anderen Knoten verbindet. (min. Rel.: $\#Knoten-1$). Bei der Annahme, dass auf einem Gebäudeplan 2000 Knoten gespeichert werden müssen, um eine korrekte Wegfindung sicherzustellen, müssen mindestens 3999 Datensätze erstellt werden. (2000 Knoten + mind. 1999 Relationen). Wenn jeder Datensatz 8 Kilobyte groß ist, dann werden mindestens 32Megabyte Speicherplatz benötigt.

1.6 Akzeptanzkriterien

Testschritt	Erwartetes Verhalten
Knoten setzen	Knoten von verschiedenen Arten werden gesetzt.
Relationen setzen	Die Knoten können miteinander verbunden werden und die Relation gewichtet werden.
Gebäude hinzufügen	Mehrere Gebäude werden angelegt werden und die Stockwerke können zu den jeweiligen Gebäuden hinzugefügt werden.
Sonderfälle berücksichtigen	Knoten können barrierefrei oder gerade nicht benutzbar gemacht werden.
Export	Der Gebäuderplan samt seinen Knoten wird in einen String umgewandelt und an die Wayfinding Systeme verteilt.
Eingabe des Zielorts	Der Zielort wird aus den verfügbaren Knoten gewählt.
Kürzester Pfad	Nach Eingabe des gewünschten Zielorts wird ein Gebäudeplan mit dem geeignetsten Pfad angezeigt.
Druck der Karte	Nach Klick des Druck-Buttons wird das aktuell angezeigte Stockwerk gedruckt.

2 Kriterienkatalog

Grundlage für die Erstellung einer Marktanalyse ist die Entwicklung eines Kriterienkataloges. Mithilfe dieses Kriterienkataloges können die einzelnen Produkte nicht nur verglichen, sondern auch bewertet werden. Bei der Entwicklung eines Kriterienkataloges ist insbesondere auf die Marktgegebenheiten und die Voraussetzungen für die einzelnen Einsatzgebiete zu achten.

Im folgenden Ansatz wurde versucht, eine möglichst neutrale und umfassende Darstellung von ,für verschiedene Branchen, relevanten Kriterien zu wählen.

Folgende Kriterien erscheinen uns in diesem Zusammenhang als spezifikationswürdig:

1. Server
 - (a) Art des Servers
 - (b) Operating System des Servers
2. Player
 - (a) Operating System des Players
 - (b) Tauglichkeit für Einplatinencomputer
 - (c) Stromverbrauch
3. Einbindung externer Quellen
 - (a) Einbindung externer Quellen via Internet
 - (b) Einbindung externer Quellen über zusätzliche Geräte im Player
4. Usability
 - (a) Usability in Bezug auf die Bedienung des Backends
 - (b) Usability in Bezug auf die Bedienung des Frontends
5. Multi-Monitor
6. Medienformate
7. Touchfunktion

Begründung für die Kriterien

2.1 Server

2.1.1 Art des Servers

Mit der Art des Servers sollen die verschiedenen Möglichkeiten definieren werden, wo die Software tatsächlich betrieben wird. Folgende Möglichkeiten wurden bei den einzelnen Anbietern solcher DS-Produkte gefunden:

- Betrieb des Servers im eigenen Rechenzentrum
- Auslagerung des Servers an den jeweiligen DS-Hersteller
- Betrieb des Servers in einem unabhängigen Rechenzentrum, welches zugekauft wird (Cloudlösung)

Der Betrieb des Servers im eigenen Rechenzentrum bietet einerseits den Vorteil, dass man selbst über die Höhe der Wartungskosten entscheiden kann, und nicht unbedingt an einen Wartungsvertrag gebunden ist. Ein Wartungsvertrag erscheint dennoch sinnvoll in Fällen, wo Spezialisten eingeflogen werden müssen und die Kosten für die Wartung des Servers schnell in die Höhe schießen. Außerdem bedeutet ein weiterer Server im Rechenzentrum einen höheren Stromverbrauch und erfordert möglicherweise auch bessere Kühlungsmaßnahmen des Rechenzentrums.

Eine Cloudlösung oder die Auslagerung des Servers an den Digital Signage Hersteller bietet jedoch mehr Vorteile als den Betrieb eines eigenen Rechenzentrums.

Die gemeinsame Eigenschaft aller Arten von Cloud-Diensten lässt sich mit zwei Worten beschreiben: "On Demand" - bei Bedarf. Auch wenn sich die nachfolgend aufgeführten Cloud-Modelle voneinander unterscheiden, steht immer der Wunsch des Kunden im Mittelpunkt: Statt bei einem wie auch immer gearteten Bedarf in eigene Infrastrukturen, Software, Dienste oder Lösungen zu investieren, mietet er selbige bei einem Dienstleister an. Kunden stehen über Cloud-Computing maßgeschneiderte, jederzeit skalierbare Dienstleistungen mit kalkulierbaren Kosten zur Verfügung. Cloud-Dienste haben noch einen weiteren Vorteil: Während vor allem kleine und mittelständische Unternehmen, bei Hard- und Software oder

Dienstleistungen, oft Kompromisse eingehen müssten, können sie dank Cloud auf Top-Lösungen zurückgreifen, die von Spezialisten betreut werden. Grundlage für beinahe sämtliche Cloud-Spielarten ist Virtualisierung. Denn nur so lassen sich die vorhandenen Hardware-Ressourcen optimal ausnutzen. Für Modelle wie PaaS (Platform as a Service) ist Virtualisierung zudem unabdingbar, um auf derselben Hardware verschiedene Mandanten versorgen zu können. Insofern muss die verwendete Hardware optimal mit den besonderen Anforderungen der Virtualisierung zurechtkommen. Intels Xeon-CPU's sind insbesondere durch die Intel Virtualization Technology (Intel VT) und die Intel Trusted Execution Technology (Intel TXT) sehr gut für virtualisierte Umgebungen geeignet. Dank Intel VT können virtuelle Maschinen gezielter auf die Hardware zugreifen. Das resultiert in verbesserter Ausfallsicherheit, Sicherheit und System-Performance virtueller Umgebungen. Intel TXT etabliert eine "Chain of Trust" zwischen Hardware und virtueller Maschine: Die Technik überwacht den Start der VM und erkennt jede Modifikation durch Schad-Software oder durch Anwender.[15]

Cloudlösung und die Auslagerung des Servers an den DS-Hersteller sind besser, da die Gefahr einer Verletzung des Datenschutzes zwar besteht aber sehr klein ist und die Kosten für Stromverbrauch, bessere Kühlungen und Wartungsgebühren extrem hoch ausfallen könnten.

Aus den oben angeführten Begründungen ist zu erkennen, dass das Produkt, welches alle Möglichkeiten unterstützt, mit der höchsten Punktzahl zu bewerten ist.

2.1.2 Operating System des Servers

Bei den verschiedenen DS-Produkten wurden folgende OS-Möglichkeiten gefunden:

- Microsoft Windows Server
- Linux basierte Betriebssysteme

Im direkten Vergleich von Microsoft Windows Server zu Linux basierten Betriebssystemen wurde festgestellt, dass erstere durch die Bank kostenpflichtig sind und zweitere unter den Bedingungen der GNU General

Public License kostenlos verfügbar sind. Wobei die Bedienung und Administration von Linux basierten Serversystemen meist ein höheres Know-How als die Bedienung und Administration von Microsoft Windows Serversystemen voraussetzt. Unter Umständen muss hier mehr investiert werden als durch geringere Lizenzierungskosten eingespart werden kann.

Den Linux basierten Betriebssystemen wird trotzdem der Vorzug gegeben, da diese in solchen Umgebungen mittlerweile als defakto Standard zu betrachten sind.

Auffällig ist, dass keine Implementierungen auf Basis AppleOS gefunden wurden.

Folglich ist das Digital Signage Produkt, welches beide Betriebssysteme unterstützt, mit der vollen Punktzahl zu bewerten.

Zusätzlich muss in Betracht gezogen werden, dass die komplette Serverinstallation auch in einer virtuellen Umgebung betrieben werden kann. Dafür haben wir zusätzliche Punkte vergeben.

Tabelle mit Punkten:

Unterkriterium	Punkte
Server im eigenen Rechenzentrum	2 Punkte
Auslagerung des Servers an den jeweiligen DS-Hersteller	1 Punkt
Betrieb des Servers in einem unabhängigen Rechenzentrum	2 Punkte
Linux basierte Betriebssysteme	3 Punkte
Microsoft Windows Server	2 Punkte

Tabelle 1: Server

2.2 Player

2.2.1 Operating System des Players

Bei den verschiedenen DS-Produkten wurden folgende OS-Möglichkeiten gefunden:

- Microsoft Windows

-
- Linux basierte Betriebssysteme

Im direkten Vergleich von Microsoft Windows zu Linux basierten Betriebssystemen wurde festgestellt, dass erstere durch die Bank kostenpflichtig sind und zweitere unter den Bedingungen der GNU General Public License kostenlos verfügbar sind. Wobei die Bedienung und Administration von Linux basierten Betriebssystemen meist ein höheres Know-How als die Bedienung und Administration von Microsoft Windows Betriebssystemen voraussetzt. Unter Umständen muss hier mehr investiert werden, als durch geringere Lizenzierungskosten eingespart werden kann.

In unserer Betrachtungsweise geben wir trotzdem den Linux basierten Betriebssystemen den Vorzug, da diese in solchen Umgebungen mittlerweile als defakto Standard zu betrachten sind.

Wegen den oben angeführten Begründungen ist die höchste Punktezahl für das Digital Signage Produkt zu vergeben, das beide Betriebssysteme unterstützt.

2.2.2 Tauglichkeit für Einplatinencomputer

Gerade beim Player ist es wichtig, dass die eingesetzte Lösung möglichst stabil und verlässlich auch im 24/7 Betrieb funktioniert. Die höchsten Kosten entstehen bei Ausfällen der Playerhardware. Dies erklärt sich dadurch, dass spezialisiertes Personal den Fehler diagnostizieren und anschließend beheben muss. Die derzeitigen Personalkosten in diesem Bereich sind deutlich höher einzustufen als etwaige zusätzliche Investitionen in die Hardware. In diesem Zusammenhang muss auch untersucht werden, wie einfach ein kaputter Player durch Personal vor Ort ausgetauscht werden kann. Sollte es genügen diesen Player einfach durch einen neuen, mittels Ein- und Ausstecken zu ersetzen und die erforderliche Installation Remote durchzuführen, dann kann man sich die spezialisierten Personalkosten ersparen.

Neben den hohen Kosten bei Hardwareausfällen ist der Kostenaufwand bei Playern, welche auf einem Personalcomputer laufen, nicht außer Acht zu lassen. Dieser Mehraufwand kann durch viele im Einsatz stehende Standrechner verursacht werden die einen viel höheren Einkaufspreis haben als Single Board Devices.

Damit der Player auf Single Board Devices lauffähig sein kann, ist es wichtig die Software so leicht wie möglich zu programmieren. Das heißt, dass die Software ressourcenschonend ist und mit einer geringen Leistung, welche bei Einplatinencomputern im Vergleich mit Standrechnern der Fall ist, auskommen kann und trotzdem schnell genug ist, damit kein Performanceunterschied erkennbar ist.

2.2.3 Stromverbrauch

Bei vielen DS-Installationen sind sowohl Player als auch Anzeigegerät im 24/7 Betrieb. Dadurch wird auch der Stromverbrauch dieser Geräte zu einem zu berücksichtigenden Faktor.

Die neueste Generation der Einplatinencomputer (Odroid, Raspberry PI, etc.) hat einen durchschnittlichen Stromverbrauch von 4 Watt.

Berechnungsformel für den Stromverbrauch:

Stundenverbrauch (Watt) x Nutzungstunden pro Tag x 365,25 x
KWh-Preis [33]

Mittels der Berechnungsformel für den Stromverbrauch ergibt sich pro Player zwischen ersterem und zweiterem ein Einsparungsfaktor von ca. 283,70 Euro pro Jahr.

Da der Player rund um die Uhr läuft, müssen die Kosten für den Strom berücksichtigt werden. Die entstehenden Stromkosten können durch einen Player gesenkt werden, der tauglich für Einplatinencomputer ist.

Player, die auf Einplatinencomputern laufen können, bekommen 2 Punkte. Für sehr stromintensive Player fällt die Bewertung mit null Punkten mager aus. Die Player die wenig Strom verbrauchen aber nicht auf Single Board Devices laufen erhalten daher einen Punkt.

Tabelle mit Punkten

Unterkriterium	Punkte
Stromverbrauch < 10 W	2 Punkte
Stromverbrauch < 100 W	1 Punkt
Tauglich für Einplatinencomputer	2 Punkte
Microsoft Windows OS	2 Punkte
Linux basiertes OS	3 Punkte

Tabelle 2: Player

2.3 Einbindung externer Quellen

Im Rahmen der Diplomarbeit haben wurden externe Quellen gefunden, welche in solchen Systemen eingebunden werden können:

- RSS-Feed (Technologie für Newsticker)
- Streaming Inhalte (z.B. YouTube)

Für den Betrieb von DS Systemen erscheint es wichtig zu sein, dass man auf externe Inhalte live zugreifen kann. Dabei können wir unterscheiden ob auf diese Quellen via Internet oder über zusätzliche im Player eingebaute Hardware (DVBT Empfänger) zugegriffen wird. Auf jeden Fall ist zu prüfen ob der Zugriff und die Anzeige dieser externen Quellen alle gesetzlichen Grundlagen einhält.

Die gefundenen externen Quellen lassen sich in Quellen aus dem Internet und Quellen über zusätzliche Hardware im Player unterteilen.

Die externen Quellen, welche via Internet bezogen werden, können Videos, Fotos, Text oder andere Dateien sein. Sie sind eine einfachere Bezugsquelle und meistens auch kostenlos. Sie brauchen keine zusätzliche Hardware im Player. Eine Gefahr, bei aus dem Internet bezogenen Quellen stellt die Stabilität des Datenflusses dar, da man von einem ISP abhängig ist. Ohne Zweifel bekommt der Player der beide Arten und RSS-Feeds einbindet die höchste Punktzahl.

Tabelle mit Punkten:

Unterkriterium	Punkte
RSS-Feed	2 Punkte
Internet Stream	4 Punkte
Hardwarestream (DVB-T)	4 Punkte

Tabelle 3: Externe Quellen

2.4 Usability

2.4.1 Usability in Bezug auf die Bedienung des Backends

Unter der Usability in Bezug auf die Bedienung des Backends verstehen man, die einfache Bedienung und Handhabung der Software, welche dem Verwalter zur Verfügung steht, um neue Playlists auf dem Player zu erstellen oder auch zu ändern.

Wichtig ist, dass die Menschen welchen den Player verwalten und für diese Playlists erstellen kein großes technisches Know-How benötigen müssen. Sie müssen ohne Einschulungen und Benutzerhandbücher die Software bedienen können. Dies erfolgt nur dann, wenn die Texte und Beschreibungen der einzelnen Komponenten der Software für sich selbst sprechen und diese auch logisch von nicht technisch versierten Menschen miteinander in Verknüpfung gebracht werden können.

Es ist ebenfalls wichtig, dass die Software nicht zu komplex ist, denn auch wenn die einzelnen Komponenten der Software selbstbeschreibend sind und man diese auch logisch verknüpfen kann, kann der Benutzer etwas falsch eingeben und erst zum Schluss draufkommen. Wenn die Software also sehr komplex ist verliert er sich in ihr und findet seinen Fehler nicht mehr, obwohl sie vom Anfang bis zum Ende einfach zu bedienen war.

Um die Bewertung durchzuführen und Punkte vergeben zu können, für dieses Kriterium, muss die Software unter die Lupe genommen und getestet werden, sodass gesagt werden kann, ob die Software auf den ersten Blick ansprechend und diese auch einfach zu verstehen ist, oder nicht.

2.4.2 Usability in Bezug auf die Bedienung des Frontends

Unter der Usability in Bezug auf die Bedienung des Frontends verstehen man, die einfache Bedienung und Handhabung der Software, welche dem

Zuseher zur Verfügung steht, um in einem Gebäude den geeignetsten Weg zu seinem Zielort zu finden.

Wichtig ist, dass die Menschen welchen den Player bedienen kein großes technisches Know-how benötigen müssen.

Alle weiteren Begründungen entstammen aus dem vorherigen Unterkriterium **Usability in Bezug auf die Bedienung des Backends**.

Um die Bewertung durchzuführen und Punkte vergeben zu können, für dieses Kriterium, muss die Software unter die Lupe genommen und getestet werden, sodass gesagt werden kann, ob die Software auf den ersten Blick ansprechend und diese auch einfach zu verstehen ist, oder nicht.

Tabelle mit Punkten:

Unterkriterium	Punkte
Zufriedenheit mit der Frontendanwendung	5 Punkte
Zufriedenheit mit der Backendanwendung	5 Punkte

Tabelle 4: Usability

2.5 Multi-Monitor

Eine Multi-Monitor-Lösung wird sicher die Aufmerksamkeit von mehr Menschen an sich ziehen.

Folgende zwei Arten von Multi-Monitor-Lösungen unterscheiden sich von einander:

- Videowall
- Bildschirmsynchronisierung

Die Videowall ermöglicht es beispielsweise ein Bild auf mehrere Bildschirme so aufzuteilen, dass ein riesiges Bild auf mehreren Monitoren entsteht. Dies zieht die Aufmerksamkeit der Menschen an sich, da eine viel größere Fläche zur Verfügung steht, als bei einem einzelnen Bildschirm. Weiters können mit solchen Videowalls mehr Menschen, als mit einem Bildschirm, gleichzeitig erreicht werden, vorausgesetzt es sind auch mehr Menschen im

Raum. Unter der Synchronisierung von Bildschirmen versteht man, dass mehrere Bildschirme ein und denselben Inhalt wiedergeben. Die Bildschirmsynchronisierung bietet eine gute Möglichkeit dafür, Menschen an etwas zu erinnern und gewisse Inhalte, die angezeigt werden, länger im Gedächtnis zu behalten. Weiteres kann sie auch dafür verwendet werden, wichtige Informationen an Menschen weiterzugeben und diese Information auch an möglichst viele Leute zu verteilen. Wenn diese Information an mehreren Bildschirmen ausgestrahlt wird, so ist die Chance geringer, die Information nicht zu sehen, als wenn sie nur auf einem Bildschirm ausgestrahlt wird. Der Player könnte auch als einzelner mehrere Bildschirme verwalten können, falls er natürlich mehrere Videoausgänge hat. Wenn seine Grafikkarte aber für das nicht ausgelegt ist, empfehlen sich mehrere Player die gleichzeitig dieselbe Playlist abspielen. vgl. [21]

Wenn beide oben angeführten Möglichkeiten implementiert sind, so ist die höchste Punktezahl für das Produkt zu vergeben. Kann es sogar mehr ist es selbstverständlich, dass Zusatzpunkte vergeben werden.

Tabelle mit Punkten:

Unterkriterium	Punkte
Videowall	5 Punkte
Bildschirmsynchronisierung	5 Punkte

Tabelle 5: Multi-Monitor

2.6 Medienformate

Eine wesentliche Voraussetzung für die reibungslose Erstellung von Playlists ist die Unterstützung der gängigen Medienformate. Bei unseren Untersuchungen haben wir herausgefunden, dass folgende Medienformate als MUSS-Kriterium zu unterstützen sind:

- Videoformate
- Bildformate
- andere Dateiformate

2.6.1 Arten von Videoformaten

- WMV
- MPEG
- AVI
- FLV

Windows Media Video

Windows Media Video kurz WMV ist ein proprietäres Videoformat von Microsoft und Teil der Windows-Media-Plattform. Neben dem Windows Media Player und vielen anderen Software-Playern beherrschen auch einige eigenständige Hardware-Abspielgeräte das Dekodieren von WMV-Dateien.
vgl.[27]

Moving Picture Experts Group

Die Moving Picture Experts Group (ist eine Gruppe von Experten, die sich mit der Standardisierung von Videokompression und den dazugehörigen Bereichen, wie Audiodatenkompression oder Containerformaten, beschäftigt. Umgangssprachlich wird mit „MPEG“ meistens nicht die Expertengruppe, sondern ein spezieller MPEG-Standard bezeichnet.. Etwa 350 Experten aus 200 Unternehmen und Organisationen aus 20 verschiedenen Ländern nehmen an diesen Treffen, den MPEG-Meetings, teil. vgl.[10]

MPEG I + MPEG II (.mpg)

MPEG-1 ist ein Standard der Moving Picture Experts Group (MPEG) zur verlustbehafteten Video- und Audiodatenkompression. MPEG-1 wurde in den 1980er Jahren mit dem Ziel entwickelt (1991 vorgestellt), Filme auf die beschränkte Datenrate (bis 1,5 Mbit/s) einer mit normaler Geschwindigkeit abgespielten Audio-CD zu komprimieren. Das Ergebnis, mit dementsprechend eher bescheidener Qualität, wird Video-CD genannt.[11]

MPEG-4 (.mp4)

MPEG-4 ist ein MPEG-Standard, der unter anderem Verfahren zur Video- und Audiodatenkompression beschreibt. Ursprünglich war das Ziel von MPEG-4, Systeme mit geringen Ressourcen oder schmalen Bandbreiten (Mobiltelefon, Video-Telefon, ...) bei relativ geringen Qualitätseinbußen zu unterstützen. . vgl.[12]

Audi Video Interleave (.avi)

Audio Video Interleave (AVI) ist ein vom Softwarehersteller Microsoft definiertes Video-Containerformat, das von dem für Windows 3.1 eingeführten RIFF (Resource Interchange File Format) abgeleitet ist. „Audio Video Interleave“ bedeutet, dass Audio- und Videodaten ineinander verzahnt, also „interleaved“ abgespeichert werden. Die erste Definition von AVI ist so alt wie die Multimedia-PCs. Das Format wurde 1992 von Microsoft im Rahmen der Video for Windows Programmierschnittstelle geschaffen, um die Wiedergabe von audiovisuellen Daten innerhalb Windows zu ermöglichen. vgl.[?]

Flash Video (.flv oder .swf)

Flash Video (FLV) ist ein von Adobe Systems entwickeltes offenes Containerformat, das vornehmlich für Internetübertragungen von Videoinhalten genutzt wird. Es ist je nach verwendetem Codec kompatibel zu den Adobe Flash Playern ab Version 6 bzw. 7 (s. u.). Zusätzlich besteht die Möglichkeit, derartige Flash-Video-Dateien in einer SWF-Datei eingebettet zu laden. Derzeit unterstützt der Adobe Flash Player, und damit auch das Flash-Video-Format, die Video-Codecs Sorenson (eine MPEG-4 ASP H.263-Codec Variante), VP6 (von On2) und MPEG-4 nach dem H.264-Standard. vgl.[3]

2.6.2 Arten von Bildformaten

- JPEG

-
- PNG
 - GIF
 - BMP

Joint Photographic Experts Group (.jpg)

PEG (ist die gebräuchliche Bezeichnung für die 1992 vorgestellte Norm ISO/IEC 10918-1 bzw. CCITT Recommendation T.81, die verschiedene Methoden der Bildkompression beschreibt. Die Bezeichnung „JPEG“ geht auf das Gremium Joint Photographic Experts Group zurück, das die JPEG-Norm entwickelt hat. JPEG schlägt verschiedene Komprimierungs- und Kodierungsmethoden vor, darunter verlustbehaftete und verlustfreie Kompression, verschiedene Farbtiefen sowie sequenzielle oder progressive Modi (normaler Bildaufbau bzw. allmähliche Verfeinerung). Die JPEG-Norm beschreibt lediglich Bildkompressionsverfahren, legt aber nicht fest, wie die so entstandenen Daten gespeichert werden sollen. Gemeinhin werden mit „JPEG-Dateien“ oder „JPG-Dateien“ Dateien im Grafikformat JPEG File Interchange Format (JFIF) bezeichnet. JFIF ist jedoch nur eine Art, JPEG-Daten abzulegen; SPIFF und JNG sind weitere, wenn auch wenig gebräuchliche, Möglichkeiten. vgl.[6]

Portable Network Graphics (.png)

Portable Network Graphics ist ein universelles, vom World Wide Web Consortium (W3C) anerkanntes Grafikformat für Rastergrafiken mit verlustfreier Kompression. Es wurde als freier Ersatz für das ältere, bis zum Jahr 2006 mit Patentforderungen belastete Graphics Interchange Format (GIF) entworfen und ist weniger komplex als das Tagged Image File Format (TIFF). PNG ist das meistverwendete verlustfreie Grafikformat im Internet. vgl.[14]

Graphics Interchange Format (.gif)

GIF für Graphics Interchange Format (engl. Grafikaustausch-Format) ist ein Grafikformat für Bilder mit Farbpalette. Es erlaubt eine verlustfreie

Kompression der Bilder. Darüber hinaus können mehrere (übereinanderliegende) Einzelbilder in einer Datei abgespeichert werden, die von geeigneten Betrachtungsprogrammen wie Webbrowsern als Animationen interpretiert werden.[4]

Windows Bitmap (.bmp oder .dib)

Windows Bitmap („BMP“) oder device-independent bitmap (DIB) ist ein zweidimensionales Rastergrafikformat, das für die Betriebssysteme Microsoft Windows und OS/2 entwickelt und mit Microsoft Windows 3.0 eingeführt wurde, welches 1990 erschien. Die Dateierweiterung ist .bmp, seltener .dib. BMPs gibt es in drei verschiedenen Versionen. Die meisten BMP-Dateien liegen in der Version 3 vor; es gibt keine früheren Versionen. vgl.[16]

2.6.3 Andere Dateiformate

- PDF
- Microsoft Office
 - Word
 - Excel
 - PowerPoint
- Apache OpenOffice
 - Writer
 - Calc
 - Impress

Portable Document Format (.pdf)

Das Portable Document Format (kurz PDF; deutsch: (trans)portables Dokumentenformat) ist ein plattformunabhängiges Dateiformat für Dokumente, das vom

Unternehmen Adobe Systems entwickelt und 1993 veröffentlicht wurde. Ziel war es, ein Dateiformat für elektronische Dokumente zu schaffen, das diese unabhängig vom ursprünglichen Anwendungsprogramm, vom Betriebssystem oder von der Hardwareplattform originalgetreu weitergeben kann. Das Ziel wurde erreicht und findet seinen Niederschlag in einer Normenserie der ISO (ISO 15930 für PDF/X). Hierzu griff man wesentlich auf die Funktionsweise des PostScript-Formats zurück. Ein Leser einer PDF-Datei soll das Dokument immer in der Form betrachten und ausdrucken können, die der Autor festgelegt hat. vgl.[13]

Microsoft Office

Microsoft Word

Die für Microsoft Word bis zur Version 2003 bzw. 2004 verwendeten Dateinamenerweiterungen heißen .doc für Dokumente und .dot für Dokumentenvorlagen. Bemerkenswert dabei ist die Inkompatibilität insbesondere der Versionen bis einschließlich 7 (Microsoft Office 95) und der Versionen ab 8 (Microsoft Office 97). Sie beruht darauf, dass Microsoft die Backup-Funktionalität grundlegend änderte, was der Einführung eines neuen Dateiformates gleichkam, auf eine Änderung der bekannten Dateinamenerweiterung jedoch verzichtete. Word kann mit verschiedenen Dateiformaten arbeiten, jedoch wurden einige wichtige Fremdformate bis zum Service Pack 2 von Word 2007 beziehungsweise Word 2010 für Windows auf Word für Mac OS X bis heute nicht unterstützt. vgl.[9]

Microsoft Excel

Microsoft Excel ab 2007 und 2008 erstellen standardmäßig Dateien mit den Dateierweiterungen .xlsx (Excel Spreadsheet) oder .xlsm (Excel Spreadsheet mit Makros). Zusätzlich wird eine Reihe von Dateierweiterungen wie .xlsb (platzsparendes Binärformat), .xlam (Excel Add-ins), .xltx (Excel-Vorlagen), .xltm (Excel-Sicherungskopien) und .xll (Excel-Makrobibliothek) verwendet. Einige der neuen Dateiformate basieren auf dem offenen OpenXML, das auf XML basiert und eine Zip-Kompression nutzt. vgl.[7]

Microsoft PowerPoint

Nachdem Microsoft bisher das proprietäre, nicht offen dokumentierte Format .ppt verwendete, werden die Präsentationen seit PowerPoint 2007 bzw. 2008 standardmäßig im Office Open XML-Format (.pptx) gespeichert. Die erstellten Präsentationen lassen sich auch als Webseiten (*.html) oder als einzelne Folien in den gängigsten Grafikformaten ausgeben. Neben dem Standard-Präsentationsformat (.ppt bzw. .pptx und .pptm für Dateien mit Makros, letzteres nur Windows) wird häufig das Bildschirmpräsentationsformat (.pps bzw. .ppsx) verwendet, das der sofortigen Anzeige der Präsentation im Vollbildmodus dient. Zwischen den beiden Formaten besteht inhaltlich kein Unterschied; es werden die gleichen Informationen gespeichert. Zur Anzeige der Präsentationen muss entweder die Vollversion des Programms oder der kostenlose PowerPoint Viewer für Windows auf dem Computer vorhanden sein. vgl.[8]

Apache OpenOffice

Writer

Mit Writer können sowohl kurze Texte wie Briefe, Serienbriefe, Memos, Etiketten, Visitenkarten als auch umfangreiche Schriften wie Bücher oder mehrteilige Dokumente mit Tabellen sowie Inhalts- und Literaturverzeichnissengeschrieben und gestaltet werden. Die Textverarbeitung bietet gängige Funktionen wie Textbausteine, Teamfunktionen, Rechtschreibprüfung, Silbentrennung, Thesaurus, Autokorrektur, mehrstufiges Undo sowie verschiedene Dokumentvorlagen. vgl.[17]

Calc

In Calc werden Daten in Tabellen bearbeitet, analysiert, verwaltet und verdeutlicht. Daten können angeordnet, gespeichert und gefiltert werden. Die Tabellenkalkulation bietet über 450 Berechnungsfunktionen etwa aus den Bereichen elementare Mathematik, Finanzen, Statistik, Datum und Zeit. Es steht ein Funktionsassistent zum Erstellen von Formeln und komplexen Berechnungen einschließlich Vektoralgebra (Matrizen-Rechnen) zur Verfügung. Mit externen Add-ins lassen sich weitere Funktionen nachrüsten. vgl.[1]

Impress

Mit Impress können Vortragsfolien mit Animationen und verschiedenen Hintergründen erstellt werden. Präsentationen können mit Diagrammen, Zeichenobjekten, Multimedia- und vielen anderen Elementen versehen werden. Einzelnen Folien können unterschiedliche Übergangseffekte zugeordnet werden. vgl.[5]

Dateiformate

Das Dateiformat von OpenOffice.org wurde von der Organization for the Advancement of Structured Information Standards (OASIS) als Basis für das neue offene Austauschformat OpenDocument verwendet, welches das Standardformat von OpenOffice.org ab Version 2.0 ist. Die XML-Dateien sind gepackt und belegen deshalb sehr wenig Speicherplatz. Die Dokumentinhalte werden im Java-Archive-Format gespeichert, einer ZIP-Datei mit speziellen Einträgen. Die Dateiendung eines Java-Archivs ist normalerweise „.jar“, jedoch werden für OpenDocument-Dateien die Dateiendungen „.od*“ verwendet. vgl.[?]

Tabelle mit Punkten:

Unterkriterium	Punkte
Pro Dateiformat	1 Punkt (max. 10 Punkte)

Tabelle 6: Medienformate

2.7 Touchfunktion

Ein zusätzliches Feature von Digital Signage Systemen kann die sogenannte Touchfunktion sein. Damit wird es ermöglicht, dass der Zuseher auch direkt mit dem System in Interaktion treten kann. So ist es zum Beispiel möglich eine Digital Signage Playlist zu unterbrechen und direkt am Bildschirm die Route zu einem bestimmten Punkt im Objekt zu finden (Wayfinder).

Für das Vorhandensein einer Touchfunktion werden x Punkte vergeben.

Die höchste Punktzahl ist an die Produkte zu vergeben, welche ein Leitsystem in ihrer Software haben. Die Punktzahl für die Produkte die keine Touchfunktion haben, fällt mit 0 Punkten schlecht aus.

Tabelle mit Punkten:

Unterkriterium	Punkte
Touchfunktion vorhanden	10 Punkte

Tabelle 7: Touchfunktion

3 Verwendete Technologien

3.1 Systemarchitektur

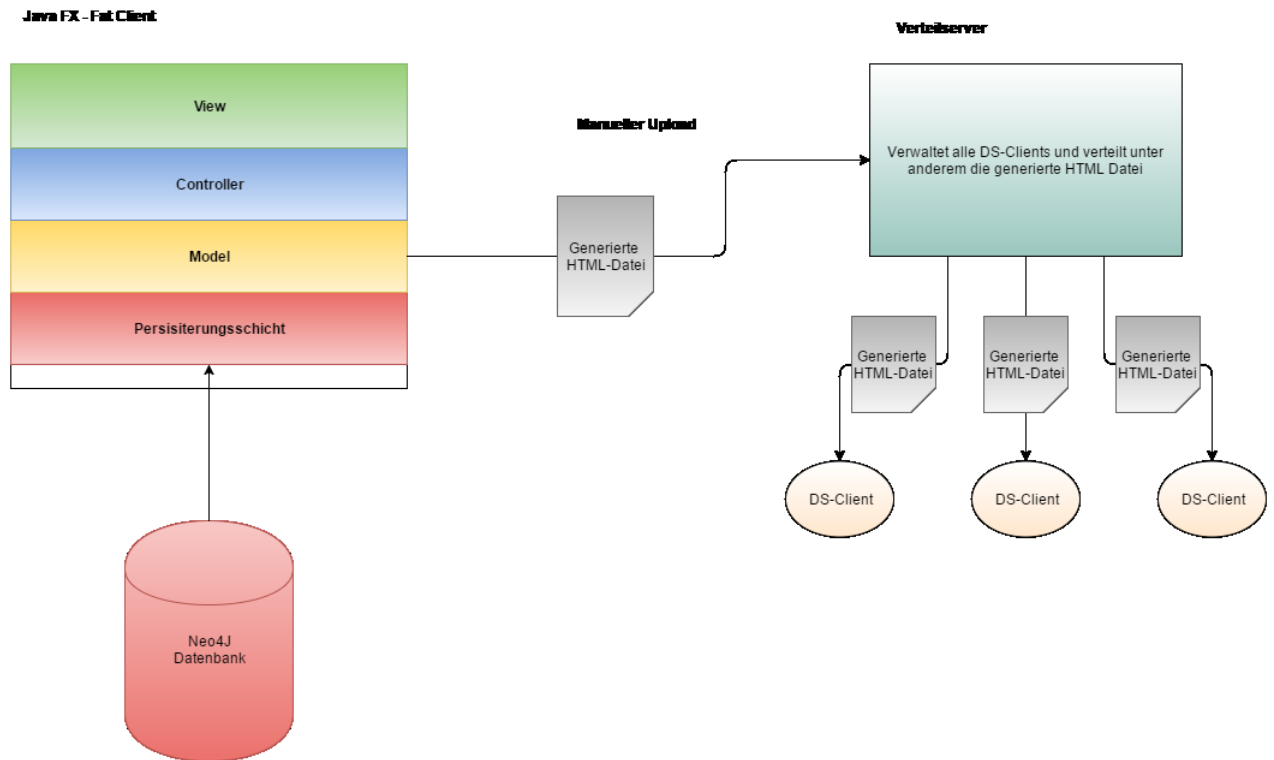


Abbildung 3: Systemarchitektur

3.2 Hypertext Markup Language

Hypertext Markup Language, kurz HTML, ist eine textbasierte Auszeichnungssprache zur Strukturierung von digitalen Dokumenten. HTML-Dateien werden von Webbrowsern dargestellt und sind die Grundlage des World Wide Web. HTML-Dateien können auch Metainformationen enthalten wie zum Beispiel die verwendete Sprache.[20] Im JavaScript-Client wird die Version HTML5 verwendet.

3.2.1 Nutzung

Um eine HTML-Datei erstellen zu können, ist grundsätzlich nicht viel notwendig. Wichtig ist, dass die Dateiendung **.html** sein muss. So kann der Browser dann erkennen, dass es sich um eine HTML-Datei handelt und diese HTML-Datei auch so auswerten.

Der grundlegende Aufbau einer HTML-Datei sieht wie folgt aus.

```
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

3.2.2 Anwendung

HTML-Dateien bestehen aus Tags. Tags bestehen aus einer spitzen Klammer nach links, dem Namen des Tags und einer spitzen Klammer nach rechts. Jeder Tag muss auch immer geschlossen werden. Tags werden geschlossen, indem vor den Tagnamen ein Frontslash gesetzt wird. Tags die in einem anderen Tag geöffnet werden, müssen auch in dem Tag geschlossen werden. Das äußerste Element jeder HTML-Datei ist der Tag **<html>**. Dieses Element beinhaltet alle Elemente, die in der HTML-Datei dargestellt werden sollen und auch die Metainformationen. Die Metainformationen stehen in dem Element **<head>**. In dem Element **<head>** findet man auch alle Verweise auf die Gestaltungsvorlagen für diese Datei. Das Element **body** beinhaltet die Elemente die in der Datei dargestellt werden. Ein weiteres wichtiges Element ist **<div>**. Das **<div>**-Element ist ein Container und fasst zusammengehörende Elemente zusammen. So bewahrt man den Überblick über seine Datei. Jeder Tag hat verschiedene Attribute die man definieren kann. Attribute die jedes Element hat sind zum Beispiel **class** und **id**. Mit den Attributen der Elemente kann man ihnen beispielsweise Namen geben, den anzuzeigenden Text setzen, Verweise auf andere Seiten machen und vieles mehr. Um eine HTML-Datei mit der Version HTML5 zu erstellen setzt man den Tag **<!DOCTYPE HTML>** vor den Tag **<html>**. Dieses Tag muss aber nicht geschlossen werden.

3.3 JavaScript

JavaScript, kurz JS, ist eine Skriptsprache die für dynamische HTML-Dateien entwickelt wurde. Die Skriptsprache ermöglicht es, Inhalte zu verändern oder auch Benutzeraktionen auszuwerten.[26] ECMAScript ist die Standardisierung von JavaScript. Im JavaScript-Client wird ECMAScript 5 genutzt.

3.3.1 Nutzung

Eine JavaScript-Datei hört mit der Dateiendung **.js** auf. Um die JS-Datei auch wirklich nutzen zu können, muss man diese in eine HTML-Datei einbinden. Dies kann entweder im **<body>** oder im **<head>** Element geschehen. Die Stelle an der die JS-Datei im Programmcode ist, ist insofern wichtig, da die Datei erst ab dieser Stelle ausgeführt wird.

Die Einbindung der JS-Datei geschieht über das Attribut **src** des **<script>**-Elementes und sieht wie folgt aus:

```
<!-- Einbindung im Element <head> -->
<head>
  <script src="Gesamter Pfad der JS-Datei oder Pfad im Projekt">
  </script>
</head>

<!-- Einbindung im Element <body> -->
<body>
  <script src="Gesamter Pfad der JS-Datei oder Pfad im Projekt">
  </script>
</body>
```

3.3.2 Anwendung

Wichtig ist zu wissen, dass es in JavaScript mit der Version ECMAScript 5 nur den Typ **var** gibt. Sobald man den Typ var aber initialisiert, wird er zu einem bestimmten Typen, wie zum Beispiel zu einer Zeichenkette, einer Nummer oder einem Array. Über die Variable **document** bekommt man zum Beispiel Elemente über deren ID aus der HTML-Datei. Diese Elemente

kann man dann in der JS-Datei verändern. Die Variable **document** muss man nicht selbst deklarieren.

Das nachfolgende Codebeispiel soll JavaScript so einfach wie möglich erläutern:

```
//Aus der HTML-Datei das Element mit der Id Element holen
var element = document.getElementById("element");

//Den Text der variable element setzen
function setTextInElement(dieserText){
    element.text = dieserText;
}

//Die Funktion aufrufen
setTextInElement("Hallo Welt");
```

3.4 Cascading Stylesheet

Cascading Stylesheet, kurz CSS, ist eine Stylesheet-Sprache für elektronische Dokumente. CSS ist mit HTML und DOM eine der Kernsprachen des World Wide Webs. Mit CSS werden Gestaltungsvorlagen für Auszeichnungssprachen wie HTML erstellt.[24] In JS-Client wird die Version CSS3 benutzt.

3.4.1 Nutzung

Eine CSS-Datei hört mit der Dateierweiterung .css auf. Um sie auch wirklich verwenden zu können, muss man die Datei in eine Auszeichnungssprache wie HTML einbinden. In einer HTML-Datei bindet man die CSS-Datei im Element **<head>** ein. Im Attribut **href** des Elementes **<link>** wird dann die Datei in die HTML-Datei eingebunden. Wichtig ist, dass im Attribut **rel** das Wort **stylesheet** steht. Das Element **<link>** darf nicht geschlossen werden.

Es folgt ein Beispiel wie man eine CSS-Datei in eine HTML-Datei einbindet.

```
<head>
  <link rel="stylesheet"
```

```
href="Gesamter Pfad oder der Pfad im Projekt der CSS-Datei">
</head>
```

3.4.2 Anwendung

In der CSS-Datei kann man auf die Elemente in der HTML-Datei über den Tagnamen, das Attribut **class** oder das Attribut **id** zugreifen. Über das Attribut **style** kann man Designvorlagen direkt im HTML-Code erstellen. Dies ist aber sehr unübersichtlich wenn man eine Vielzahl an Designänderungen durchführen will. Um mit einer prozentuellen Größenangabe der Breite und der Höhe bei den Elementen in der HTML-Datei arbeiten zu können, muss man die Höhe und Breite der Elemente **<html>** und **<body>** auf 100% setzen.

Im folgenden Beispiel wird der Aufbau der Syntax erklärt:

```
/*Zugriff auf den Tagnamen des Elementes */
html{
/*Hier können Designvorlagen gesetzt werden*/
    width: 100%;
    height: 100%;
}

/*Zugriff über das Attribut ID des Elementes*/
#elementId{
    margin: 10px;
}

/*Zugriff über das Attribut class des Elementes*/
.elementClass{
    color: red;
}
```

3.5 JQuery

JQuery ist eine freie JavaScript Bibliothek. Sie ist sehr schnell und funktionsreich. Sie macht Animationen, Event Handling, Ajax und DOM-Manipulationen viel einfacher. Die jQuery API ist einfach zu nutzen

und funktioniert über viele Browser hinweg. Ein Großteil der Webseiten die es im Internet gibt, benutzen diese Bibliothek. Sie ist die meistverwendete JavaScript-Bibliothek.[34]

3.5.1 Implementierung

Die jQuery-Implementierung erfolgt entweder über einen Link auf eine jQuery CDN oder indem man die jQuery-Dateien herunterlädt. Die heruntergeladene Datei, beziehungsweise den Link, setzt man dann im HTML-Tag `<body>`. In diesem HTML-Tag setzt man ein Tag `<script>`. Dieses HTML-Element wird am Ende des bereits erwähnten `<body>` Elements versetzt. Im nachfolgenden Code sieht man ein Beispiel für einen Link auf eine jQuery CDN.

```
<body style="width: 100%; height: 100%; text-align: center; ">

  <!--Hier würde der HTML-Code stehen-->

  <script
    src="https://cdn.jsdelivr.net/g/
    jquery.jcanvas@13.04.26(jcanvas.min.js+jcanvas.js)">
  </script>
</body>
```

3.5.2 Anwendung

Man kann jQuery-Code von JavaScript code sehr einfach unterscheiden, da jQuery mit `$('objekt')` auf ein Element in der HTML-Datei verweist und JavaScript dies zum Beispiel mit

`document.getElementById('objektID')` löst. Es ist zu erwähnen, dass das `$` vor den Klammern wichtig ist, da der Compiler so erkennt, dass ab diesem Zeichen bis zum nächsten Strichpunkt jQuery-Code folgt. Auf das über jQuery erhaltene Element kann man somit alle jQuery spezifischen Funktionen ausführen wie zum Beispiel die `.click` Funktion. Es folgt ein Beispiel für die Erläuterung des Unterschiedes zwischen jQuery und JavaScript.

```
//Mit jQuery ein Element aus dem HTML-Code über das Attribut id
var element = $('#element');
```

```
//Mit JavaScript ein Element aus dem HTML-Code über das Attribut id
var element = document.getElementById("element");
```

3.5.3 Benutzte Funktionen

Im Gegensatz zu JavaScript hat jQuery den Vorteil, dass man meistens weniger Code für das gleiche Ergebnis braucht. Die wichtigsten benutzten Funktionen sind die `.click`, `.css`, `.text` und die `.empty` Funktion. Weitere verwendete Funktionen werden im Abschnitt JCanvas noch genauer erläutert. Die `.click` Funktion ermöglicht es ein Element, welches vorher über die ID, die Klasse oder den Tagname ausgesucht wurde, klickbar zu

machen. Die `.empty` Funktion leert den Inhalt des ausgesuchten Elementes. Die `.css` Funktion ermöglicht es, das Aussehen der Webseite dynamisch zu ändern. Die `.text` Funktion liefert den Text eines Elements oder setzt einen Text für ein Element. Den Vorteil, den jQuery gegenüber von JavaScript hat, sieht man gut an folgendem Beispiel.

```
//.click Funktion von jQuery auf den Button mit der ID element.  
//Die Funktion ruft die Methode someFunction auf.  
$("#element").click(someFunction);  
  
//.click Funktion von JavaScript auf den Button mit der ID element.  
//Die Funktion ruft die Methode someFunction auf.  
document.getElementById("element")  
.addEventListener('click',someFunction);  
  
//.empty Funktion von jQuery  
$("#element").empty();  
  
//.css Funktion für die Größe des Hintergrundes von jQuery  
$("#element").css("background-size", "100% 100%");  
  
//.text Funktion von jQuery für das Auslesen eines Textes  
$("#element").text();
```

3.5.4 JCanvas

Das `<canvas>` Element ist ein neues Element in HTML5. Es erlaubt dem Entwickler auf einem Element mit dem Namen Canvas Formen, Pfade, Bilder und andere Dinge zu zeichnen.[30] Die jCanvas-Bibliothek beinhaltet einige nützliche Funktionen für dieses neue HTML5 Element. Alle jCanvas-Methoden sind reguläre jQuery-Methoden, daher haben sie auch die gleiche Syntax.[31] Im JavaScript-Client wurde jCanvas verwendet, um den Pfad im Canvas-Element zu zeichnen. Der Pfad besteht aus einem Anfangs- und einem Endknoten. Zwischen diesen Knoten sind Linien. Die zwei Funktionen von jCanvas die im Client benötigt wurden, sind daher die `.drawArc` und die `.drawLine` Funktionen.

```

$("#canvas").drawLine({
    strokeStyle: '#000',
    strokeWidth: 5,
    x1: getKnotById(sPath[i])['xCoordinate']*faktorW,
    y1: getKnotById(sPath[i])['yCoordinate']*faktorH,
    x2: getKnotById(sPath[i + 1])['xCoordinate']*faktorW,
    y2: getKnotById(sPath[i + 1])['yCoordinate']*faktorH
});

$("#canvas").drawArc({
    fillStyle: '#000',
    x: getKnotById(sPath[i + 1])['xCoordinate']*faktorW,
    y: getKnotById(sPath[i + 1])['yCoordinate']*faktorH,
    radius: 2
});

```

3.6 Bootstrap

Bootstrap ist ein freies CSS-Framework. Es enthält Gestaltungsvorlagen für viele verschiedene HTML Elemente. Bootstrap enthält ebenfalls optionale JavaScript-Erweiterungen, doch die Gestaltungsvorlagen sind das Hauptmerkmal des Frameworks.[23] Es erleichtert die Arbeit mit HTML um einiges, da man keine eigenen komplexen Stylesheets mehr schreiben muss, um auch ein ansprechendes Design zu bekommen. Bootstrap ist ein Framework, das aus einer Tabelle mit 12 Spalten besteht. Dieses Tabellensystem erleichtert dem Programmierer die Entwicklung von responsive Webseiten. Bootstrap basiert ebenfalls auf einem Klassensystem, was bedeutet, dass der Entwickler über das Attribut class das Design des jeweiligen Elements ändern kann.

3.6.1 Implementierung

Um Bootstrap implementieren zu können, muss man entweder den Link auf die Bootstrap CDN oder die heruntergeladene Datei in die HTML-Datei einfügen. Dies wird im Element `<link>` in dem Attribut href eingefügt, wobei das Element `<link>` im Element `<header>` sein muss. In der folgenden Abbildung sieht man ein Beispiel für einen Bootstrap CDN Link, der in die HTML-Datei eingebunden wurde.

```
<head>
  <link
    rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.6/css/bootstrap.min.css"
    integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8h
DDdjZlpLegxhjVME1fgjWPGmkzs7"
    crossorigin="anonymous">
</head>
```

3.6.2 Anwendung

Bootstrap besteht aus vielen Designvorlagen für viele verschiedene Elemente. Um die Designvorlagen einbinden zu können, muss man im Attribut class eines Elements gewisse Namen eingeben. Diese Namen sind von Bootstrap vorgegeben und durch die Namen kommt man zu einem entsprechenden Design. Neben den Namen für das Design gibt es ebenfalls Namen für eine responsive Webseite. Eine responsive Webseite erhält man in Bootstrap über Spalten des Bootstrap Gitters. Bootstrap teilt die Ganze Webseite in ein Gitter mit 12 Spalten, wobei das Gitter mit schrumpfenden Webseiten verändert werden kann. In jeder Spalte kann man weitere Gitter mit 12 Spalten einsetzen. Da die Spalten mit kleinerer Größe des Browsers schrumpfen, verkleinern sich auch die Elemente in den Spalten und man erhält ein ansprechbares Design auch auf kleineren Geräten wie einem Smartphone oder einem Tablet-PC. Die folgende Abbildung zeigt Beispiele für Designvorlagen die von Bootstrap gestellt sind. Alle Texte in dem Attribut class sind Bootstrap-Namen.

```
<!-- Die Klasse btn-info gibt dem Button eine blaue Farbe -->

<button id="back" class="btn btn-info">Zurück</button>
<button id="next" class="btn btn-info">Weiter</button>
```

3.6.3 Verwendete Klassen

Die benutzten Klassen im JavaScript-Client sind einerseits **clearfix** und andererseits **btn**. Die btn Klasse beinhaltet weitere Klassen wie **btn-info**

oder **btn-danger**. Die Klasse `btn` designt ein Element in Form eines Buttons. Die Unterklassen `btn-info` oder `btn-danger` dienen nur zur Farbänderung des Elements. Um die Unterklassen wie `btn-info` in das Attribut `class` einfügen und nutzen zu können, muss man vorher in dem Attribut `class` die Klasse `btn` einfügen. Die Klasse `clearfix` ist für das Löschen der vorher gesetzten Floats vor dem Element mit dieser Klasse zuständig. `Clearfix` wird nur dann benötigt, wenn man weitere Floats setzen will.

```
<!-- Die btn Klasse mit einer der verschiedenen Unterklassen. -->
<button id="element" class="btn btn-success">Text</button>

<!-- Die clearfix Klasse zum Löschen der vorherigen Floats -->
<div class="clearfix">
    <!--Auf die Elemente hier kann man neue Floats setzen -->
    <p>Text</p>
</div>
```

3.7 Afterburner - FX

3.7.1 Was ist Afterburner - FX?

Afterburner - FX implementiert die Konventionen des gängigen JavaFX, verfügt zusätzlich aber über die Möglichkeit per **@Injection** im View, Controller und im Model Objekte zu injizieren. Das Laden von FXML, CSS und Property Files, das Instantieren des Controllers, und das Injizieren von Models und Services, kann durch die Vererbung von **com.airhacks.afterburner.views.FXMLView** ermöglicht werden.

vgl. [29]

3.7.2 Konvention

Im Gegensatz zur normalen JavaFX Konvention, wo für Views, Controllers und Models eigene Packages existieren, werden bei Afterburner - FX alle zusammengehörigen Views, FXML Dateien und Controllers (in Afterburner - FX Presenter genannt), in ein Package zusammengefasst. Da auf Models

per Injection zugegriffen wird, können diese wie gewohnt separat gespeichert werden.

vgl. [29]

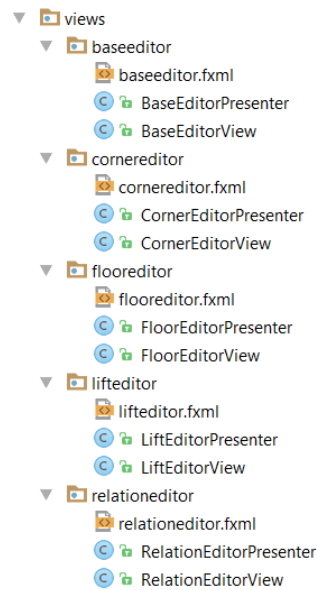


Abbildung 4: ECB - Pattern (Entity - Control - Boundary)

3.7.3 Implementierung

Afterburner - FX ist in der Maven-Zentrale vorhanden und kann über folgenden Code in ein Projekt eingebunden werden.

Pom.xml

```
<dependency>
  <groupId>com.airhacks</groupId>
  <artifactId>afterburner.fx</artifactId>
  <version>1.6.1-SNAPSHOT</version>
</dependency>
```

vgl. [28]

In einem eigenen Package können nun 3 Dateien erstellt werden. (2 Java Dateien und 1 FXML Datei) Die zwei Java Klassen tragen den Namen der

FXML Datei und zusätzlich wird noch Presenter bzw. View hinten angefügt.

Der Inhalt der View Klasse kann leer gelassen werden, die Funktionalität wird von **com.airhacks.afterburner.views.FXMLView** vererbt.

```
import com.airhacks.afterburner.views.FXMLView;
public class BaseEditorView extends FXMLView {
}
```

Der Presenter kann als gewöhnlicher Controller implementiert werden, mit dem Interface Initializable.

```
public class BaseEditorPresenter implements Initializable {

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        //Befehle die beim Laden der View auszuführen sind
    }

}
```

In diesen Controller kann auch ein Model injiziert werden. Der Vorteil eines **@Inject**'s ist, dass alle Klassen, die dieselbe Klasse injizieren, mit dem gleichen Objekt arbeiten.

```
public class BaseEditorPresenter implements Initializable {

    @Inject
    private EditorModel model;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        //Befehle die beim Laden der View auszuführen sind
    }

}
```

Das Starten eines JavaFX Programms mit Afterburner - FX ist fast identisch zum konventionellen Weg, benötigt aber weniger Code. In der

überladenen Methode **start()** der Application Klasse, muss eine neue Instanz der View Klasse erstellt werden. Mit der Methode **getView()** des View - Objekts kann die FXML Datei geladen werden, um die **Stage** zu befüllen.

```
public class App extends Application {

    @Override
    public void start(Stage stage) throws Exception
    {
        BaseEditorView view = new BaseEditorView();
        Scene baseScene = new Scene(view.getView());
        stage.setScene(baseScene);
        stage.show();
    }
}
```

Nicht nur beim Starten eines JavaFX Programms bietet Afterburner - FX eine saubere und übersichtliche Lösung, sondern auch das Austauschen einzelner Sub - Views, wird damit erleichtert.

```
public class BaseEditorPresenter implements Initializable {

    @FXML
    private AnchorPane subViewPane;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        SubEditorView subView = new SubEditorView();
        subViewPane.getChildren().clear(); //Löscht vorherige View
        subViewPane.getChildren().add(subView.getView());
        //Nun wurde das Design des SubEditorViews innerhalb der
        SubViewPane eingefügt
    }
}
```

3.7.4 Property Injection

Eine weitere große Stärke von Afterburner - FX ist die Möglichkeit, nicht nur Models und andere Klassenobjekte zu injetieren, sondern auch Properties aus einer HashMap, Properties Datei, Datenbank etc. Dies ermöglicht eine schöne Trennung zwischen dem Ein- und Auslesen einer Properties Datei, und der eigentlichen Manipulation seiner Werte. Die **start()** und **stop()** Methoden des **App**-Objekts können sich um die Datei sorgen, während sich der Controller, durch Injection, um die Wertemanipulation kümmert.

Falls im gleichen Package einer View eine **configuration.properties** Datei existiert, werden die Key-Value Pairs automatisch geladen und können ohne separatem Einlesen injiziert werden!

```
public class App extends Application {
    @Override
    public void start(Stage stage) throws Exception {

        Properties prop = new Properties(); //Könnte auch eine HashMap
            sein
        //-- Einlesen der Properties Datei --
        Injector.setConfigurationSource(prop::get);
        BaseEditorView view = new BaseEditor();
        Scene baseScene = new Scene(view.getView());
        stage.setScene(baseScene);
        stage.show();
    }
    @Override
    public void stop(){
        //Zurückspeichern der Properties in eine Datei, DB etc.
        Injector.forgetAll();//Löschen aller Werte aus dem Injector
    }
}
```

Angenommen eine eingelesene Property würde den Key **name** beinhalten könnte man diesen wie folgt im Presenter verwenden:

```
public class BaseEditorPresenter implements Initializable {

    @Inject
    private String name;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        //Befehle die beim Laden der View auszuführen sind
    }
}
```

3.8 GSON

3.8.1 Was ist GSON?

GSON ist eine Java Bibliothek, die verwendet werden kann, um Objekte in eine JSON Repräsentation umzuwandeln. Sie kann aber auch genauso verwendet werden um aus einem JSON String ein Java Objekt zu erstellen. GSON hat nicht nur die Möglichkeit, selbst erstellte Objekte zu serialisieren, sondern auch Objekte die z.B.: von Java zur Verfügung gestellt werden.

Im Gegensatz zu anderen JSON Serialisationstechnologien, vertraut GSON nicht ausschließlich auf Annotationen, sondern kann auch ohne annotieren von Feldern, diese serialisieren. Über Annotationen ist es jedoch möglich, eine Standardserialisierung anzupassen.

vgl. [36]

3.8.2 Implementierung

GSON kann über Maven eingebunden werden.

Pom.xml

```
<dependencies>
<!-- Gson: Java to Json conversion -->
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.5=6.2</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

vgl. [36]

3.8.3 Anwendung

Um ein Objekt zu serialisieren, oder einen JSON-String zu deserialisieren, muss eines neues **Gson** Objekt erstellt werden.

```
class BagOfPrimitives {

    private int value1 = 1;
    private String value2 = "abc";

    private transient int value3 = 3;
    // "transient" verhindert, dass dieses Feld serialisiert wird.

    BagOfPrimitives() {
        // no-args constructor
    }

}
```

```
BagOfPrimitives bop = new BagOfPrimitives();
Gson gson = new Gson();
String json = gson.toJson(bop);
```

Ergebnis-JSON

```
{"value1":1,"value2":"abc"}
```

Deserialisierung

```
String json = "{\"value1\":1,\"value2\":\"abc\"}";
Gson gson = new Gson();

BagOfPrimitives bop = gson.fromJson(json,
    BagOfPrimitives.getClass());
//Hier wurde nun ein BagOfPrimitives Objekt mit den Werten des
//JSON-Strings erstellt, ohne "new" aufrufen zu müssen.
```

vgl. [36]

3.9 Neo4j

3.9.1 Was ist Neo4j?

„Neo4j ist eine quelloffene NoSQL-Datenbank, die Daten in Graphen organisiert. Sie ermöglicht die effiziente Verarbeitung unterschiedlichster Anwendungsfälle, die zum Beispiel in relationalen oder NoSQL-Datenbanken mit anderen Datenmodellen eher aufwendig sind. [...] Neo4j eignet sich für wenig strukturierte Daten, die stark vernetzt sind. Wer große Mengen stark zusammenhängender strukturierter Daten verwalten muss, ist vermutlich mit einer relationalen Datenbank besser bedient.”[32]

In dieser Datenbank werden Entitäten (Stockwerke, Räume, Stiegen etc.) in der Form von Knoten gespeichert und dargestellt. Abhängigkeiten bzw. relationale Verbindungen zwischen 2 Entitäten, werden durch gerichtete Kanten gespeichert. Jeder Knoten und auch jede Kante kann „Properties”

(Eigenschaften) besitzen (beispielsweise der Name eines Raumes oder die Länge einer Kante zwischen zwei Räumen).

Die nachfolgenden Abbildungen zeigen einen groben Überblick über die Datenstruktur einer Neo4J Datenbank:

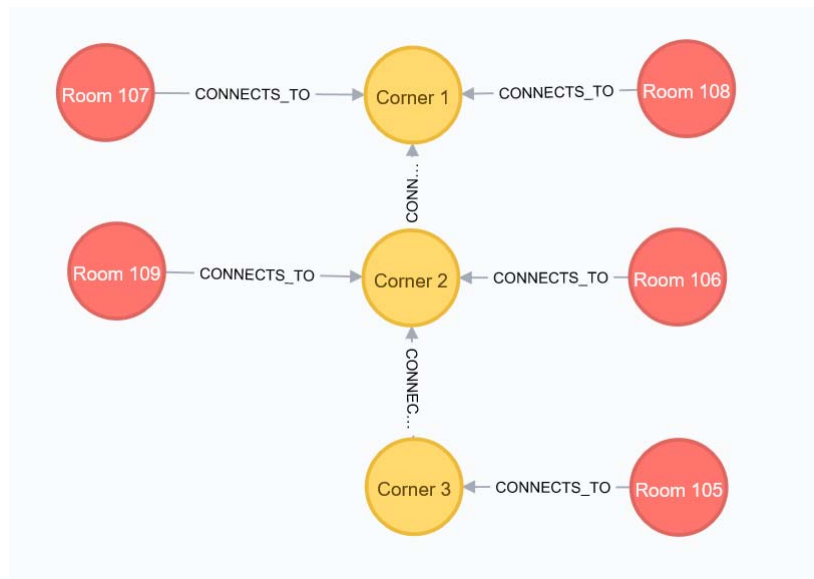
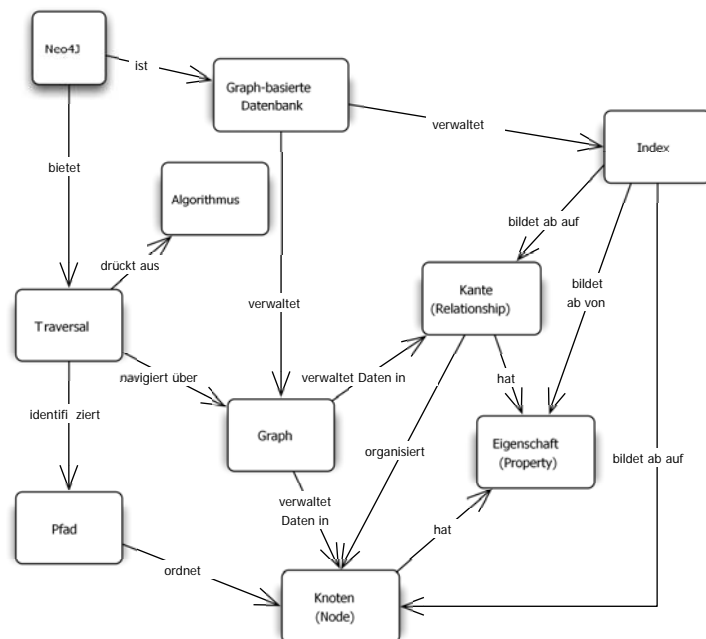


Abbildung 5: Neo4J - Darstellung



[32]

Abbildung 6: Neo4J - Überblick

Ein Vorteil der Neo4J Datenbank ist, dass Properties nicht im voraus bekannt sein müssen. Diese können im Verlauf der Datenstruktur bei zukünftigen Knoten oder Kanten hinzugefügt werden, ohne die Integrität früherer Knoten zu beeinflussen. Folgende Datentypen werden von Neo4J für die Speicherung von Properties unterstützt:

Typ	Beschreibung
boolean	true / false
byte	8-bit Integer
short	16-bit Integer
int	32-bit Integer
long	64-bit Integer
float	32-bit IEEE 754 floating-point number
double	64-bit IEEE 754 floating-point number
char	Unicode (16 bit Integer)
string	Sequenz von Unicode Charakteren

[35]

Tabelle 9: Neo4j - Property Values

3.10 Neo4j - OGM

3.10.1 Was ist Neo4j - OGM ?

Neo4j - OGM (Object Graph Mapping) ist eine von Neo4j entwickelte Bibliothek, die eine Persistierungsschicht für Java zur Verfügung stellt. Diese Schicht sorgt dafür, dass eingelesene Knoten und Relationen aus einer Neo4j Datenbank direkt einem Objekt zugeteilt werden können. Änderungen an diesen Objekten können zurück in die Datenbank gespeichert werden. Sie bietet außerdem die Möglichkeit neue Datenbankeinträge aus selbst erstellten Objekten zu erstellen, oder sogar zu löschen.

3.10.2 Implementierung

Neo4j - OGM lässt sich problemlos mithilfe von Maven, Gradle oder Ivy einbinden.

Maven

```
<dependency>
  <groupId>org.neo4j</groupId> Klassen
  <artifactId>neo4j-ogm</artifactId>
  <version>{version}</version>
</dependency>
```

Gradle

```
dependencies {
    compile 'org.neo4j:neo4j-ogm:{version}'
}
```

Ivy

```
<dependency org="org.neo4j" name="neo4j-ogm" rev="{version}"/>
```

Nach dem Import haben wir nun alle nötigen Bibliotheken um Objekte zu persistieren und Daten in der Neo4j Datenbank zu manipulieren.

Um nun eine Klasse persistierbar zu machen müssen wir eine Annotation zu den jeweiligen Klassendeklarationen hinzufügen. Somit werden diese Objekte in der Datenbank als Knoten dargestellt.

```
@NodeEntity
public abstract class MapNode extends Circle {
}
```

Jedes Objekt benötigt aber auch eine eindeutige Identifikationsnummer (Id) um in der Datenbank identifiziert werden zu können. Standardmäßig reicht es eine Long Variable mit der Bezeichnung „id“ zu erstellen.

```
@NodeEntity
public abstract class MapNode extends Circle {
    private Long id;
}
```

Sollte die Id aber einen anderen Namen haben, so muss diese Variable mit der Annotation @GraphId explizit gekennzeichnet werden.

```

@Entity
public abstract class MapNode extends Circle {
    @GraphId
    private Long id;
}

```

Ist zwischen 2 Knoten eine Relation erwünscht, so kann sie mit der Annotation **@Relationship** gekennzeichnet werden. Dabei ist zu beachten, dass jede Relation einen eindeutigen „type“ (Name der Relation) haben muss!

Wenn ein Knoten sich mit mehreren anderen Knoten über den gleichen Relationstyp verbinden lässt, so kann eine Liste (List<>) verwendet werden.

```

@Entity
public abstract class MapNode extends Circle {
    @GraphId
    private Long alternate_id;

    @Relationship(type="CONNECTS_TO")
    private List<MapNode> mapNodeList;
}

```

Zusätzlich kann mittels dem Attribut „direction“ die Richtung dieser Beziehung angegeben werden. Es gibt 3 verschiedene Schlüsselwörter um die Richtung der Relation zu beschreiben:

Direction	Beschreibung
OUTGOING (Default)	Von diesem Knoten geht die Relation aus
INCOMING	Zu diesem Knoten führt die Relation hin
UNDIRECTED	Die Richtung der Relation ist egal.

Tabelle 10: Neo4j - OGM Direction

```

@Entity
public abstract class MapNode extends Circle {
    @GraphId
    private Long id;
}

```

```
    @Relationship(type="CONNECTS_TO",
        direction=Relationship.UNDIRECTED)
    private List<MapNode> mapNodeList;
}
```

Falls eine Relation eigene Eigenschaften besitzen sollte (z.B.: die Länge eines Weges zwischen zwei Räumen oder die Stärke einer Freundschaft), muss für diese Beziehung eine eigene Klasse erstellt, und mit **@RelationshipEntity**, mit dem jeweiligen Relationstypen, annotiert werden. Außerdem müssen die verbundenen Knoten mittels den Annotationen **@StartNode** und **@EndNode** in dieser Klasse gekennzeichnet werden.

```
@RelationshipEntity(type = "CONNECTS_TO")
public class ConnectRelation extends Line{
    @GraphId
    Long relationId;

    @StartNode
    private MapNode startNode;

    @EndNode
    private MapNode endNode;
}
```

Damit das Programm nun 2 Knoten mit der neu erstellen Relationsklasse verbindet, muss die Direktbeziehung **private List<MapNode> mapNodeList;** durch **private List<ConnectRelation> mapNodeList;** ersetzt werden.

```
@NodeEntity
public abstract class MapNode extends Circle {
    @GraphId
    private Long id;

    @Relationship(type="CONNECTS_TO",
        direction=Relationship.UNDIRECTED)
    private List<ConnectRelation> mapNodeList;
}
```

Nachdem die Klassen definiert wurden kann durch das Erstellen einer **Session** auf die Datenbank zugegriffen werden. Es gibt die Möglichkeit anstatt sich über HTTP auf einen externen Server zu verbinden, eine Embedded Datenbank zu verwenden.
HTTP

```
Components.configuration()
    .driverConfiguration()
    .setDriverClassName("org.neo4j.ogm.drivers.http.driver.HttpDriver")
    .setURI("http://localhost:7474");
```

Embedded - Datenbank

```
Components.configuration()
    .driverConfiguration()
    .setDriverClassName("org.neo4j.ogm.drivers.embedded.driver.EmbeddedDriver");
    .setURI("file:///C:/Users/Cristopher/Documents/Neo4j/embeddedDB");
```

Session - Erstellung

```
SessionFactory sessionFactory = new SessionFactory(configuration,
    "org.htl.leonding.digitalsignage.wayfinder.entities");
    // Package in denen sich die Knoten und
    // Relationen befinden
Session session = sessionFactory.openSession();
```

Mit folgenden Session - Methoden können Daten geladen, persistiert oder gelöscht werden.

<code>session.loadAll(Class.class, depth);</code>	Ladet aus der Datenbank eine Liste von Knoten oder Relationen der angegebene Klasse.
<code>session.Load(Class.class, id, depth);</code>	Ladet aus der Datenbank einen bestimmten Knoten oder bestimmte Relation der angegebenen Klass mit der jeweiligen Id.
<code>session.save(Object o, depth);</code>	Speichert ein Objektes bzw. ihre Änderungen in die Datenbank.
<code>session.delete(Object o);</code>	Löscht das jeweilige Objekt aus der Datenbank.

„depth“ gibt an in welcher Entfernung Nachbarsknoten von der Methode beeinflusst bzw. mitgeladen werden. Eine Depth von 1 (Standardwert) bedeutet dass nur die nächsten Nachbarsknoten mitgeladen bzw. verändert werden.

Tabelle 11: Neo4j - OGM Befehle

4 Umsetzung

4.1 Überblick - Editor

Wayfinding - Editor ist ein Programm welches einem User erlaubt, ein Raumnetz anhand von Gebäudeplänen zu zeichnen. Sie dient dazu Stockwerke, Räume, Lifte, Stiegen etc. zu Definieren und diese zu verbinden. Sie verfügt auch über die Möglichkeit, nach dem definieren aller gewollten navigierbaren Bereiche des Gebäudes, dieses Netz als HTML-Datei (mit JS und CSS) zu exportieren. Dieser Export kann mithilfe eines Touchscreens einem Besucher des Gebäudes eine benutzerfreundliche Oberfläche bieten, um in kürzester Zeit einen Zielort zu finden.

4.2 GUI

Der Editor lässt sich in 6 Hauptbereiche teilen:

- Canvas
- Menubar
- Knoten - Wähler
- Stockwerkliste
- Editor Einstellungen
- Properties

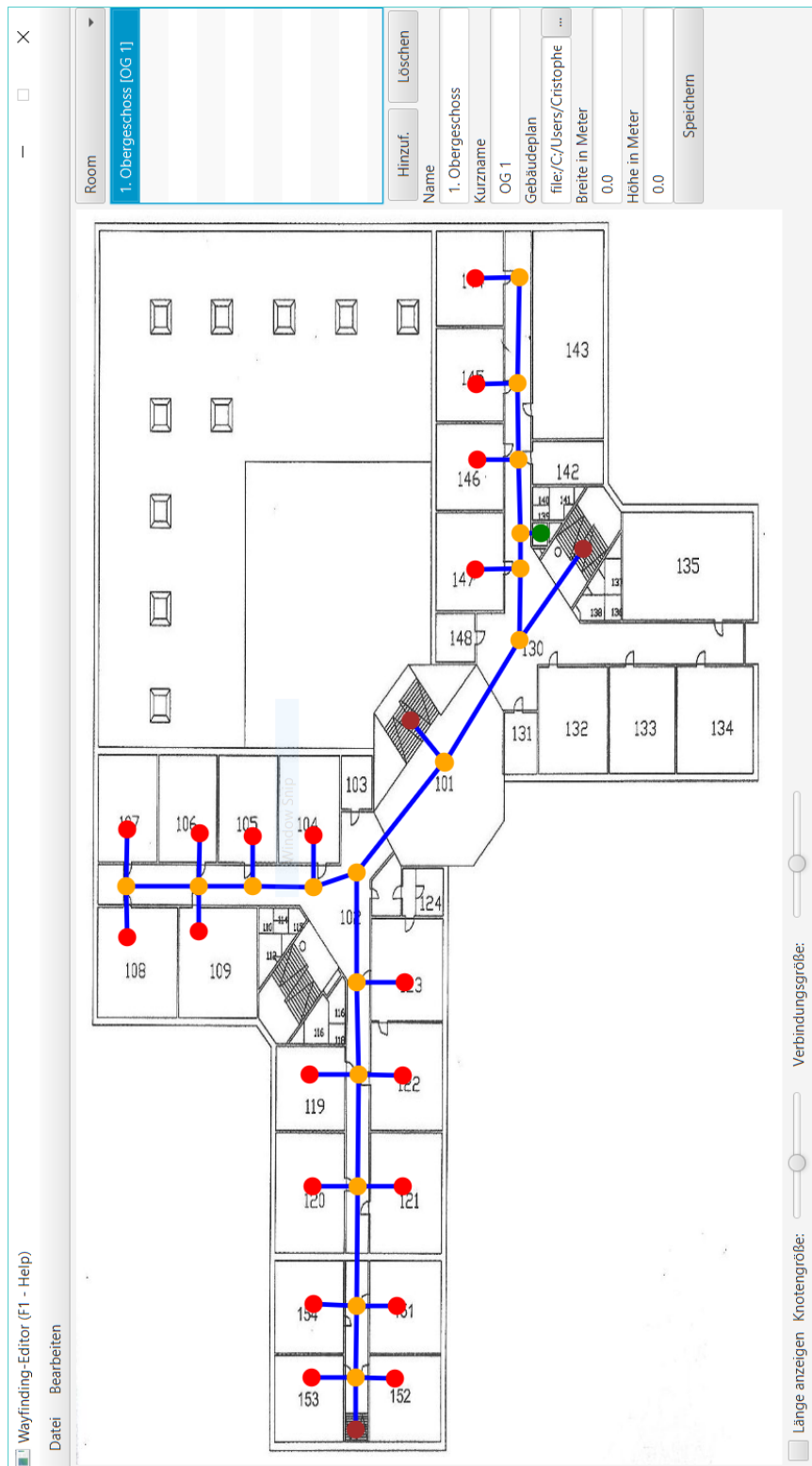


Abbildung 7: GUI
62

4.2.1 Canvas

In diesem Bereich werden Knoten (Räume, Corners, Lifte und Stiegen) definiert. Diese können auch verbunden werden, um zwischen ihnen einen begehbaren Weg anzugeben. Beim Anklicken eines Knotens oder einer Relation wird diese blau hinterlegt, um anzuzeigen, welcher Knoten bzw. welche Relation gerade bearbeitet wird. Durch das Ziehen eines Knotens mit der Maus kann die Position verändert werden. Falls 2 Knoten verbunden sind und einer davon wird verschoben, bewegt sich die Linie mit, um die Relation bzw. den Weg an die neue Position anzupassen.

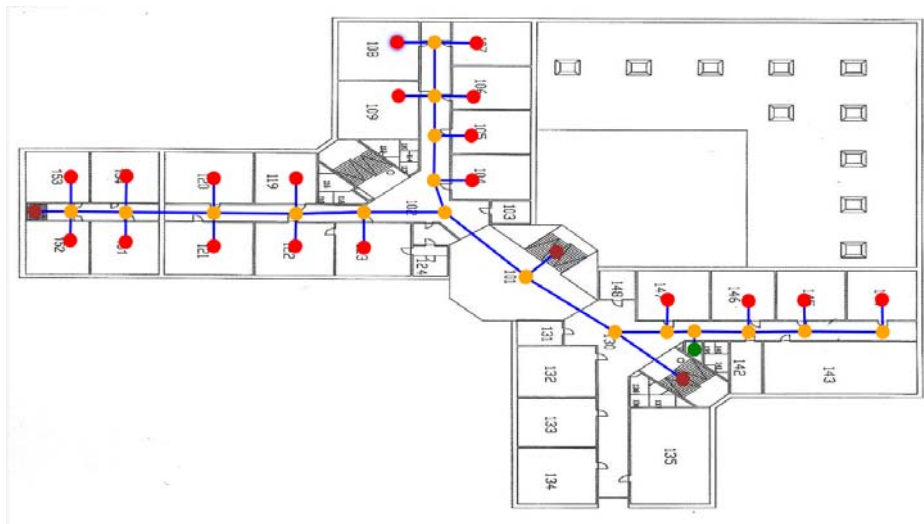


Abbildung 8: Canvas

Mithilfe verschiedener Maus und Tastenkombinationen, können die Elemente am Canvas erzeugt, gelöscht oder verbunden verbunden werden.

- **STRG + LINKS CLICK**

- Fügt einen neuen Knoten an der derzeitigen Mausposition hinzu.(Ob Raum, Corner, Stiege oder Lift hängt von der vorher definierten Auswahl ab)

-
- **SHIFT** (halten) + **LINKS CLICK** (auf 2 separate Knoten)
 - Verbindet 2 Knoten und erstellt zwischen ihnen eine neue Relationslinie. Nachdem die Relation erstellt wurde, wird sie automatisch ausgewählt, damit ihre Eigenschaften sofort geändert werden können.
 - **ALT** + **LINKS CLICK**
 - Löscht einen Knoten bzw. eine Relation. Wenn ein gelöschter Knoten mit anderen verbunden war, werden die verbindenden Relationen sofort mitgelöscht.



Abbildung 9: Hinzufügen und Verbinden 2er Knoten

4.2.2 Menubar

In der sogenannten **Menubar**, im oberen Bereich der GUI, gibt es zwei verschiedene Drop-Down-Menüs.

Unter dem Menüpunkt **Datei** kann ein Stockwerk aus einer Datei **importiert** bzw. in eine Datei **exportiert** werden. Dies hat den Vorteil, dass ein Stockwerk von einem Computer zum anderen übertragen werden kann, selbst wenn beide PCs eine andere Datenbank verwenden. Als Dateieindung wird das Format **.wfef (WayFindingEditorFloor)** akzeptiert. Als dritte Option steht dem Benutzer die Option **HTML Generieren** zur Verfügung, welches eine fertige HTML Datei generiert mit allen Stockwerken und Knoten, um diesen an einen, für einen Kunden zur Verfügung stehenden, Touchscreen zu schicken. Über diesen Touchscreen kann der Kunde einen, zuvor im Editor vordefinierten, Raum suchen, und anschließend auswählen. Nach seiner Auswahl wird ihm der genaue und effizienteste Weg dorthin gezeigt.

Bei Auswahl einer der oben genannten 3 Optionen wird ein File-Dialog Fenster geöffnet um eine Datei zum Öffnen bzw. Speichern auszuwählen. Falls beim Importieren eines Stockwerks, ein Stockwerk mit dem Namen bereits existiert, öffnet sich ein Fenster, wo man auswählen kann, ob man dieses Stockwerk überschreiben, den Importvorgang abbrechen, oder beide Versionen behalten will.

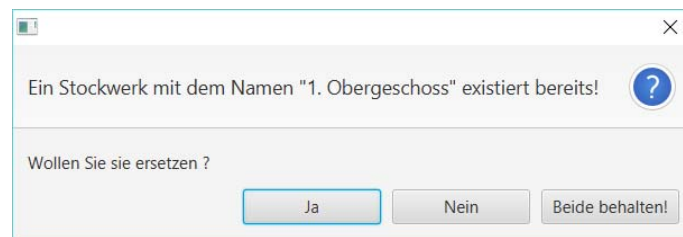


Abbildung 10: Stockwerk überschreiben

In **Tools** hat der User die Möglichkeit **Alle Längen automatisch berechnen** zu lassen, d.h. sofern im Stockwerk ein Maßstab angegeben wurde, werden alle Abstände zwischen den Knoten maßstabsgetreu berechnet und beim Verschieben dynamisch mitverändert.

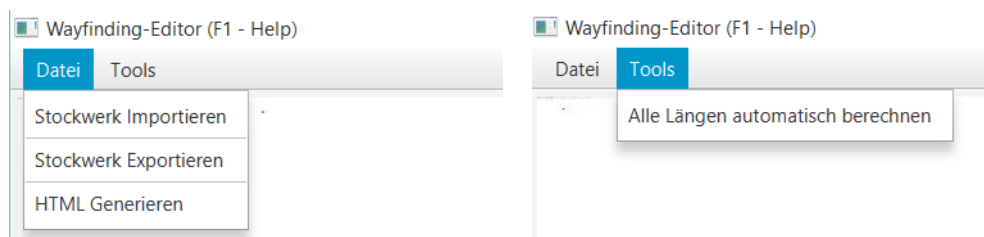


Abbildung 11: Menubar

4.2.3 Knoten - Choicebox

Über diese Choicebox kann ausgewählt werden, welcher Knoten zurzeit gezeichnet werden kann. Zur Verfügung stehen:

- Raum

-
- Corner
 - Stiege
 - Lift

Jeder Knoten hat eine eigene Farbe, um sie schnell auf der Karte identifizieren zu können. Alle Knoten können aber miteinander verbunden werden, allerdings wird davon abgeraten, 2 Knoten zu verbinden ohne einen Corner zwischen ihnen zu haben.

Um schnell zwischen den verschiedenen Knoten zu wechseln, können die Tasten 1 - 4 verwendet werden. (Zuordnung entspricht der Reihenfolge in der Liste (1 = Raum etc.)



Abbildung 12: Knoten - Choicebox

4.2.4 Stockwerkliste

In dieser Liste werden alle Stockwerke des Gebäudes mit ihrem Namen und ihrer Kurzschreibweise (z.B.: OG1 oder EG) angezeigt. Beim Anklicken eines Stockwerks wird der Hintergrund des Canvas durch die, des jeweilig festgelegten Gebäudeplans für dieses Stockwerk, ersetzt und alle assoziierten Knoten und Relationen gezeichnet. Mit den 2 darunter liegenden Buttons ist es möglich, eine Etage hinzuzufügen oder auch eine zu löschen. Beim Hinzufügen einer neuen Etage wird diese automatisch ausgewählt, um im Properties Bereich ihre Eigenschaften zu ändern. Wird ein Stockwerk gelöscht, so wird das erste in der Liste vorkommende Stockwerk ausgewählt.

Abbildung 13: Stockwerkliste

4.2.5 Editor Einstellungen

Hier können die Knoten, Relationen etc. optisch verändert werden. Die Längen der Relationen können durch anklicken einer Checkbox eingeblendet werden und mit den 2 Slidern ist es möglich die Größe der Kreise und der Relationen zu ändern. Die Änderung werden jeweils sofort sichtbar und es ist nicht nötig die Knoten und Relationen neu zu laden. Die Einstellungen werden zusätzlich in einer eigenen Datei gespeichert, damit die Größen der Figuren und die Anzeige der Länge bei einem Neustart nicht zurückgesetzt werden.

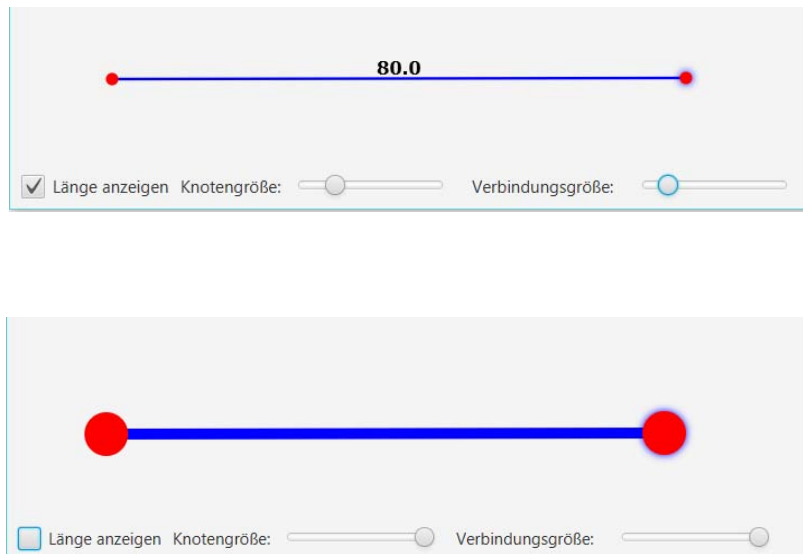


Abbildung 14: Editor Einstellungen

4.2.6 Properties

Je nachdem welches Objekt zurzeit ausgewählt ist (Knoten, Relation oder Stockwerk), wird hier dynamisch eine andere View angezeigt um dessen Eigenschaften zu ändern. Folgende Views können in diesem Bereich angezeigt werden.

Raum

Name	<input type="text"/>
Fachbereich	<input type="text"/> ▼
Station	<input type="text"/> ▼
<input type="button" value="Speichern"/>	

Abbildung 15: Raum - Properties

Einem Raum kann man einen Namen vergeben, welches beim Kunden - UI angezeigt wird, und nach dem er sich auch beim Auswählen eines bestimmten Raumes richten kann. Zusätzlich ist es auch möglich einem Raum einen Fachbereich und eine Station zuzuordnen, um bestimmte Räume zu gruppieren. So ist es auch möglich, Bereiche eines Gebäudes zu finden, wenn der genaue Raumname nicht bekannt ist.

Corner

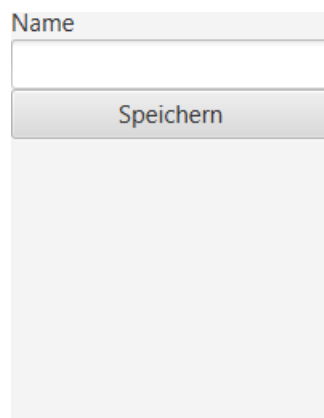
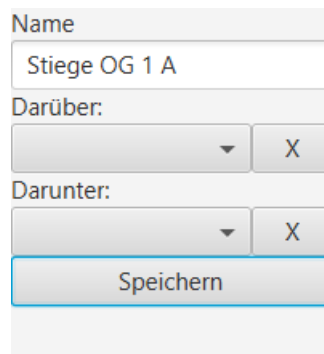


Abbildung 16: Corner - Properties

Ein Corner besitzt als Eigenschaft nur einen Namen. Dieser Name ist unwichtig für die spätere Navigation und hilft dem Benutzer des Editors nur während der Erstellung des Raumnetzes. Falls z.B.: die Richtung einer Relation nicht bidirektional sein kann, da eine Tür nur in eine Richtung aufgeht, so muss man erkennen können von welchem Corner zu welchem Corner sich die Relation bewegt. Zu diesem Zweck kann man einem Corner einen Namen geben. Die Hauptaufgabe eines Corners ist schließlich nur die Richtung eines Weges zu ändern, also benötigt er keine weiteren Eigenschaften.

Stiege

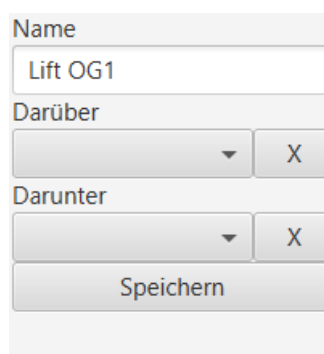


Name	
Stiege OG 1 A	
Darüber:	
<input type="text"/>	X
Darunter:	
<input type="text"/>	X
Speichern	

Abbildung 17: Stiege - Properties

Eine Stiege kann einen Namen besitzen, und zusätzlich noch eine Verbindung zu 2 weiteren Stiegenknoten. Diese 2 Knoten beschreiben die darüber und darunter liegenden Stockwerksverbindungen. Wählt man als Stiege A eine darunterliegende Stiege B, so wird automatisch für Stiege B, Stiege A als darüberliegend eingespeichert. Mit einem Klick auf das X neben diesen Knoten wird die Verbindung gelöscht. Stiegen im untersten bzw. obersten Geschoß haben keine darunter- bzw. darüberliegenden Stockwerke, somit kann das jeweilige Feld leer gelassen werden.

Lift



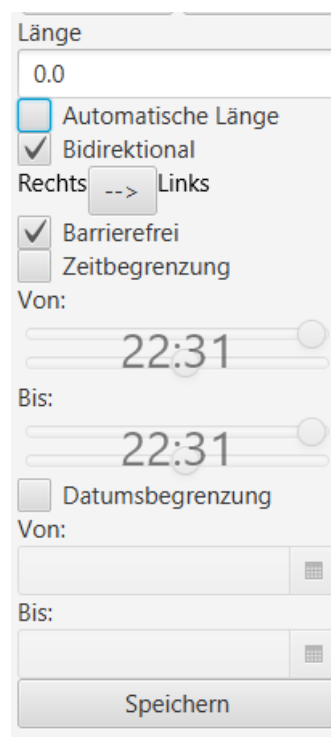
Name	
Lift OG1	
Darüber	
<input type="text"/>	X
Darunter	
<input type="text"/>	X
Speichern	

Abbildung 18: Lift - Properties

Ein Lift kann einen Namen besitzen, und zusätzlich noch eine Verbindung

zu 2 weiteren Liftknoten. Diese 2 Knoten beschreiben die darüber und darunter liegenden Stockwerksverbindungen. Wählt man als Lift A einen darunterliegenden Lift B, so wird automatisch für Lift B, Lift A als darüberliegend eingespeichert. Mit einem Klick auf das X neben diesen Knoten wird die Verbindung gelöscht. Lifte im untersten bzw. obersten Geschoß haben keine darunter- bzw. darüberliegenden Stockwerke, somit kann das jeweilige Feld leer gelassen werden. Lifte werden (im Gegensatz zu Stiegen) immer als barrierefrei gesehen. Sollte ein Lift aus irgendeinem Grund nicht barrierefrei sein, so wird empfohlen alle Relationen zu diesem Lift als nicht barrierefrei anzugeben.

Relation



The image shows a software interface for defining a 'Relation' between two nodes. The interface includes the following elements:

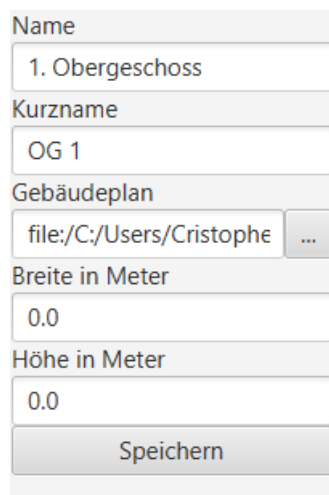
- Länge:** A text input field containing the value '0.0'.
- Automatische Länge:** An unchecked checkbox.
- Bidirektional:** A checked checkbox.
- Rechts --> Links:** A button with a double arrow pointing right.
- Barrierefrei:** A checked checkbox.
- Zeitbegrenzung:** An unchecked checkbox.
- Von:** A time range slider set to '22:31'.
- Bis:** A time range slider set to '22:31'.
- Datumsbegrenzung:** An unchecked checkbox.
- Von:** A date range input field with a calendar icon.
- Bis:** A date range input field with a calendar icon.
- Speichern:** A button at the bottom to save the settings.

Abbildung 19: Relation - Properties

Eine Relation zwischen 2 Knoten hat wesentlich mehr Eigenschaften, als die Knoten selber. Eine Länge gibt an wie weit entfernt die 2 verbunden Knoten von einander sind. Dies kann als Wegeinheit aber auch als Zeiteinheit interpretiert werden. Wichtig ist nur, dass für alle Relationen

eines Gebäudes diese Interpretation gleich bleibt. (Zeiteinheiten und Weeinheiten sollten nicht gemischt werden). Durch das Anhängen von **Automatische Länge** wird, sofern im Stockwerk ein Maßstab angegeben wurde, die Länge automatisch berechnet und beim Verschieben eines Knotens dynamisch neu berechnet. Ein Haken bei **Bidirektional** bedeutet, dass dieser Weg in beide Richtungen begangen werden kann. Sollte die Richtung eine Rolle spielen, so kann durch Anklicken des darunterliegenden Pfeils die Richtung geändert werden. Hierbei ist wichtig, dass die verbunden Knoten Namen besitzen, um die Richtung bestimmen zu können. Sollte ein Weg durch eine Stufe oder ähnliche Barriere blockiert sein, kann der Haken bei **Barrierefrei** weggelassen werden. Wenn sich ein Kunde bei der Raumsuche einen barrierefreien Weg wünscht, werden nur Wege angezeigt die barrierefrei sind. Mit den letzten 2 Checkboxes kann eine Zeit- bzw. eine Datumsbegrenzung aktiviert oder deaktiviert werden. Außerhalb der eingegebenen Zeiten ist das Navigieren über diese Relation dann nicht mehr möglich!

Stockwerk



Name	1. Obergeschoss
Kurzname	OG 1
Gebäudeplan	file:/C:/Users/Cristophe ...
Breite in Meter	0.0
Höhe in Meter	0.0
Speichern	

Abbildung 20: Stockwerk - Properties

Ein Stockwerk kann einen Namen besitzen, und zusätzlich noch einen Kurznamen, um ihn mit wenigen Buchstaben bzw. Zahlen zu beschreiben. Dieser kann z.B.: beim Navigieren verwendet werden, falls für den vollen Namen nicht genug Platz frei wäre. Durch das Hinzufügen eines

Gebäudeplans wird der Hintergrund des Canvas gesetzt auf den man nun Knoten und Relationen hinzufügen kann. In den letzten zwei Feldern kann die maßstabsgetreue Breite und Höhe des Gebäudeplans festgelegt werden. Diese dienen der Berechnung der Längen zwischen den eingeblendeten Knoten (falls eine automatische Längenberechnung ausgewählt wurde).

4.3 Ungewollte Knotenüberschreibung

4.3.1 Problem

Ein großes Problem, beim Entwickeln des Wayfinding - Editors, stellt ein ungewolltes Überschreiben von Knotenkoordinaten dar. Beim Verschieben eines Knotens (nach dem Verschieben wird ein Knoten in der Datenbank aktualisiert) kommt es öfters vor, dass die neuen Koordinaten, nicht nur beim verschobenen Knoten gespeichert werden, sondern in häufigen Fällen, die Koordinaten eines benachbarten Knotens überschreiben. Eine Folge dieser Überschreibung, ist eine Überlagerung von Knoten nach einem Neustart des Editors. Sichtbar wird diese Überlagerung durch das Zusammentreffen von mehreren Relationen zum scheinbar selben Knoten.

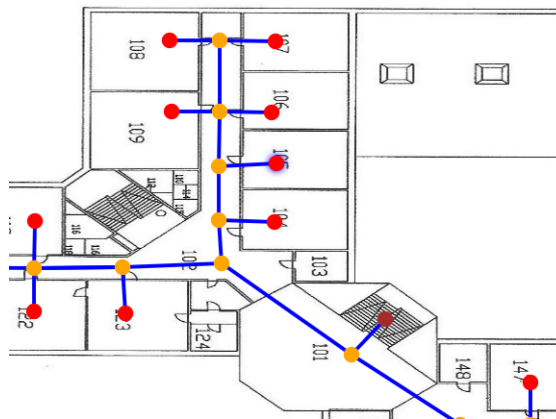


Abbildung 21: Vor dem Überschreiben

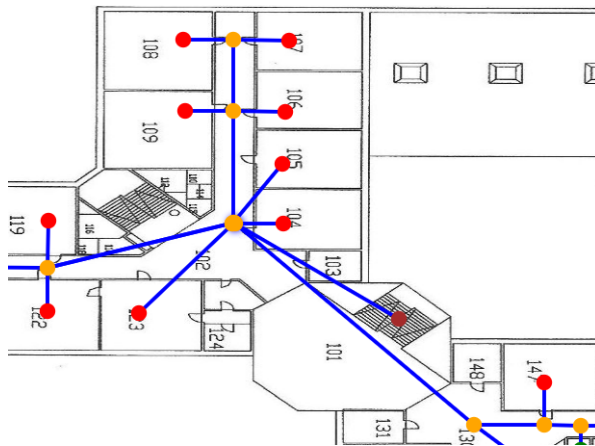


Abbildung 22: Nach dem Überschreiben

4.3.2 Ursache

Die Ursache dieses Problems, ist ein Bug in **Neo4j - OGM**. Eine mitspeicherung von benachbarten Knoten ist zwar gewollt, jedoch scheint es einen Bug zu geben, wodurch nicht die benachbarten Werte gespeichert werden, sondern die Werte des Wurzelknotens übernommen werden.

4.3.3 Behebung

Um dieses Problem zu beheben wird der Parameter **depth** der **save** Methode von **Neo4j - OGM** verwendet. Durch das Setzen dieses Parameters auf **0** wird der Persistierungs-Schicht mitgeteilt, dass keine benachbarten Knoten mitgespeichert werden sollten.

```
/*
    In diesem Thread werden Objekte in der Datenbank
    gespeichert und aktualisiert
*/
@Override
public void run()
{
    if((objectToUpdate instanceof Room ||
        objectToUpdate instanceof Corner)){

        /*
            Falls dieses Objekt ein Raum oder ein Corner ist,
            werden Nachbarn nicht mitgespeichert! (depth = 0)
        */
        DatabaseRepository.getInstance().getSession().save(objectToUpdate,0);

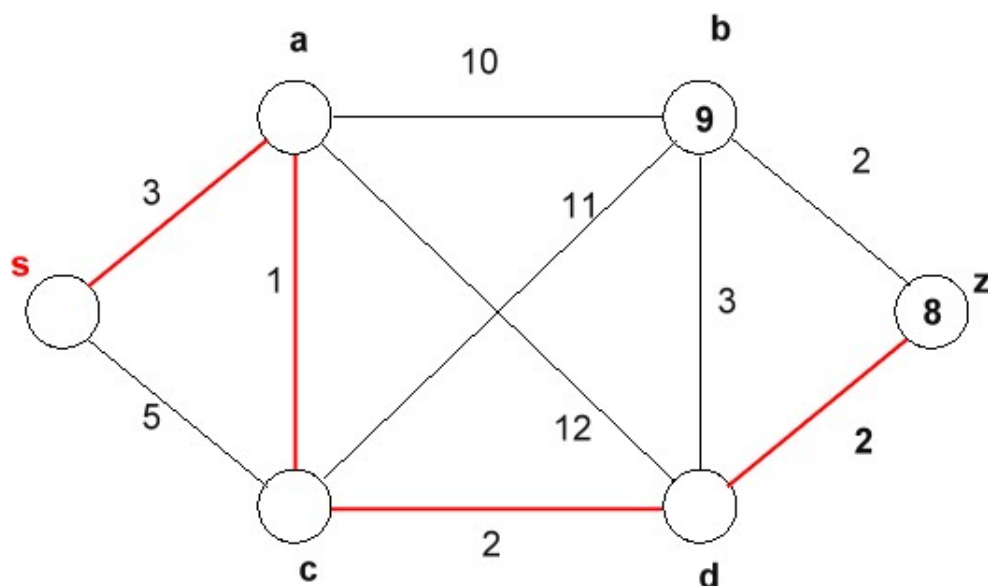
    }else {
        /*
            Objekte wie Lifte, Stiegen und Floors müssen, aufgrund
            ihrer gegenseitigen Abhängigkeit, stets ihre Nachbarn
            mitaktualisieren! (depth = 1)
        */
        DatabaseRepository.getInstance().getSession().save(objectToUpdate,1);
    }
}
```

4.4 Realisierung im Client

Um die Anforderung, den kürzesten Pfad zu finden, erfüllen zu können, musste ein Algorithmus benutzt werden, welcher die Berechnung dieses Pfades auch machen konnte. Die Wahl fiel schnell auf den Dijkstras-Algorithmus.

4.4.1 Dijkstras-Algorithmus

Dieser Algorithmus errechnet sich in einem kantengewichteten Graphen den Weg von einem Startknoten zu einem oder mehreren Endknoten. Es ist zu beachten, dass dieser Graph keine Negativkanten enthält. vgl.[25]



[2]

Abbildung 23: Dijkstras-Algorithmus

Der Dijkstras-Algorithmus ist im Gegensatz zum Bellman-Ford-beziehungsweise Floyds-Warshall-Algorithmus extrem schnell und hat eine Zeitkomplexität von lediglich $O(n * \log(n) + m)$ wobei n die Anzahl der Knoten und m die Anzahl der Kanten sind. vgl.[25] Im Gegensatz dazu hat der Bellman-Ford-Algorithmus eine Komplexität von $O(n * m)$ vgl.[19] und der Floyd-Warshall-Algorithmus eine Komplexität von $O(n^3)$ hat. vgl.[18]

Der A*-Algorithmus hat eine Abschätzfunktion und kann durch diese in manchen Fällen schneller sein als der Dijkstras-Algorithmus. Dieser ist aufgrund seines enormen Speicherplatzverbrauchs undenkbar für eine Lösung die an einen Raspberry-Pi angepasst ist. vgl.[22]

Die in unserem Programm verwendete Lösung für den Dijkstras-Algorithmus stammt von James Jackson. Diese Lösung fanden wir im Internet und das JavaScript-File heißt Dijkstras.js .

4.4.2 Dijkstras.js

Die Dijkstras.js beruht auf einem Graphen, der als Array in JavaScript realisiert wurde.

```
[  
  ['Knoten1', [[ 'Knoten2', 20], ['Knoten3', 20]]],  
  ['Knoten4', [[ 'Knoten2', 10], ['Knoten5', 40]]]  
]
```

So sieht der Aufbau des verschachtelten Arrays aus. Das erste Array beinhaltet alle anderen in ihm enthaltenen Arrays und ist sozusagen der ganze Graph. Die inneren Arrays beinhalten jeweils einen Knoten und ein Array das aus den Kantenlängen und den Knoten zu den die Kanten laufen besteht. Die Knoten müssen alle als Text und die Kantengewichtungen als Zahlen übergeben werden. Dieses Array wird der Funktion setGraph übergeben. Diese Funktion prüft die Richtigkeit des Arrays und speichert ihn, wenn er den Anforderungen der Dijkstras.js entspricht, als Graph ab. Die Ermittlung des kürzesten Pfades erfolgt über die Funktion getPath. Dieser Funktion werden der Start- und der Endknoten als Parameter übergeben. Die verwendeten Kanten und Knoten stammen aus einer JSON-Variable in der JavaScript-Datei. Das Array für den Wayfinding-Client sieht wie folgt aus:

```
function createMap(){  
  var k=0;  
  for(var i =0;i<knots.length;i++){  
    {  
      map[i]= new Array();  
      map[i][0]= knots[i]["nodeId"].toString();  
      map[i][1]= new Array();  
      for(var j =0;j<relations.length;j++)
```

```

    {
        if(knots[i]["nodeId"] ===
            relations[j]["startNode"]["nodeId"])
        {
            map[i][1][k]=new Array();
            map[i][1][k][0]=relations[j]["endNode"]["nodeId"].toString();
            map[i][1][k][1]=relations[j]["length"];
            k++;
        }

        else if(knots[i]["nodeId"] ===
            relations[j]["endNode"]["nodeId"])
        {
            map[i][1][k]= new Array();
            map[i][1][k][0]=relations[j]["startNode"]["nodeId"].toString();
            map[i][1][k][1]=relations[j]["length"];
            k++;
        }
    }
}

```

Im Array knots sind alle Knoten aus der JSON-Variable gespeichert. Die Kantengewichtung zu bestimmten Knoten von einem Startknoten weg ist in der JSON-Variable relations enthalten. In den JSON-Variablen sind nicht nur die Knoten und deren Kantenlängen vorhanden, sondern auch die Koordinaten zu jedem einzelnen Knoten, um diesen dann auf der Website grafisch darstellen zu können.

4.4.3 Aufbau der JSON-Datei

```

floors = [
{
    "floorId": 123,
    "name": "Obergeschoss",
    "shortName": "OG",
    "picture": (Bild als base64 kodiert),
    "nodes":[
        {
            "nodeId": 1,
            "isA": "Lift"

```

```

        "name": "Lift OG",
        "xCoordinate": 149.0,
        "yCoordinate": 200.0
    },
    {
        ...
    }
]
},
{...}]
relations = [
    {
        "timeFrom": "",
        "timeRestricted": false,
        "bidirectional": true,
        "timeTo": "",
        "startNode": {
            "isA": "Lift",
            "nodeId": 992
        },
        "length": 3.75,
        "dateTo": "",
        "dateRestricted": false,
        "relationId": 1605,
        "dateFrom": "",
        "disabledFriendly": true,
        "endNode": {
            "isA": "Corner",
            "nodeId": 886
        }
    },
    {
        ...
    }
]

```

Die Variable `floors` enthält Knoten, die auf die dazugehörigen Stockwerke aufgeteilt sind. In dieser JSON-Datei ist das Hintergrundbild sowie Zusatzinformationen zum Stockwerk wie der Name ebenfalls enthalten. Die Variable `relations` enthält die Verbindungen zwischen den einzelnen Knoten. Es enthält neben der Länge der Verbindungen auch Zusatzinformationen wie Datumseinschränkung, Zeiteinschränkung oder Barrierefreiheit.

4.4.4 Koordinaten anpassen

Die über die JSON-Datei übergebenen Koordinaten aus dem Editor, müssen im Client angepasst werden, da die Zeichenfläche im Editor andere Maße für die Länge und Breite hat als die im Client. Die Punkte würden ohne Anpassung nicht an der richtigen Stelle im Canvas des Clients stehen und würden den Benutzer nicht zum gewünschten Zielort führen. Durch einen Faktor, mit dem man die Koordinaten multipliziert, kann man sicher stellen, dass jeder Knoten auch am richtigen Platz im Canvas ist. Den Faktor für die Y-Koordinate erhält man, indem man die Höhe des Canvas im Client durch die Höhe des Canvas im Editor dividiert. Weiters muss man ebenfalls einen Faktor für die X-Koordinate haben. Diesen erhält man, indem die Breite der Zeichenfläche im Client durch die Breite der Zeichenfläche im Editor dividiert wird.

4.4.5 Zeichnen der Knoten und Kanten

Das Zeichnen der Kanten und Knoten im Canvas wird durch eine Bibliothek von jQuery deutlich erleichtert, da viele Bugs der normalen JavaScript-Canvas-Funktionen behoben wurden. Um den Pfad zeichnen zu können, muss man das Array der Knoten durchgehen und den Knoten nur dann zeichnen, wenn dieser dem nächsten Knoten im Pfad entspricht und auf dem dargestellten Stockwerk liegt. Falls der Knoten nicht im dargestellten Stockwerk liegt, wird der Index des Arrays in dem der Pfad gespeichert ist vorgemerkt. Beim nächsten Durchlauf wird dann ab diesem Index gestartet. Dies ist wichtig, um die Knoten auf das richtige Stockwerk zu zeichnen. Die Relationen werden als Linien dargestellt und werden durch die Koordinaten des aktuellen und des nächsten Knotens gezeichnet. Die Knoten zwischen dem Start und dem Ende werden nicht hervorgehoben. Das Ende und der Start in jedem Stockwerk werden besonders hervorgehoben, um dem Benutzer die Bedienung und Lesbarkeit des Leitsystems zu erleichtern. Der Startknoten ist in der Farbe grün gezeichnet und der Endknoten in der Farbe rot. Neben dem farblichen Unterschied steht über dem Startknoten auch textuell Start und neben dem Endknoten Ende. Das Zeichnen ist wie folgt im Client verwirklicht:

```
for (var i = pathIndex[pathIndex.length - 1]; i < sPath.length;
    i++) {
    if ((getFloorIndexByNodeId(sPath[i + 1]) !== floorIndex))
        {
```

```
        if(cnt === 0)
        {
            drawKnot(sPath[i], 'Ende', "red");
        }
        else {
            drawKnot(sPath[pathIndex[pathIndex.length - 1]],
                'Start', "green");
            drawKnot(sPath[i], 'Ende', "red");
        }

        pathIndex.push(i + 1);
        break;
    }
    cnt++;
    $("#canvas").drawLine({
        strokeStyle: '#000',
        strokeWidth: 5,
        x1: getKnotById(sPath[i])['xCoordinate']*faktorW,
        y1: getKnotById(sPath[i])['yCoordinate']*faktorH,
        x2: getKnotById(sPath[i + 1])['xCoordinate']*faktorW,
        y2: getKnotById(sPath[i + 1])['yCoordinate']*faktorH
    });
    $("#canvas").drawArc({
        fillStyle: '#000',
        x: getKnotById(sPath[i + 1])['xCoordinate']*faktorW,
        y: getKnotById(sPath[i + 1])['yCoordinate']*faktorH,
        radius: 2
    });
}
```

4.4.6 Darstellung

Das Leitsystem wird im Client so realisiert, dass es in einer Webseite dargestellt wird. Die Webseite ist eine sogenannte Single-Page-Application. Das heißt, dass die Webseite nur aus einer HTML-Datei besteht, die in Teile (sections) aufgeteilt ist. Jede Section ist eine sogenannte Hero-Section. Das heißt, dass sich eine Hero-Section über die volle Breite und Höhe des Browserfensters streckt. Das händische hinunterscrollen auf der Webseite ist ausser Kraft gesetzt. Um auf einen anderen Teil der Webseite zu kommen, muss man auf einen Button (Knopf) drücken oder Elemente aus einer Liste

antippen. Der Client besteht aus vier Hero-Sections. Da das Leitsystem die letzte der vier Hero-Sections ist, muss man also in allen vorherigen Sections ein Element aus einer Liste auswählen. Die Webseite könnte aus nur zwei Hero-Sections bestehen. Dies würde aber die Benutzerfreundlichkeit einschränken, da eine große Anzahl an Elementen in der Liste enthalten wäre. Aus diesem Grund wurden die Listen so aufgeteilt, dass man in der dritten Section nur eine eingeschränkte Anzahl an Zimmern hat. Diese Anzahl ist durch die vorherige Auswahl eingeschränkt worden. Somit bewahrt der Benutzer den Überblick über die Webseite. Neben dem gezeichneten Pfad gibt es auch eine Wegbeschreibung in textueller Form. Diese Wegbeschreibung findet man neben der Zeichenfläche. Die Buttons unter der Zeichenfläche dienen zum Schalten auf das nächsthöhere oder nächsttiefere Stockwerk. Jede Section hat ebenfalls einen „Zurück zur Startseite“-Button. In der ersten Section kann man neben der Station auch wählen ob man einen barrierefreien Weg benötigt oder nicht. Dieser Weg wird dann so gezeichnet, in dem beispielsweise Stiegen oder andere Hindernisse, die für Personen mit eingeschränkter Mobilität nicht zumutbar sind, ignoriert. In der zweiten Section kann man die Unterstation zur gewählten Station auswählen. Die Überschrift über der Liste ist in der zweiten Section in der Farbe eingefärbt, die zur vorher ausgewählten Station gehört.

4.5 Überblick-Client

Wayfinding-Client ist ein Programm, welches dem Benutzer ermöglicht sich den kürzesten Pfad zu seinem gewünschten Zielraum anzeigen zu lassen. Es ermöglicht den Benutzer auch einen barrierefreien Pfad auszuwählen, in dem beispielsweise statt den Stiegen Lifte genommen werden. Der Wayfinding-Client ist die generierte HTML-Datei aus dem Editor.

4.6 Probleme bei der Umsetzung des Clients

In diesem Unterabschnitt wurde versucht, die Probleme, die während der Entwicklung des JavaScript-Clients entstanden sind, näher zu erläutern. Ebenfalls wurden die entsprechenden Lösungen zu den Problemen, die im Client realisiert sind, dargestellt.

4.6.1 Erstellung des Arrays

Die Erstellung des Arrays für die Dijkstras.js-Datei barg einige Probleme mit sich. Das Array wurde benötigt, um die im JSON-Format bekommenen Daten aus der JAVA-FX Applikation in ein Format zu verwandeln, mit dem die Dijkstras.js einen Graphen anlegen kann. Das Array wurde einfach an die Methode setGraph der Dijkstras.js übergeben und der Graph wurde somit angelegt. Ein großes Problem beim Erstellen des Arrays war, dass das Array vierfach verschachtelt sein musste. Eine Verschachtelungsstufe höher oder niedriger war nicht möglich, da die Methode setGraph ein vierfach verschachteltes Array entgegennahm und bei nicht Erfüllen dieser Bedingung wird kein Graph gesetzt. Weiters war eines der Probleme, dass die Arrays, die mit `var Array = new Array();` angelegt werden, nur numerische Werte als Index annehmen und keine Buchstabenfolge. Aus diesem Grund wird das Array sehr unübersichtlich und kann zu Fehlern führen, die man sehr schwer finden kann. Neben den beiden bereits erwähnten Komplikationen trat ein weiteres Problem ein. Da der Graph für Personen mit eingeschränkter Mobilität einige Knoten, die nicht barrierefrei sind, nicht beinhalten sollte, musste eine Lösung gefunden werden, um Teile des Arrays herauszunehmen und in ein zweites Array so wieder einzubringen, dass dieses den Richtlinien, die die Funktion setGraph hatte entsprach. Da der JavaScript-Client einfach aufgebaut sein sollte, wurde entschieden das Array nicht durch eine Deep-Copy umzukopieren, sondern vielmehr die Daten ein zweites Mal aus dem vorhandenen JSON-Format zu lesen und nur die Teile in das Array einzufügen, die einen barrierefreien Weg ermöglichten.

4.6.2 Zeichnen des Pfades

Das Teilen des zu zeichnenden Pfades auf mehrere Stockwerke stellte ein Problem dar, dass durch einige Kniffe relativ simpel gelöst wurde. Der Pfad, der durch die Funktion getPath(), die aus dem Dijkstras.js stammt, wurde so lange durchgegangen, bis der nächste zu zeichnende Knoten in einem anderen Stockwerk war. Im JSON, das aus dem Editor stammt, ist festgelegt, welche Knoten zu welchem Stockwerk gehören. Sobald der nächste Knoten in einem anderen Stockwerk, wurde diese Position in einer Variable vorgemerkt und die Funktion für das Zeichnen des Weges wurde abgebrochen. Das Zeichnen und die Anzeige des Pfades, der in einem anderen Stockwerk ist, wird durch das Drücken des Weiterbuttons oder des

Zurückbuttons gestartet. Die Verbindungsknoten zwischen den Stockwerken, wie Stiegen und Lifte wurden als spezielle Knoten gelöst und deren Verbindung und die Lösung dieses Problems wurde im Editor vorgenommen.

4.6.3 Weiter & Zurück-Button

Der Zurückbutton stellte insofern ein Problem dar, da der Zähler für das nächste Stockwerk nun zu einem Array umgewandelt werden musste, in dem für jedes Stockwerk der Anfangsindex des Pfades angegeben sein muss, um den Weg zum vorherigen Stockwerk auch wirklich zeichnen zu können, denn ansonsten wäre der Weg nicht nachvollziehbar. Das einzige, dass möglich wäre ohne ein Array wäre, das man am Anfangsknoten des Pfades anfängt. Dies wäre aber nicht sehr sinnvoll. Aus diesem Grund ist der Einsatz eines Arrays hier sinnvoll und auch notwendig, wenn man den Benutzer des Wegfindungssystems nicht verwirren möchte und ihm unnötig viele Clicks ersparen möchte. Der Weiterbutton bereitete Schwierigkeiten, da es möglich war über das Ende des Pfades zu zeichnen beziehungsweise einen weiteren Stock anzuzeigen. Dies wurde gelöst in dem eine Abfrage eingebaut wurde, die überprüft ob der nächste zu zeichnende Floor überhaupt einen gültigen Index besitzt oder ob der Index nicht definiert ist. Falls der Index nicht definiert ist wird das Stockwerk nicht gezeichnet ansonsten schon.

4.7 GUI-Client

Der Client lässt sich in 4 Teile gliedern:

- Auswahl der Station
- Auswahl der Unterstation
- Auswahl des Raumes
- Canvas

4.7.1 Auswahl der Station

Auf diesem Teil der Webseite lässt sich über einen Klick auf den Touchscreen die gewünschte Station auswählen. Eine Station besteht aus einer oder mehreren Unterstationen.



Abbildung 24: Auswahl der Station

4.7.2 Auswahl der Unterstation

Dieser Teil der Webseite lässt den Benutzer mit einem Klick auf den Touchscreen die gewünschte Unterstation auswählen. Eine Unterstation besteht aus einem oder mehreren Knoten.



Abbildung 25: Auswahl der Unterstation

4.8 Auswahl des Raumes

Hier wählt der User schließlich seinen Zielraum mittels Klick auf den Touchscreen aus. Nach diesem Teil der Webseite wird der Pfad berechnet und angezeigt.



Abbildung 26: Auswahl des Raumes

4.9 Canvas

Dies ist der Hauptteil der Webseite. Hier erfolgt die Veranschaulichung des Pfades. Die Berechnung des Pfades erfolgt durch den

Dijkstras-Algorithmus. Der Algorithmus berechnet sich den kürzesten Pfad durch die Gewichtung beziehungsweise die Länge der Relationen zwischen den Knoten. Der erste Knoten in einem Stockwerk wird grün eingefärbt und mit dem Text **Start** belegt. Der letzte Knoten in einem Stockwerk wird rot eingefärbt und mit dem Text **Ende** belegt. Falls in

einem Stockwerk nur ein Knoten sein sollte, so wird der Knoten rot eingefärbt und mit dem Text Ende belegt.



Abbildung 27: gezeichneter Pfad

Um das nächste Stockwerk anzuzeigen muss auf den Button **Weiter** gedrückt werden. Falls man nochmal ein Stockwerk zurück will, klickt man auf den Button **Zurück**.



Abbildung 28: Weiter- & Zurück-Button

Über den Button **Neue Suche** kommt man wieder zurück zur Auswahl der Station.

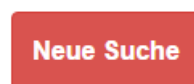


Abbildung 29: Neue Suche - Button

5 Anhang

Literatur

- [1] Calc. https://de.wikipedia.org/wiki/Apache_OpenOffice#Calc.
- [2] Der dijkstras algorithmus.
<http://www.zahlendoktor.de/kuerzesterWeg.html>.
- [3] Flash video. https://de.wikipedia.org/wiki/Flash_Video.
- [4] Graphics interchange format.
https://de.wikipedia.org/wiki/Graphics_Interchange_Format.
- [5] Impress. https://de.wikipedia.org/wiki/Apache_OpenOffice#Impress.
- [6] Jpeg. <https://de.wikipedia.org/wiki/JPEG>.
- [7] Microsoft excel. https://de.wikipedia.org/wiki/Microsoft_Excel.
- [8] Microsoft powerpoint.
https://de.wikipedia.org/wiki/Microsoft_PowerPoint.
- [9] Microsoft word. https://de.wikipedia.org/wiki/Microsoft_Word.
- [10] Moving pictur experts group.
https://de.wikipedia.org/wiki/Moving_Picture_Experts_Group.
- [11] Mpeg-1. <https://de.wikipedia.org/wiki/MPEG-1>.
- [12] Mpeg-4. <https://de.wikipedia.org/wiki/MPEG-4>.
- [13] Portable document format.
https://de.wikipedia.org/wiki/Portable_Document_Format.
- [14] Portable network graphics.
https://de.wikipedia.org/wiki/Portable_Network_Graphics.
- [15] Vorteil von cloud loesungen. <http://www.pcwelt.de/ratgeber/Vorteile-von-Cloud-Loesungen-1486121.html>.
- [16] Windows bitmap. https://de.wikipedia.org/wiki/Windows_Bitmap.
- [17] Writer. https://de.wikipedia.org/wiki/Apache_OpenOffice#Writer.

-
- [18] Algorithmus von floyd und warshall. https://de.wikipedia.org/wiki/Algorithmus_von_Floyd_und_Warshall, August 2015.
 - [19] Bellman-ford-algorithmus. <https://de.wikipedia.org/wiki/Bellman-Ford-Algorithmus>, November 2015.
 - [20] Hypertext markup language. https://de.wikipedia.org/wiki/Hypertext_Markup_Language, 2015.
 - [21] Multi-monitor. <https://de.wikipedia.org/wiki/Multi-Monitor>, 2015.
 - [22] A*-algorithmus. https://de.wikipedia.org/wiki/A*-Algorithmus, Januar 2016.
 - [23] Bootstrap(framework). [https://de.wikipedia.org/wiki/Bootstrap_\(Framework\)](https://de.wikipedia.org/wiki/Bootstrap_(Framework)), 2016.
 - [24] Cascading style sheets. https://de.wikipedia.org/wiki/Cascading_Style_Sheets, 2016.
 - [25] Dijkstras-algorithmus. <https://de.wikipedia.org/wiki/Dijkstra-Algorithmus>, Februar 2016.
 - [26] Javascript. <https://de.wikipedia.org/wiki/JavaScript>, 2016.
 - [27] Windows media video. https://de.wikipedia.org/wiki/Windows_Media_Video, 2016.
 - [28] A. Bien. Igniting javafx applications. <http://afterburner.adam-bien.com/>, 2014.
 - [29] A. Bien. Structuring complex javafx 8 applications for productivity. <http://www.oracle.com/technetwork/articles/java/javafx-productivity-2345000.html>, October 2014.
 - [30] Caleb Evans. Learn all about jcanvas - introduction. <http://projects.calebevans.me/jcanvas/docs/introduction/>.
 - [31] Caleb Evans. Learn all about jcanvas - syntax. <http://projects.calebevans.me/jcanvas/docs/syntax/>.

-
- [32] P. Ghadir. Neo4j-eine graph-basierte transaktionale datenbank.
<https://www.innoq.com/en/articles/2012/09/neo4j-rockt/>, September 2012.
 - [33] Dr. Jörg Heidjann. Stromkosten berechnen.
<https://www.stromauskunft.de/strompreise/stromkosten-berechnen/>.
 - [34] The jQuery Foundation. jquery api. <https://api.jquery.com/>, 2016.
 - [35] o.V. The neo4j manual v2.3.2 33.3. property values.
<http://neo4j.com/docs/stable/property-values-detailed.html>, o.J.
 - [36] Wilson J. Singh I., Leitch J. Gson user guide.
<https://sites.google.com/site/gson/gson-user-guide>, o.J.

Abbildungsverzeichnis

1	Use Case Editor	11
2	Use Case Client	12
3	Systemarchitektur	35
4	ECB - Pattern (Entity - Control - Boundary)	46
5	Neo4J - Darstellung	53
6	Neo4J - Überblick	54
7	GUI	62
8	Canvas	63
9	Hinzufügen und Verbinden 2er Knoten	64
10	Stockwerk überschreiben	65
11	Menubar	65
12	Knoten - Choicebox	66
13	Stockwerkliste	67

14	Editor Einstellungen	68
15	Raum - Properties	68
16	Corner - Properties	69
17	Stiege - Properties	70
18	Lift - Properties	70
19	Relation - Properties	71
20	Stockwerk - Properties	72
21	Vor dem Überschreiben	73
22	Nach dem Überschreiben	74
23	Dijkstras-Algorithmus	76
24	Auswahl der Station	86
25	Auswahl der Unterstation	86
26	Auswahl des Raumes	87
27	gezeichneter Pfad	88
28	Weiter- & Zurück-Button	88
29	Neue Suche - Button	88

Tabellenverzeichnis

1	Server	20
2	Player	23
3	Externe Quellen	24
4	Usability	25
5	Multi-Monitor	26

6	Medienformate	33
7	Touchfunktion	34
9	Neo4j - Property Values	55
10	Neo4j - OGM Direction	57
11	Neo4j - OGM Befehle	60

Protokolle

Diplomarbeitsprotokoll

Emrah Gudic 11.09.2015

Anwesende: Hans-Christian Hammer, Matthias Derntl, Thomas Stütz,

Cristopher Hornsey, Reuben Meixner, Emrah Gudic

Teamleiter: Cristopher Hornsey

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Allgemeines zur Diplomarbeit:

Es ist festgestellt worden, dass der Tomcat7-Server vom Herrn Derntl weiterverwendet werden soll, da dieser einwandfrei läuft und ein umschreiben des Programmcodes zu umständlich ist, da der Server ein großer Monolit ist. Der Digital Signage Player vom Herrn Derntl wurde ebenfalls vorgestellt. Der bereits vorhandene Player wurde in Android programmiert und dieser läuft auf dem Android basierten Einplatinencomputer Odroid. Der Prototyp vom Herrn Derntl und der von unserem Team wurden auf einem Touchscreen-Fernseher getestet und die vorhandenen Funktionen, wie zum Beispiel das Umschalten auf ein anderes Video ausprobiert. Ein Leitsystem, dass von einem Kollegen vom Herrn Derntl programmiert wurde, könnte möglicherweise zum Teil verwendet werden. Ein Problem bei dem Leitsystem ist die statische Eingabe der Wege zu den verschiedenen Räumen. Herr Derntl schickte eine PDF über den Aufbau und die Daten des Servers an den Betreuungslehrer. Die Verbindung zu dem Tomcat7-Server wurde, nach leichten Schwierigkeiten mit dem installieren von Tomcat7 auf dem Linuxrechner, einwandfrei hergestellt.

Fragen an Herrn Derntl:

- Alle Funktionalitäten für das Digital Signage sind verfügbar.

-
- Die Identifikation über IP-Adressen könnte ein Problem darstellen. Die Lösung dieses Problems könnte sein, dass der zu erstellende Client das Verbindungsmodul zum Server modular gestalten muss, damit später dieser ausgewechselt werden kann z.B. Websockets.
 - Bei der Veränderung der Playlist schreibt der Server die Playlist im XML-Format (Port 8080) neu auf den Socket. Dateien werden nicht automatisch mitgeschickt sondern nur ihre Beschreibung. Falls eine Datei am Client fehlt, wird eine neue Socketverbindung am Port 80 aufgebaut und die Datei heruntergeladen.
 - Ein Testclient soll geschrieben werden, der nur anzeigt, was er vom Server bekommt. Dies soll die Basis für den richtigen Client sein. Der Testclient benutzt lediglich die Funktionen des Verteilervers und prüft ob die Playlist richtig ist und ob der Download funktioniert hat.
 - Im XML steht die RegionID für die Region am Bildschirm.
 - Man kann die Layouts im Server ändern
 - Für .pps gibt es einen Workaround, in dem man aus der .pps-Datei eine JPEG oder MPEG-Serie macht oder eine Google-Drive-Konvertierung vornimmt.

Technologien:

Herr Derntl ist positiv gegenüber HTML,JS und AngularJS eingestellt. Diese Technologien wurden vorgeschlagen, da diese Technologien eine Zukunft haben und sehr anpassbar sind.

Annahmen:

Störungsmeldungen werden nicht berücksichtigt. Laut Herrn Derntl ist die Funktion dieser Störungsmeldungen fragwürdig.

Verbindungsprozess zum Server:

- Der Client schickt eine Initialisierung
- Der Server schickt die Playlist
- Der Client aktualisiert die Verbindung
- Der Client schickt seine derzeitige Playlist (alle 120 Sekunden)

Folgende Ziele:

- Testclient für die Serverkommunikation: Karma?
 - Digital Signage
- Wie sieht die GUI aus?
 - Leitsystem
 - Digital Signage
- Testclient Protokoll testen
- GUI - Browser am PPI testen
- Kleinigkeiten wie zum Beispiel Leisten verschwinden lassen
- Datenmodell für die Hausnavigation

Marktanalyse:

- Produktbeschreibung (nicht länger als eine Seite)
- Kriterienkatalog
- Screenshots der Produkte
- Grafiken
- Unique Selling Points herausuchen

-
- Eigene Unique Selling Points aufstellen
 - Usermeinungen einholen
 - Plattformen auf denen die Produkte laufen herausfinden

TODO:

- Marktanalyse soll bis 21.09.2015 von Herrn Hornsey, Meixner und Gudic fertig gestellt werden.
- Matthias Dentl soll Überlegungen anstellen, ob eine Zusammenarbeit mit dem Diplomarbeitsteam sinnvoll ist.

Emrah Gudic 03.11.2015

Diplomarbeitsprotokoll

Anwesende: Hans-Christian Hammer, Cristopher Hornsey, Reuben Meixner,

Emrah Gudic

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Teamleiter: Cristopher Hornsey

Aufbau des Inhaltsverzeichnisses:

- Abstract
- Einleitung
- Theoretische Grundlagen (alle Unterpunkte sind Hauptkapitel)
 - Kriterienkatalog
 - Marktüberblick
 - Produkte im Überblick

-
- Näherer Eingang auf das MMDS
 - Fallstudie des MMDS
 - Umsetzungsprozess
 - Erfahrungen aus dem Umsetzungsprozess
 - mögliche Optimierungsschritte (diese Schritte sind außerhalb des Rahmens der Diplomarbeit noch möglich)
- Abschließende Bemerkungen
 - Anhang

Fehlendes im Kriterienkatalog:

- Die einzelnen Kriterien müssen unter der Überschrift kurz aufgelistet werden.
- Unter der Überschrift muss ebenfalls angegeben werden, warum man den Kriterienkatalog erstellt.
- Es soll eine Bewertungsskala unter den Kriterien angegeben werden.
- Der Player soll Unterpunkte beinhalten.
- Der Server soll ebenfalls Unterpunkte beinhalten.

Emrah Gudic 17.11.2015

Diplomarbeitsprotokoll

Anwesende: Hans-Christian Hammer, Cristopher Hornsey, Reuben Meixner,

Emrah Gudic

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Teamleiter: Cristopher Hornsey

Änderungen für den Kriterienkatalog:

- In der Begründung für die Kriterien sollen keine Produkte sein.
- Es sollen einfachere Sätze verwendet werden.
- Das Operating System des Players und des Servers muss neu geschrieben werden.
- Es soll bewertet werden wie viele Punkte man zu welchem Server vergibt.
- Es soll positiv im Kriterienkatalog geschrieben werden.
- Es muss ein Header und Footer im Kriterienkatalog enthalten sein.
- Es sollen Zeilenumbrüche eingefügt werden.
- Die Punkte Bedienbarkeit und User Interface der Administration sollen zusammengefasst werden.
- Den Punkt Bedienbarkeit auf Useability umbenennen und den Wikipedia Artikel darüber durchlesen.
- Für jedes Kriterium soll zuerst die Beschreibung des Kriteriums angeführt werden und erst dann die Begründung.

Aufbau der Marktanalyse:

- Server
- Operating System des Servers
- Player
- Operating System des Players, Ressourcenverbrauch und Tauglichkeit für Einplatinencomputer
- Einbindung externer Quellen
- User Interface für den Benutzer der die Player verwaltet
- Multimonitor

-
- Touchfunktion

Diplomarbeitsprotokoll

Emrah Gudic 04.12.2015

Anwesende: Hans-Christian Hammer, Michael Manner, Matthias Derntl,

Thomas Stütz, Cristopher Hornsey, Reuben Meixner, Emrah Gudic

Teamleiter: Cristopher Hornsey

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Antworten zu den noch offenen Fragen der Diplomarbeit:

- Server und Player für den Tag der offenen Tür zusammenfügen.
- Es wird das VCS von Subversion, welches in der Firma HMMG läuft, zur Verfügung gestellt.
- Vorstellung der graphenorientierten Datenbank für Herrn Manner
- Kriterienkatalog fertigschreiben mit Herrn Hammer Herr Stütz will sich jetzt jede Woche am Freitag mit uns treffen.
- Protokolle, Meilensteine und alle fertigen Unterlagen, sollen in einer Mappe sein.
- Es sollen Uhrzeiten zu den Relationen gespeichert werden können, um beispielsweise zu wissen, wann man in einen Raum kann und wann nicht.
- Unidirektionale Konten müssen ebenfalls möglich sein zum Beispiel für den Notausgang, da man bei dem nicht mehr rein kommt.
- Anschlussknoten auf andere Stockwerk sind sehr wichtig zum Beispiel für die Stiegen.

-
- Es sollen Farben am Knoten hinterlegt werden.
 - Im REDMINE kann die aufgewendete Zeit für einzelne Teilaufgaben gespeichert werden.
 - Es gibt nur eine HTML-Datei in der die Scripts und die Stylesheets enthalten sind.

Allgemeines:

- ALLE fertigen Unterlagen sollen ausgedruckt werden.
- Die geplante Struktur des VCS muss im Anhang der Diplomarbeit dokumentiert sein.
- Der nächste Termin ist am 21.12.2015 um 13:00 Uhr

TODO:

- Im JAVA FX Projekt muss das Canvas fertig sein und das Importieren eines Bildes für das Canvas
- Im JAVA FX Projekt muss man Punkte und Relationen setzen können
- Das Datenmodell muss bis 10.12.2015 fertig sein.
- Der Export muss nicht voll funktionsfähig sein.
- Es soll über RHINO nachgelesen werden.

Emrah Gudic 03.02.2016

Diplomarbeitsprotokoll

Anwesende: Hans-Christian Hammer, Thomas Stütz, Matthias Derntl,

.. Liebitzer, Cristopher Hornsey, Emrah Gudic

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Teamleiter: Cristopher Hornsey

Änderungen am JAVA-FX Client

- Die Stiegen sollen im Editor eine andere Farbe haben
- Die Markierung der angeklickten Sachen muss besser sichtbar sein.
- Ein Codefehler bei den Liften ist geschehen. Der Lift below und above ist nicht änderbar.
- Die Bilderkonvertierung soll im base64 im JSON erfolgen.
- Der Wert für die Checkbox für den bidirektionalen Weg muss standradmäßig auf true sein.
- Im JSON müssen beim TimeTo die führenden 0er bei den Minuten weg.
- Der JAVA-FX Client funktioniert ohne die Installation einer neo4j-Datenbank.
- Es soll die Eingabe eines Maßstabes im Floor möglich sein.

Änderungen am JS-Client

- Der Client muss mehrsprachig sein.
- Der Endknoten soll anders sein als die anderen Knoten die angezeigt werden.
- Der Startknoten muss ebenfalls anders sein als die anderen angezeigten Knoten.
- Das Textfeld für die Eingabe des Startortes wird nicht benötigt.
- Der Weiter und Zurückbutton soll heller sein, wegen dem Highlighting.
- Es sollen Überlegungen angestellt werden über eine einfache Eingabe des Zielorts.

Allgemeines

Es sollen Screenshots an Herrn Hammer gesendet werden. Die E-Mail-Adresse lautet: hammer@hmmg.at.

Emrah Gudic 23.02.2016

Diplomarbeitsprotokoll

Anwesende: Hans-Christian Hammer, Thomas Stütz, Matthias Derntl,

Reuben Meixner ,Cristopher Hornsey, Emrah Gudic

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Teamleiter: Cristopher Hornsey

Änderungen am JAVA-FX Client

- Gebäude zum Editor hinzufügen.
- Import hinzufügen
 - Ein Problem sind die möglichen Überschreibungen
 - * Auf Ebene der Floors sollen nur die Floors importiert werden, die nicht vorhanden sind.
 - Man könnte auch zwei Varianten nehmen, die man optional beim Import wählen kann.
 - * Überschreiben aller Floors
 - * Importieren der Floors die fehlen.
- Ein Deletbutton soll erstellt werden, der alles aus dem Editor löscht
- Checkbox zum Einblenden der Länge der Relationen
- Versionsnummer für den JSON-Export auch wenn dieser im JS-Client nicht benötigt wird. Beim Import des JSON in den Editor wird überprüft, ob die Versionsnummer übereinstimmt. Beim Import der Datei eine soll aussagekräftige Fehlermeldung enthalten sein.

-
- Es sollen Demodaten für die HTL-Leonding angelegt werden, damit man nicht immer alles neu schreiben muss im Editor.
 - Neue Felder sollen zum JAVA-FX Client hinzugefügt werden. Das wären die Felder CreateDate und UpdateDate.
 - Es soll recherchiert werden, ob man in der neo4j-Datenbank mehrere Graphen nebeneinander haben kann.

Allgemeines

- Der bestehende Export ist in Ordnung

Emrah Gudic 09.03.2016

Diplomarbeitsprotokoll

Anwesende: Hans-Christian Hammer, Thomas Stütz, Matthias Derntl, Ewald Dannerer, Reuben Meixner ,Cristopher Hornsey, Emrah Gudic

Betreuungslehrer: Thomas Stütz

Auftraggeber: Hans-Christian Hammer

Teamleiter: Cristopher Hornsey

Geforderte Änderungen am JavaScript-Client

- Im Hintergrund der Stationen die Farben zu der Station z.B. Chirurgie hat die Farbe rot
- Drucken des Canvas (seh wichtig!!)
 - Mit Herrn Derntl ist ausgemacht, dass dies über die config-Datei geschehen kann.
- Überschrift des aktuellen Standorts bei der Stations-section
- Buttons gleich lassen

-
- Die Überschrift bei den Unterstationen soll in der Farbe der ausgewählten Station sein.
 - Die Verteilung an Stationen bei beispielsweise einem Liftausfall muss aus dem MMDS gemacht werden.
 - Der Aufbau steht in der config-Datei
 - schmälere Buttons in den Listen
 - größere Schrift in den Listen
 - Es sollen 2 Listen nebeneinander Platz haben.
 - Von Herrn Ing. Dannerer bekommen wir die Liste der obersten Ebene der Organisationseinheiten.
 - Die Farbcodes bekommen wir ebenfalls von Herrn Ing. Dannerer