

**Fachrichtung Informatik**

**Schuljahr 2023/2024**

**DIPLOMARBEIT**

Gesamtprojekt

**Indoor Navigation - Fill Navi**

**Ausgeführt von:**

Michael Aigner, 5AHIF-1

Sebastian Raith, 5AHIF-12

Felix Weigert, 5AHIF-18

**Betreuer/Betreuerin:**

DI (FH) Krispin Hable

Grieskirchen, am 04.04.2024

---

Abgabevermerk:  
Datum: 04.04.2024

Betreuerin/Betreuer:  
DI (FH) Krispin Hable

## **Eidesstaatliche Erklärung**

„Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.“

Griekskirchen, 04.04.2024

---

VerfasserInnen: Vor- und Zunamen

---

VerfasserInnen: Vor- und Zunamen

---

VerfasserInnen: Vor- und Zunamen

## **Genderhinweis**

Bei allen Bezeichnungen, die auf Personen bezogen sind, meint die Formulierung alle bekannten Geschlechter, unabhängig von der in der Formulierung verwendeten konkreten geschlechtsspezifischen Bezeichnung. Zur besseren Lesbarkeit wird auf eine politisch korrekte Schreibweise beispielsweise mit ‚-Innen‘, ‚-innen‘ und ‚-in‘ verzichtet. Selbstverständlich richten sich die Inhalte auch an alle Personen jeglichen Geschlechtes.

# DIPLOMARBEIT

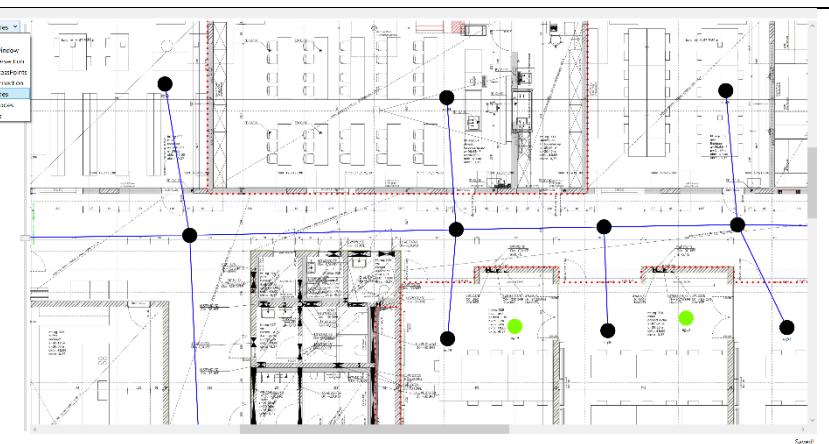
## DOKUMENTATION

<b>Namen der Verfasser/innen</b>	Michael Aigner Sebastian Raith Felix Weigert
<b>Jahrgang Schuljahr</b>	5AHIF 2023/2024
<b>Thema der Diplomarbeit</b>	Fill-Navi – Indoor Navigation
<b>Kooperationspartner</b>	Fill GmbH Fillstraße 1, 4942 Gurten Hauptansprechpartner: Fabian Wilflingseder

<b>Aufgabenstellung</b>	Ziel ist die Evaluierung (und Umsetzung) einer Indoor-Navigationsapp für die Firma Fill. Neue Mitarbeiter/Lieferanten – also Personen, die nicht mit dem Aufbau des Fill-Gebäudes vertraut sind, haben derzeit keine Möglichkeit sich schnell und einfach ohne Hilfe eines anderen zurecht zu finden bzw. einen bestimmten Raum zu finden. Die Firma Fill möchten wissen, ob eine Navigation per Handy möglich und sinnvoll ist.
-------------------------	--

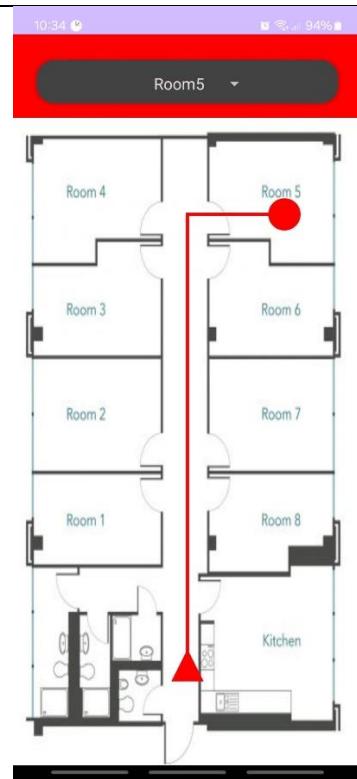
<b>Realisierung</b>	Wir haben 3 Teilprojekte realisiert. Wir haben einen Karteneditor, einen Server und eine Android App programmiert. Der Karteneditor erfüllt die Funktion, Wegpunkte und APs, die später für die Navigation benötigt werden auf eine beliebige Karte einzuziehen. Der Server übernimmt die Kommunikation zwischen Karteneditor und der Android App. Weiters übernimmt er die Aufgabe, der Berechnung der Navigationsroute. Die Android App wird von dem Endnutzer verwendet und dient der Navigation durch das Gebäude.
---------------------	--

<b>Ergebnisse</b>	Als Ergebnis haben wir erfolgreich unsere 3 Teilprojekte umgesetzt. Wir haben ein funktionierendes Proof of Concept für die Firma Fill erstellt. Wir haben dabei die Umgebung und sämtliche UmgebungsvARIABLEN zu unserem Vorteil manipuliert um damit, so einfach und ressourcensparend wie möglich, die Funktionalität einer Indoor-Navigation umzusetzen.
-------------------	--



**Typische Grafik, Foto etc.  
(mit Erläuterung)**

Der Karteneditor erfüllt die Aufgabe, z.B. einen Gebäudeplan mit den nötigen Nodes (APs, Knotenpunkte) und Verbindungen zu versehen, damit diese Datei vom Backend und in weiterer Folge von der Navigations-App verwendet werden kann. Im Screenshot sieht man links oben auf einen Blick, welche Optionen der Nutzer beim "Bearbeiten" der Grafik hat. Die schwarzen Punkte (Nodes) und die blauen Verbindungslien sind das visuelle Resultat der Bearbeitung.



Der Screenshot der Navigations-App zeigt eine "idealisierte" Benutzeroberfläche für den Nutzer der Indoor-Navigation. Im Dropdown-Menü kann ein Zielraum ausgewählt werden. Die rote Linie zeigt den schnellsten Weg zum Ziel.

**Teilnahme an Wettbewerben,** Edison Junior

**Auszeichnungen** <https://www.edison-der-preis.at/edison-junior>

**Möglichkeiten der Einsichtnahme in die Arbeit**

**Approbation**

Prüfer/Prüferin

Direktor/Direktorin

(Datum/Unterschrift)		
----------------------	--	--

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung .....</b>	<b>9</b>
1.1	Arbeitgeber Fill.....	9
1.1.1	Firmengeschichte .....	9
1.2	Projektteam .....	10
1.3	Motivation/Hintergründe.....	10
1.3.1	Indoor-Navigation .....	11
1.3.2	Proof of Concept .....	11
1.3.3	APs.....	11
1.3.4	SPF.....	12
1.4	Aufgabenstellung .....	13
<b>2</b>	<b>Ist-Zustand .....</b>	<b>14</b>
2.1	Probleme/Vorteile .....	14
<b>3</b>	<b>Soll-Zustand .....</b>	<b>15</b>
3.1	Anforderung.....	15
<b>4</b>	<b>Planung/Konzept .....</b>	<b>15</b>
4.1	Projektmanagement .....	16
4.1.1	Kommunikationstools.....	17
4.2	Karten-Editor .....	17
4.3	Backend .....	18
4.4	Android-App .....	18
<b>5</b>	<b>Umsetzung .....</b>	<b>18</b>
5.1	Entwicklungsumgebungen.....	18
5.1.1	Visual Studio 2022.....	19
5.1.2	Android-Studio .....	19
5.2	Tools/Programme .....	19
5.2.1	GeoGebra.....	19
5.2.2	Ngrok .....	19
5.3	Karten-Editor .....	20
5.3.1	Main Project.....	20
5.3.2	EditorLib.....	24
5.4	Backend .....	56
5.4.1	Nuggets.....	56

5.4.2	Klassen/Methoden/Variablen.....	57
5.5	Android-App .....	68
5.5.1	Base64Handler .....	69
5.5.2	MainActivity.....	71
5.5.3	NavigationView .....	75
5.5.4	NodeData.....	78
<b>6</b>	<b>Ergebnis .....</b>	<b>79</b>
6.1	Karten-Editor .....	79
6.2	Backend .....	80
6.3	Android-App .....	80
6.4	Ablauf/Funktion.....	80
6.4.1	Karteneditor Ablauf.....	80
6.4.2	Backend Funktionen.....	81
6.4.3	Android App Ablauf.....	81
6.5	Conclusio .....	81
<b>7</b>	<b>Resümee .....</b>	<b>82</b>
7.1	Probleme/Problemlösungsstrategien .....	82
7.2	Lessons Learned.....	83
7.2.1	Michael Aigner .....	83
7.2.2	Sebastian Raith.....	83
7.2.3	Felix Weigert.....	83
<b>8</b>	<b>Verwendete Lösungswege mit Begründung/Erklärung .....</b>	<b>84</b>
<b>9</b>	<b>Vorschläge/Empfehlungen Weiterentwicklung .....</b>	<b>85</b>
<b>10</b>	<b>Statement des Auftraggebers .....</b>	<b>85</b>
<b>11</b>	<b>Danksagung .....</b>	<b>86</b>
	<b>Quellen und Literaturverzeichnis .....</b>	<b>87</b>
	<b>Abbildungsverzeichnis.....</b>	<b>89</b>

# **1 Einleitung**

Die Notwendigkeit präziser Indoor-Navigation in Unternehmen wird mit zunehmender Vernetzung und Komplexität der Gebäudestrukturen immer dringlicher. Das trifft bei einem so schnell wachsenden Unternehmen wie Fill genau zu. Insbesondere für externe Besucher und Mitarbeiter stellt die Orientierung oftmals eine Herausforderung dar. Diese Diplomarbeit konzentriert sich darauf, eine Indoor-Navigationslösung für Fill zu entwickeln und zu evaluieren. Ein Proof of Concept für Android wurde erstellt, um die Machbarkeit mit einfachsten Mitteln zu prüfen. Die Arbeit erläutert den theoretischen Hintergrund, analysiert Anforderungen und Herausforderungen, beschreibt den Entwicklungsprozess und präsentiert die Ergebnisse. Diese Arbeit trägt zur Vertiefung des Verständnisses für Indoor-Navigation in Unternehmen und zur Umsetzung unternehmensspezifischer Anforderungen bei.

## **1.1 Arbeitgeber Fill**

Die Erfolgsgeschichte begann 1966 als Josef Fill den Schlossereibetrieb "Josef Fill Maschinenbau" gründete. Viele Jahre und Meilensteine später ist die Firma Fill in diesem Wirtschaftssektor nicht mehr wegzudenken. Für über 1000 Menschen ist Fill ein starker Arbeitgeber, der stets am Puls der Zeit agiert. Fill steht schon lange nicht mehr nur auf einem Standbein, sondern hat ein großes Spektrum an Geschäftsbereichen. Ein Teil dieses Spektrums ist es, junge Menschen für Fill zu begeistern und daher war Fill für unsere Diplomarbeit sofort die erste Wahl. [1]

### **1.1.1 Firmengeschichte**

Wie bereits erwähnt gibt es Fill nicht erst seit gestern. Nach der Gründung 1966 folgte 1970 der erste Meilenstein in dem Fill sich in der Sparte Maschinenbau vor allem für die Skiindustrie durchgesetzt hat. Schon 1975 kam der erste Commodore Computer zum Einsatz, diese Vorreiterrolle hat Fill bis heute inne. 1986 hatte Josef Fill 117 Mitarbeiter, diese erwirtschafteten beachtliche 82 Millionen Schilling (6 Mio. Euro). Im Lauf der Jahre erfolgte in der Konstruktion der Umstieg vom Zeichenbrett auf IT-Programme. 1990 entstand die erste Anlage zum Gießen und Bearbeiten von Aluminium-Bauteilen für eine Linzer Gießerei. Die Entwicklung der Entkernungsanlage SWINGMASTER öffnete Fill schließlich die Tür zur internationalen Gießereiwelt, speziell für die Autoindustrie. 1999 wurde mit der ersten Produktionsanlage für Türscharniere der Grundstein für die Metallbearbeitungstechnik gelegt. Der Mitarbeiterstand betrug bereits 200 Beschäftigte. Im Jahr 2000 übernahm schließlich Andreas Fill die Fill GmbH von seinem Vater und führt das Unternehmen mit visionären Zielen bis heute fort. Um 2014 hat sich Fill in der Gießereitechnik, der Entkern- und Holzbandsägetechnologie

sowie für Ski- und Snowboard-Produktionsmaschinen zum Technologie- und Innovationsführer etabliert. Die internationale Automobil- und Luftfahrtindustrie sind bedeutende Auftraggeber. Die Exportquote liegt bei 90%. In China und Mexiko wurden Vertriebs- und Serviceniederlassungen gegründet. Produktionsflächen-Erweiterungen und der Neubau eines Digitalisierungszentrums – der ‘Fill Future Zone’ in Gurten folgten. Neben Andreas Fill als CEO und Eigentümer wurden 2021 drei neue Geschäftsführer ernannt - Martin Reiter, Günter Redhammer und Alois Wiesinger. Eine neue Niederlassung in den USA, Plymouth in Detroit, treibt die Internationalisierung voran. [1]

## 1.2 Projektteam

<b>Felix Weigert</b>	<b>Projektleiter</b>
<b>Michael Aigner</b>	<b>Projektmitglied</b>
<b>Sebastian Raith</b>	<b>Projektmitglied</b>
<b>DI(FH) Krispin Hable</b>	<b>Betreuer</b>
<b>Fabian Wilflingseder</b>	<b>Firmenansprechpartner</b>

## 1.3 Motivation/Hintergründe

Die Motivation zur Entwicklung einer Indoor-Navigationslösung für Fill Gurten ergibt sich aus der Notwendigkeit, die Effizienz und Produktivität in einem dynamischen Arbeitsumfeld zu steigern. Mit einer komplexen Gebäudestruktur und einer Vielzahl von Abteilungen, Besprechungsräumen und Produktionsstätten ist Fill Gurten ein Unternehmen, in dem die Navigation für Mitarbeiter, Besucher und Lieferanten öfter eine Herausforderung darstellen kann. Durch die Implementierung einer Indoor-Navigationslösung strebt Fill Gurten danach, die Zeit, die für die Orientierung benötigt wird, zu minimieren und die Interaktionen zwischen Mitarbeitern und externen Partnern zu verbessern. Die Entwicklung einer maßgeschneiderten Indoor-Navigationslösung ist daher ein logischer Schritt, um diesen Zielen gerecht zu werden.

### **1.3.1 Indoor-Navigation**

Indoor-Navigation ist die Technologie, die entwickelt wurde, um Personen innerhalb von Gebäuden oder geschlossenen Räumen zu führen und zu navigieren. Im Gegensatz zur herkömmlichen Navigation im Freien, die auf GPS basiert, sind die Signale von Satelliten in Innenräumen oft zu ungenau oder erst gar nicht verfügbar. Daher erfordert die Indoor-Navigation spezielle Ansätze, um genaue Standortinformationen bereitzustellen. Typische Technologien, die in der Indoor-Navigation eingesetzt werden, sind Bluetooth, WLAN, Infrarot, Ultraschall und RFID. Diese Technologien ermöglichen es, die Position von Personen oder Objekten innerhalb eines Gebäudes zu bestimmen und ihnen anschließend durch visuelle oder akustische Hinweise den Weg zu weisen. Indoor-Navigation findet Anwendung in verschiedenen Bereichen, darunter Einzelhandel, Gesundheitswesen, Logistik, Flughäfen und Unternehmen, um die Effizienz, Sicherheit und Benutzererfahrung zu verbessern.

### **1.3.2 Proof of Concept**

Ein Proof of Concept (PoC) ist eine Testumgebung oder eine experimentelle Implementierung, die entwickelt wurde, um die Machbarkeit einer Idee, eines Konzepts oder einer Technologie zu demonstrieren. Das Hauptziel eines Proof of Concept besteht darin, zu überprüfen, ob eine bestimmte Lösung funktioniert und den gewünschten Zweck erfüllt, ehe größere Ressourcen für die vollständige Umsetzung investiert werden. Im Kontext dieser Arbeit wurde ein Proof of Concept erstellt, um zu evaluieren, ob eine Indoor-Navigationslösung für die Firma Fill mit einfachsten Mitteln realisierbar ist. Durch die Entwicklung eines einfachen Prototyps für Android kann die Funktionalität der Lösung überprüft werden. Der Proof of Concept dient somit als Ausgangspunkt für weitere Entwicklungen.

### **1.3.3 APs**

Ein Access Point (AP) ist ein Gerät, das in drahtlosen Netzwerken verwendet wird, um Verbindungen zwischen drahtlosen Endgeräten (wie Laptops, Smartphones, Tablets usw.) und einem kabelgebundenen Netzwerk herzustellen. Im Wesentlichen dient ein Access Point als zentraler Knotenpunkt, der drahtlose Geräte mit einem bestehenden kabelgebundenen Netzwerk verbindet, um drahtlosen Zugriff auf Ressourcen wie das Internet, Dateien, Drucker usw. zu ermöglichen. In unserem Fall sind sie ein wichtiger Baustein für unsere Navigations- und Positionsbestimmungs-Applikation.

### 1.3.4 SPF

Shortest Path First (SPF) ist ein Algorithmus, der dazu dient, den kürzesten Pfad zwischen zwei Knoten in einem gewichteten Graphen zu finden. Der Algorithmus wird häufig in verschiedenen Anwendungen wie Netzwerkrouting, Verkehrsflussoptimierung und Logistik eingesetzt.

Der SPF-Algorithmus basiert im Wesentlichen auf der Idee, Schritt für Schritt den Pfad zu aktualisieren, bis der kürzeste Pfad zwischen dem Startknoten und allen anderen Knoten im Graphen gefunden ist. Typischerweise verwendet SPF eine Datenstruktur wie eine Prioritätswarteschlange, um die nächsten zu untersuchenden Knoten effizient auszuwählen.

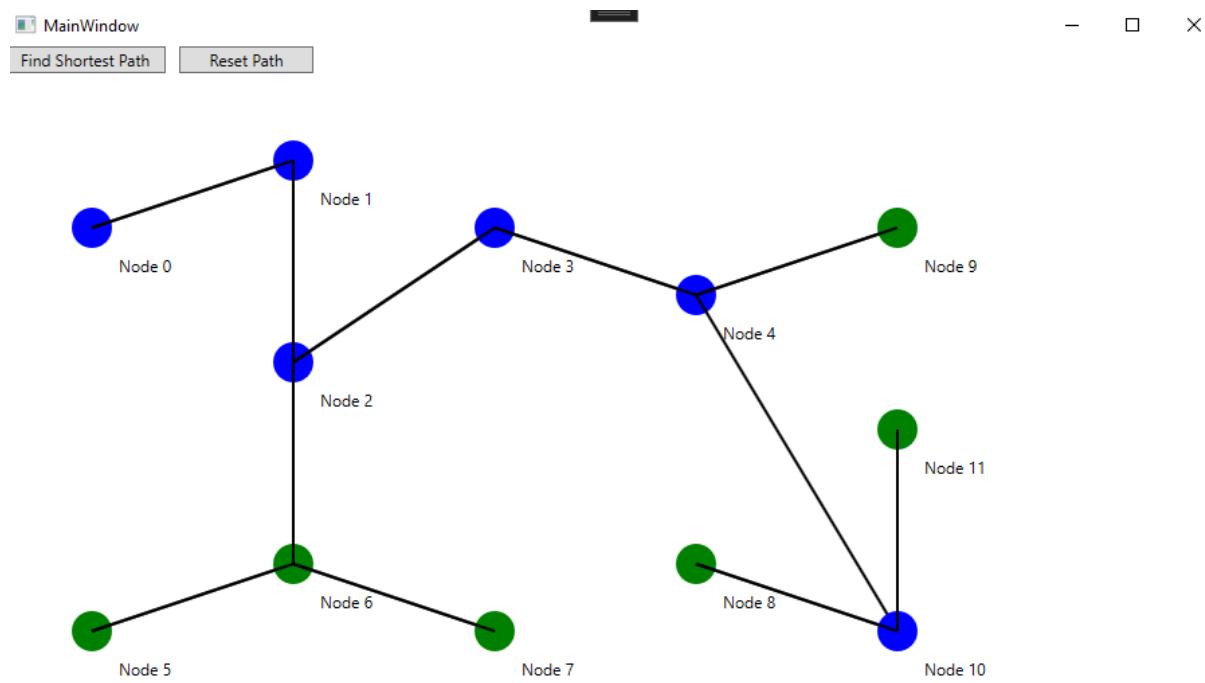


Abbildung 1: SPF - SPF Prototyp-App

#### 1.3.4.1 A\*

Der A\*-Algorithmus ist eine erweiterte Variante des Shortest Path First (SPF)-Algorithmus. Im Gegensatz zu den klassischen SPF-Algorithmen wie Dijkstra oder Bellman-Ford, die alle möglichen Pfade systematisch untersuchen, verwendet A\* eine heuristische Schätzung, um den Suchraum zu reduzieren und effizienter zum Ziel zu navigieren. A\* ist in unserem Fall der passendere Algorithmus, weil beim A\* die Entfernung zwischen den Nodes zur Berechnung verwendet wird (mit gewichteten Nodes). A\* verwendet eine Heuristik, um die Schätzung der verbleibenden Kosten bis zum Ziel zu be-

rechnen, während Dijkstra keine solche Heuristik verwendet. Die Heuristik im A\*-Algorithmus ermöglicht es ihm, den Suchraum effizienter zu durchsuchen, indem er tendenziell in Richtung des Ziels navigiert. Dijkstra hingegen betrachtet alle Knoten gleich und untersucht sie systematisch, ohne eine spezifische Richtung zu bevorzugen. [2]

#### 1.3.4.2 Dijkstra

Initialisierung: Der Algorithmus beginnt damit, die Kosten für den kürzesten Pfad vom Startknoten zu allen anderen Knoten im Graphen zu initialisieren. Der Startknoten hat dabei Kosten von 0, und alle anderen Knoten haben vorerst unendliche Kosten.

Auswahl des nächsten Knotens: In jedem Schritt wählt der Algorithmus denjenigen Knoten aus, der die geringsten bisher bekannten Kosten hat und noch nicht als "besucht" markiert ist.

Aktualisierung der Kosten: Für den ausgewählten Knoten betrachtet der Algorithmus alle benachbarten Knoten und aktualisiert deren Kosten, falls ein neuer Pfad zu einem benachbarten Knoten mit niedrigeren Kosten gefunden wird. Dies geschieht, indem die Kosten des aktuellen Knotens und die Gewichtung der Kante zwischen den Knoten addiert werden.

Markierung des Knotens: Nachdem alle benachbarten Knoten aktualisiert wurden, wird der aktuelle Knoten als "besucht" markiert, um sicherzustellen, dass er nicht erneut untersucht wird.

Wiederholung: Die Schritte 2 bis 4 werden so lange wiederholt, bis alle Knoten im Graphen besucht wurden oder bis das Ziel erreicht ist.

Pfadrekonstruktion: Nachdem der Algorithmus beendet ist, können die kürzesten Pfade von jedem Knoten zum Startknoten anhand der aktualisierten Kosten rekonstruiert werden.

## 1.4 Aufgabenstellung

Die vorliegende Diplomarbeit hat das Ziel, eine Indoor-Navigationslösung für die Firma Fill zu evaluieren und ein Proof of Concept zu entwickeln. Die Aufgabenstellung hat mehrere Schwerpunkte:

Anforderungsanalyse: Eine umfassende Analyse der Anforderungen die eine Indoor-Navigationslösung für Fill hat.

Technologieauswahl: Bewertung und Auswahl der geeigneten Technologien und Plattformen für die Implementierung der Indoor-Navigationslösung.

Entwicklung des PoC: Implementierung eines PoC für Android, um die Machbarkeit der Indoor-Navigationslösung zu demonstrieren. Dies umfasst die Entwicklung von grundlegenden Navigationsfunktionen sowie die Integration der ausgewählten Technologien.

Evaluation und Testung: Durchführung von Tests und Evaluation des PoC in den Räumlichkeiten von Fill. Bewertung der Genauigkeit, Benutzerfreundlichkeit und Leistung der Lösung sowie Identifizierung von Optimierungsmöglichkeiten.

## 2 Ist-Zustand

Der Ist-Zustand ist geprägt von einer fehlenden strukturierten Orientierungsmöglichkeit innerhalb des Unternehmensgeländes. Insbesondere externe Besucher und neue Mitarbeiter können Schwierigkeiten haben sich direkt zurechtzufinden. Es kann vorkommen, dass bestimmte Zielorte, wie Befprechungsräume nicht sofort gefunden werden. Diese Situation kann die Effizienz der Arbeitsabläufe beeinflussen. Nicht nur die Produktivität des Unternehmens kann davon profitieren, das Unternehmen würde wieder einmal unter Beweis stellen, dass es stets mit dem Takt der Zeit geht.

### 2.1 Probleme/Vorteile

Das Fehlen einer Indoor-Navigation beinhaltet wie bereits erwähnt folgende Probleme:

- Negative Beeinflussung der Arbeitsabläufe
- Möglichkeit sich zu verlaufen

Die Implementierung einer Indoor-Navigation kann viele Vorteile haben:

- Effizienzsteigerung: Mitarbeiter, Besucher und Lieferanten können sich schneller und effizienter durch das Gebäude bewegen
- Vermeidung von Verzögerung: Ermöglicht Nutzern sich sofort zurecht zu finden
- Verbesserte Besuchererfahrung: Eine moderne Indoor-Navigation vermittelt einen positiven Eindruck von Fill als innovatives Unternehmen
- Steigerung der Sicherheit: Indoor-Navigation kann den Benutzern dabei helfen schnell und sicher den nächsten Ausgang zu finden

### **3 Soll-Zustand**

Der Sollzustand, der sich aus der erfolgreichen Umsetzung der Aufgabenstellung ergibt, ist folgender:

- Dokumentation der Anforderungen (in dieser Diplomschrift)
- Dokumentation der Technologieauswahl (in dieser Diplomschrift)
- Entwickeltes Proof of Concept für eine Indoor-Navigation
- Dokumentation und Berichterstattung sowie Evaluation und Optimierungsvorschläge

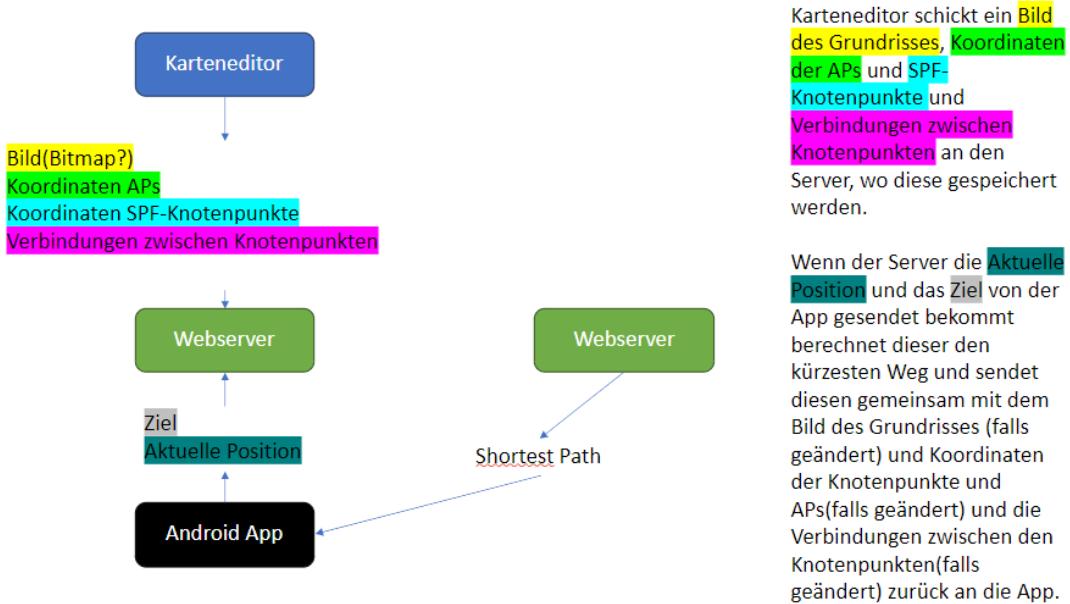
#### **3.1 Anforderung**

Die Anforderungen an die Indoor-Navigation bei Fill sind vielfältig und zielen darauf ab, die identifizierten Probleme des Ist-Zustands zu lösen und den angestrebten Soll-Zustand zu erreichen. Konkret umfassen diese Anforderungen:

- Kartierung der Räumlichkeiten
- Liste von APs
  - Deren Position im Gebäude
  - Deren MAC-Adresse
- Karteneditor
- Server für Kommunikation zwischen Karteneditor und App / Berechnungen
- Android Navigation App

### **4 Planung/Konzept**

Für den ursprünglichen Plan gibt es einen Screenshot. Es handelt sich dabei jedoch nur um einen sehr groben Grundriss und ist lediglich als Resultat eines Brainstormings zu verstehen.



**Abbildung 2: Grundlegende Planung**

Der Screenshot beschreibt grob, wie wir das Projekt aufziehen wollten. Man erkennt die einzelnen Elemente, die wir geplant haben:

- Karteneditor
- Webserver
- Android App

Der grundlegende Ablauf unseres PoC wird UML-Diagramm ähnlich dargestellt.

## 4.1 Projektmanagement

Für die Entwicklung der Indoor-Navigationslösung bei Fill Gurten wurde ein agiler Ansatz gewählt. Angesichts der Natur des Projekts, insbesondere der Notwendigkeit, bei einem Proof of Concept flexibel zu sein und den Technologie Stack je nach den erzielten Ergebnissen anzupassen, war ein agiler Ansatz besonders gut geeignet. Das Agile-Projektmanagement ermöglichte es dem Team, sich schnell an sich ändernde Anforderungen anzupassen. Der Prozess begann mit der Identifizierung von Schlüsselanforderungen und der Planung von kurzen Entwicklungszyklen, sogenannten Sprints. Während jedes Sprints wurden bestimmte Ziele festgelegt, implementiert und anschließend getestet. Regelmäßige Reviews ermöglichen es dem Team, Fortschritte zu bewerten und den Entwicklungsprozess kontinuierlich zu verbessern. Agiles Projektmanagement ist ein iterativer Ansatz, der eine flexible

und effiziente Entwicklung ermöglicht, besonders in Situationen, in denen Anforderungen und Technologien sich im Laufe des Projekts ändern können.

#### 4.1.1 **Kommunikationstools**

Eine effektive Kommunikation ist für den Erfolg agiler Projekte von entscheidender Bedeutung. Insbesondere bei einem agilen Vorgehen, das auf kontinuierliche Zusammenarbeit und schnelle Anpassung setzt, ist eine klare und effiziente Kommunikation unerlässlich. Sie ermöglicht es dem Team, Informationen auszutauschen, Herausforderungen zu besprechen, Fortschritte zu verfolgen und Feedback zu geben. Verschiedene Kommunikationstools können dabei unterstützen, die Zusammenarbeit zu erleichtern und die Effizienz zu steigern. Zu den wichtigen Kommunikationstools für das Projektmanagement gehören:

- Microsoft Teams: Eine Plattform für Chat, Videokonferenzen, Dateifreigabe und Zusammenarbeit, die eine nahtlose Kommunikation und Zusammenarbeit im Team ermöglicht.
- WhatsApp: Eine weit verbreitete Messaging-App, die es den Teammitgliedern ermöglicht, schnell und einfach miteinander zu kommunizieren, insbesondere für informelle Gespräche und schnelle Abstimmungen.
- Discord: Eine Plattform, die ursprünglich für Gamer entwickelt wurde, bietet Funktionen für Text- und Sprachchat sowie Videoanrufe und ermöglicht eine informelle und gleichzeitig effektive Kommunikation im Team.
- Outlook: Ein E-Mail- und Kalenderprogramm, das für formelle Kommunikation, Terminplanung und die Verwaltung von Aufgaben und Ereignissen verwendet wird.

Die Kombination dieser Kommunikationstools bietet dem Team eine vielfältige Palette von Möglichkeiten, um effektiv miteinander zu kommunizieren und sicherzustellen, dass alle Teammitglieder auf dem neuesten Stand sind und ihren Beitrag zum Projekt leisten können.

#### 4.2 **Karten-Editor**

Wir haben die Grundfunktionen des Karteneditors festgelegt, darunter das Öffnen und Bearbeiten von Grafiken sowie das Hinzufügen, Verschieben und Löschen von Knotenpunkten und Access Points. Mit fortschreitender Zeit haben wir zusätzliche Funktionen identifiziert und eingestuft, wie das Bearbeiten von Knotenpunkten und Access Points sowie die Implementierung von Remove-

Funktionalitäten. Während des Entwicklungsprozesses haben wir die Benutzeroberfläche im Rahmen unseres UI/UX-Unterrichts überprüft und optimiert, um eine intuitive Bedienung sicherzustellen und das Nutzererlebnis zu verbessern.

### **4.3 Backend**

Bei der Planung vom Backend haben wir eine iterative Vorgehensweise gewählt. Unsere Grundidee war, Berechnungen für den SPF auf das Backend auszulagern, um Ressourcen am Handy zu sparen. Weiters war geplant, dass das Backend den Austausch von Karteneditor und Android-App übernimmt. Das Backend war einer der Bestandteile, die sich am öftesten unseren Anpassungen und vor allem Technologie Stack Änderungen, anpassen musste. Daher wurden immer wieder neue Prototypen entworfen, um neue Technologien und Vorgehensweisen auszutesten. Machbarkeitsprototypen waren der Grundbestandteil bei der Entwicklung und Verbesserung unseres Backend.

### **4.4 Android-App**

Für die Planung der Android Navigations-App haben wir eine iterative Vorgehensweise gewählt. Wir haben die Grundfunktionen wieder vorab festgelegt. Im Laufe des Projekts sind weitere dazu gekommen. Zunächst haben wir sämtliche Funktionen in eigenen Prototypprojekten vorab ausprogrammiert, um ihre Machbarkeit zu testen und potenzielle Herausforderungen zu identifizieren. Durch diesen Ansatz konnten wir frühzeitig wichtige Erkenntnisse gewinnen und die Funktionalitäten evaluieren, um anschließend gezielte Verbesserungen vorzunehmen.

## **5 Umsetzung**

### **5.1 Entwicklungsumgebungen**

Die Auswahl der richtigen Entwicklungsumgebung ist entscheidend für die effiziente Entwicklung. Es werden nur die Entwicklungsumgebungen angeführt, die auch tatsächlich in der finalen Lösung eingesetzt werden. Jede dieser Umgebungen bietet spezifische Funktionen und Werkzeuge, die den Entwicklungsprozess unterstützen und die Anforderungen des Projekts erfüllen. Durch die Nutzung dieser Entwicklungsumgebungen konnte das Entwicklungsteam optimal zusammenarbeiten und gleichzeitig sicherstellen, dass die erstellte Lösung den höchsten Qualitätsstandards entspricht.

### **5.1.1 Visual Studio 2022**

Seit der vierten Klasse arbeiten wir unter anderem mit Visual Studio 2022. Es handelt sich hierbei um die neueste Version der IDE von Microsoft. Sie ermöglicht die Entwicklung von Anwendungen für eine Vielzahl an Plattformen. Man kann plattformübergreifende Anwendungen für Windows, Web, Mobile und mehr entwickeln, debuggen und bereitstellen. Die IDE bietet ein umfassendes Spektrum an Tools, die den Entwickler in fast allen Bereichen des Codens unterstützen können. [2]

### **5.1.2 Android-Studio**

Android Studio ist die offizielle Entwicklungsumgebung von Google für die Entwicklung von Android-Apps. Diese IDE lernten wir in der HTL-Grieskirchen in der dritten Klasse kennen. Sie bietet ein großes Spektrum an Funktionen, die Entwicklern helfen, hochwertige Android-Anwendungen zu erstellen, zu debuggen und zu testen. Android Studio basiert auf der IntelliJ IDEA-Plattform. [3]

## **5.2 Tools/Programme**

### **5.2.1 GeoGebra**

GeoGebra ist eine kostenlose, dynamische Mathematiksoftware, die für verschiedene Plattformen verfügbar ist, einschließlich Desktop, Web und Mobile. Diese vielseitige Software ermöglicht es Benutzern, mathematische Konzepte zu visualisieren und zu untersuchen, indem sie Diagramme, Grafiken und interaktive Elemente erstellen. Mit GeoGebra können Benutzer mit Algebra, Geometrie und mehr arbeiten und dabei komplexe mathematische Probleme lösen. [4]

GeoGebra erwies sich als unverzichtbares Hilfsmittel während der Entwicklung unserer Indoor-Navigationslösung. Insbesondere bei der Positionsbestimmung und Überprüfung von Ergebnissen spielte dieses vielseitige Tool eine entscheidende Rolle. Durch die Nutzung von GeoGebra konnten wir nicht nur präzise Standortdaten visualisieren und analysieren, sondern auch komplexe Berechnungen durchführen und die Genauigkeit unserer Lösung überprüfen.

### **5.2.2 Ngrok**

Ngrok ist eine leistungsstarke Tunneling-Software, die es Benutzern ermöglicht, lokale Server oder Dienste sicher und flexibel über das Internet zu veröffentlichen. Mit Ngrok können Entwickler lokale Webserver, APIs oder andere Anwendungen schnell und einfach freigeben, ohne komplexe Netzwerkconfigurationn vornehmen zu müssen. [5]

Ngrok wurde uns von unserem Professor (Prof. Planberger) empfohlen. Mit seiner Hilfe haben wir es initial aufgesetzt. Ngrok wurde in unserem Fall verwendet, um das Backend zu tunneln. Daher konnte man nun mit unserer Android-App und dem Karten-Editor darauf zugreifen. Davor hatten wir das Problem, dass die Android-App nicht auf das Backend (welches auf localhost lief) zugreifen konnte.

## 5.3 Karten-Editor

### 5.3.1 Main Project

Im Main Projekt befindet sich das MainWindow. In diesem wird Initial der gesamte Content (APs, Grafik, Knotenpunkte) geladen und in die ImageControl eingefügt. Die Services sind hier ebenfalls ausprogrammiert. Nun die Funktionalität im Detail:

Im Screenshot rechts erkennt man die einzelnen Komponenten: Content, Services und MainWindow. Im Ordner Services befinden sich der CommunicatorService, der MessageService und der Save-Service.

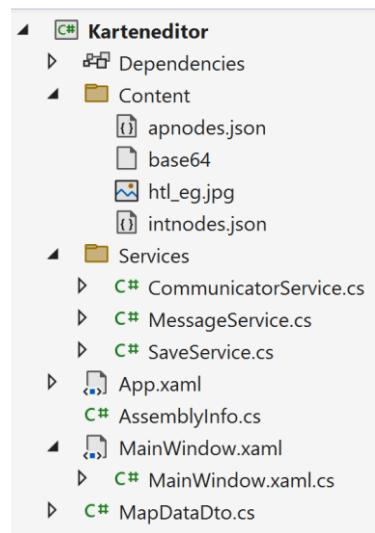


Abbildung 3: Code - Karteneditor  
Übersicht

### 5.3.1.1 Services

- CommunicatorService

```
namespace Karteneditor.Services;
1 reference
internal class CommunicatorService
{
    1 reference
    public HttpClient HttpClient { get; set; }

    0 references
    public CommunicatorService(string apiPath)
    {
        HttpClient = new()
        {
            BaseAddress = new Uri(apiPath)
        };
    }

    0 references
    public void PostMapData(MapDataDto mapData)
    {
        //string json = JsonSerializer.Serialize(mapData);
        //var stringContent = new StringContent(json, Encoding.UTF8, "application/json");

        //using var response = HttpClient.PostAsync("/ImageCrl/mapData", stringContent).Result;
        //response.EnsureSuccessStatusCode();
    }
}
```

Abbildung 4: Code - Karteneditor CommunicatorService

Dieser Service ist für das Uploaden der Daten auf den Server vorgesehen.

```
namespace Karteneditor;
1 reference
public class MapDataDto
{
    0 references
    public List<IntNodeDto> IntNodes { get; set; } = null!;
    0 references
    public List<AccessPointDto> AccessPoints { get; set; } = null!;
    0 references
    public string base64Img { get; set; } = null!;
}
```

Abbildung 5: Code – MapDataDto

- MessageService

```

namespace Karteneditor.Services;
2 references
internal class MessageService(Dispatcher dispatcher, Label lblMessage)
{
    private Dispatcher _dispatcher = dispatcher;
    private Label _lblMessage = lblMessage;

    1 reference
    public void WriteMessage(string message, int millisecondsDelay = 2000)
    {
        var task = Task.Run(() =>
        {
            _dispatcher.Invoke(() => _lblMessage.Content = message);
            Task.Delay(millisecondsDelay).Wait();
            _dispatcher.Invoke(() => _lblMessage.Content = "");
        });
    }
}

```

*Abbildung 6: Code - MessageService*

Dieser Service ist für die Anzeige einer Nachricht in einem Label verantwortlich. Zum Beispiel wird "Saved" beim Speichern angezeigt. Die Variable "millisecondsDelay" hat einen Standardwert von 2000, d.h. eine Message wird standardmäßig für zwei Sekunden im Label angezeigt.

- SaveService

```

namespace Karteneditor.Services;
2 references
public class SaveService(string basePath, ImageControl imageView)
{
    7 references
    private string basePath { get; init; } = basePath;
10 references
    public ImageControl ImageView { get; init; } = imageView;

1 reference
    public void LoadImage() => ImageView.Image = File.ReadAllText(@${basePath}\base64").ToWritableImage();

1 reference
    public void SaveContent()
    {
        var intNodeDtos = ImageView.IntNodes.ToNodeDtoList(ImageView);
        File.WriteAllText(@${basePath}\intnodes.json", JsonSerializer.Serialize(intNodeDtos));

        var accessPointDtos = ImageView.ApNodes.ToNodeDtoList(ImageView);
        File.WriteAllText(@${basePath}\apnodes.json", JsonSerializer.Serialize(accessPointDtos));

        string base64 = Convert.ToBase64String(File.ReadAllBytes(@${basePath}\htl_eg.jpg"));
        File.WriteAllText(@${basePath}\base64", base64);
    }

1 reference
    public void LoadContent()
    {
        var intNodes = JsonSerializer.Deserialize<List<IntNodeDto>>(File.ReadAllText(@${basePath}\intnodes.json"));
        var apNodes = JsonSerializer.Deserialize<List<AccessPointDto>>(File.ReadAllText(@${basePath}\apnodes.json"));

        intNodes.ToList(ImageView).ForEach(ImageView.AddNode);
        apNodes.ToList(ImageView).ForEach(ImageView.AddNode);

        intNodes.ForEach(node => node.ConnectedNodes.ForEach(x => ImageView.AddConnection(node.Id, x)));
    }
}

```

**Abbildung 7: Code – SaveService**

Der SaveService ist für das Laden und Speichern der Daten zuständig. In der "LoadImage" Methode wird die Grafik aus einem Base64-String geladen. Die Methode "SaveContent" speichert den gesamten Content. Die Methode "LoadContent" lädt ihn in das Projekt.

### 5.3.1.2 MainWindow

```

<Window x:Class="Karteneditor.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Karteneditor" xmlns:editorlib="clr-namespace:EditorLib;assembly=EditorLib"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800"
        Loaded="Window_Loaded"
        WindowState="Maximized"
        WindowStyle="SingleBorderWindow">
    <DockPanel>
        <Menu DockPanel.Dock="Top">
            <MenuItem Header="_Datei">
                <MenuItem Header="_Save" Click="OnSaveClicked"/>
            </MenuItem>
        </Menu>
        <Separator DockPanel.Dock="Top"/>
        <StackPanel DockPanel.Dock="Left">
            <ComboBox Name="cboModes"/>
        </StackPanel>
        <Label DockPanel.Dock="Bottom" Name="lblMessage" HorizontalContentAlignment="Right" Content="" />
        <editorlib:ImageControl Name="ImageControl"></editorlib:ImageControl>
    </DockPanel>
</Window>

```

**Abbildung 8: Code - MainWindow XAML**

```

namespace Karteneditor;
/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
3 references
public partial class MainWindow : Window
{
    private readonly string _contentPath = "..\\..\\..\\Content\\";
    //private CommunicatorService _communicator;
    private SaveService _saveService;
    private MessageService _messageService;

    0 references
    public MainWindow() => InitializeComponent();

    1 reference
    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        //_communicator = new("https://localhost:5001");
        _saveService = new(_contentPath, ImageControl);
        _messageService = new(Dispatcher, lblMessage);

        _saveService.LoadImage();
        _saveService.LoadContent();

        Enum.GetValues(typeof(EditModes)).OfType<EditModes>().ToList().ForEach(x => cboModes.Items.Add(x));
        cboModes.SelectionChanged += CboModes_SelectionChanged;
        cboModes.SelectedItem = ImageControl.Mode;
    }

    1 reference
    private void CboModes_SelectionChanged(object sender, SelectionChangedEventArgs e)
    => ImageControl.Mode = (EditModes)cboModes.SelectedItem;

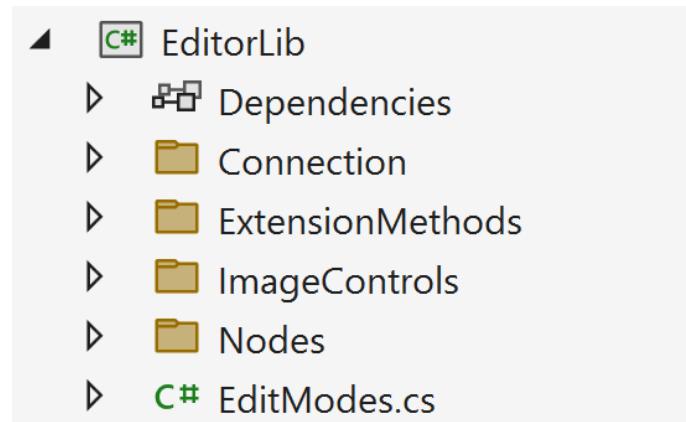
    1 reference
    private void OnSaveClicked(object sender, RoutedEventArgs e)
    {
        _saveService.SaveContent();
        _messageService.WriteMessage("Saved!");
    }
}

```

**Abbildung 9: Code – MainWindow**

Im MainWindow werden in der Window\_Loaded werden alle Services initialisiert. Weiters wird der Content hereingeladen. In OnSaveClicked kann die Bearbeitung der Grafik gespeichert werden.

### 5.3.2 EditorLib



**Abbildung 10: Code - EditorLib Übersicht**

Das EditorLib-Projekt beinhaltet die Logik des Karteneditors.

### 5.3.2.1 EditModes

```
19 references
public enum EditModes
{
    None,
    MoveWindow,
    AddIntersection,
    AddAccessPoints,
    AddConnection,
    EditNodes,
    MoveNodes,
    Remove,
}
```

Abbildung 11: Code - Enum EditModes

### 5.3.2.2 ImageControl

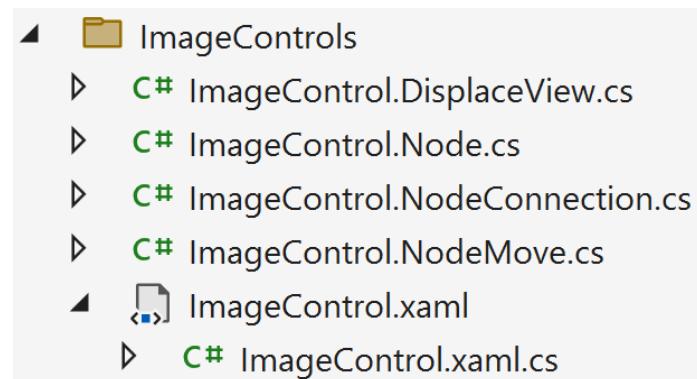


Abbildung 12: Code – ImageControl

ImageControl ist eine Partialclass. Die Logik von den einzelnen Funktionen ist in jeweils einem eigenen File.

- ImageControl.xaml

```

<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Slider Grid.Column="0" Orientation="Vertical"
        HorizontalAlignment="Left" Minimum="1" Maximum="20" x:Name="slider"/>
    <ScrollViewer Name="scrollViewer" Grid.Column="1"
        VerticalScrollBarVisibility="Visible"
        HorizontalScrollBarVisibility="Visible">

        <Grid Name="grid" Width="400" Height="400" RenderTransformOrigin="0.5,0.5">
            <Grid.LayoutTransform>
                <TransformGroup>
                    <ScaleTransform x:Name="scaleTransform"/>
                </TransformGroup>
            </Grid.LayoutTransform>

            <Viewbox Grid.Column="0" Grid.Row="0">
                <Image Name="Img"/>
            </Viewbox>

            <Viewbox Grid.Column="0" Grid.Row="0">
                <Canvas Name="NodeCanvas" Background="Transparent" Focusable="False" />
            </Viewbox>
        </Grid>
    </ScrollViewer>
</Grid>

```

*Abbildung 13: Code - ImageControl XAML*

Das WPF Control Image zeigt die Grafik an. In dem Canvas "NodeCanvas" werden alle Node-Elemente angezeigt.

15 references

```
public static EditModes Mode { get; set; } = EditModes.MoveWindow;
```

*Abbildung 14: Code - EditModes*

In der Variable Mode wird der aktuelleEditMode gespeichert.

```

private WriteableBitmap? _image;
11 references
public WriteableBitmap Image
{
    get => _image;
    set
    {
        _image = value;
        NodeCanvas.Width = Image.Width;
        NodeCanvas.Height = Image.Height;
        Img.Source = _image;
    }
}

```

*Abbildung 15: Code - FullProp Image*

In der Full-Property Image wird die angezeigte Grafik gespeichert.

```
private readonly int _nodeWidth = 50;
```

*Abbildung 16: Code - \_nodeWidth*

Die Breite einer Node.

```

private Node? _lastSelectedNode;
10 references
public Node? LastSelectedNode
{
    get => _lastSelectedNode;
    set
    {
        _lastSelectedNode = value;
        NodeCanvas.Children.OfType<Node>().ToList().ForEach(node => node.FillColor = node.DefaultColor);

        if (LastSelectedNode == null) return;
        LastSelectedNode.FillColor = Brushes.Green;
    }
}

```

*Abbildung 17: Code - FullProp lastSelectedNode*

Speichert die zuletzt angeklickte Node. Die angeklickte Node wird grün eingefärbt, alle anderen Nodes haben die Farbe ihrer DefaultColor.

```

public ImageControl()
{
    InitializeComponent();
    InitializeDisplaceEvents();

    NodeCanvas.MouseUp += OnNodeCanvasMouseUp;
}

```

*Abbildung 18: Code - ImageControl*

"InitializeDisplaceEvents" ist eine Funktion aus dem DisplaceView File. Wenn der NodeCanvas geklickt wird, wird die Methode "OnCanvasMouseUp" aufgerufen.

```

1 reference
private void OnNodeCanvasMouseUp(object sender, MouseEventArgs e)
{
    if (e.Source is not Node && Mode != EditModes.MoveWindow)
        LastSelectedNode = null;

    if (Mode is not (EditModes.AddIntersection or EditModes.AddAccessPoints)) return;

    if (Mode is EditModes.AddIntersection)
    {
        if (e.Source is Node) return;
        AddNode(new IntNode(), e);
    }
    else if (Mode is EditModes.AddAccessPoints)
    {
        if (e.Source is Node) return;
        AddNode(new AccessPointNode(), e);
    }
}

```

*Abbildung 19: Code - OnNodeCanvasMouseUp*

Wenn keine Node geklickt worden ist und man auch nicht den Canvas bewegt, wird die LastSelectedNode auf null gesetzt. Wenn der Mode AddIntersection ist, wird AddNode, eine Methode aus dem Node-File, mit einer IntNode aufgerufen, wenn er AddAccessPoints ist, wird es mit einer AccessPointNode aufgerufen.

- ImageControl.DisplaceView [6]

```

Point? lastCenterPositionOnTarget;
Point? last.mousePositionOnTarget;
Point? lastDragPoint;

```

```

1 reference
public void InitializeDisplaceEvents()
{
    scrollViewer.ScrollChanged += OnScrollViewerScrollChanged;
    scrollViewer.MouseLeftButtonUp += OnMouseLeftButtonUp;
    scrollViewer.PreviewMouseLeftButtonUp += OnMouseLeftButtonUp;
    scrollViewer.PreviewMouseWheel += OnPreviewMouseWheel;

    scrollViewer.PreviewMouseLeftButtonDown += OnMouseLeftButtonDown;
    scrollViewer.MouseMove += OnMouseMove;

    scrollViewer.PreviewMouseDown += OnMouseDown;
    scrollViewer.PreviewMouseUp += OnMouseUp;

    slider.ValueChanged += OnSliderValueChanged;
}

```

**Abbildung 20:** Code - InitializeDisplaceEvents

Registriert alle Events.

```

1 reference
private void OnMouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Middle)
        StartMovingCanvas(e);
}

1 reference
private void OnMouseUp(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Middle)
        EndMovingCanvas();
}

1 reference
private void OnMouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    if (Mode == EditModes.MoveWindow)
        StartMovingCanvas(e);
}

2 references
private void OnMouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    if (Mode == EditModes.MoveWindow)
        EndMovingCanvas();
}

```

**Abbildung 21:** Code - MouseEventArgs

Stellen sicher, dass man den Canvas nur bewegen kann, wenn man entweder den mittleren Mausknopf drückt oder der Mode MoveWindow ist.

```

2 references
private void StartMovingCanvas(MouseEventArgs e)
{
    var mousePos = e.GetPosition(scrollViewer);
    if (mousePos.X <= scrollViewer.ViewportWidth && mousePos.Y < scrollViewer.ViewportHeight)
    {
        scrollViewer.Cursor = Cursors.SizeAll;
        lastDragPoint = mousePos;
        Mouse.Capture(scrollViewer);
    }
}

```

*Abbildung 22: Code - StartMovingCanvas*

Setzt den Mauszeiger auf "Cursors.SizeAll", um das Verschieben des Steuerelements anzuzeigen, speichert die aktuelle Mausposition und erfasst Mausbewegungen, um das Ziehen des Steuerelements zu ermöglichen.

```

2 references
private void EndMovingCanvas()
{
    scrollViewer.Cursor = Cursors.Arrow;
    scrollViewer.ReleaseMouseCapture();
    lastDragPoint = null;
}

```

*Abbildung 23: Code - EndMovingCanvas*

Wird aufgerufen, um das Verschieben des Canvas-Steuerelements zu beenden. Sie setzt den Mauszeiger auf den Standardcursor Cursors.Arrow, gibt die Mauserfassung des ScrollViewer-Steuerelements frei und setzt den letzten Verschiebungspunkt auf null, um das Ziehen zu beenden und die Mausinteraktion zu beenden.

```

1 reference
private void OnMouseMove(object sender, MouseEventArgs e)
{
    if (lastDragPoint.HasValue)
    {
        Point posNow = e.GetPosition(scrollViewer);

        double dX = posNow.X - lastDragPoint.Value.X;
        double dY = posNow.Y - lastDragPoint.Value.Y;

        lastDragPoint = posNow;

        scrollViewer.ScrollToHorizontalOffset(scrollViewer.HorizontalOffset - dX);
        scrollViewer.ScrollToVerticalOffset(scrollViewer.VerticalOffset - dY);
    }
}

```

*Abbildung 24: Code - OnMouseMove*

Wird aufgerufen, wenn der Mauszeiger über dem Canvas-Steuerelement bewegt wird. Überprüft zuerst, ob lastDragPoint einen Wert hat, und berechnet dann die Änderung der Mausposition seit dem letzten Punkt. Diese Änderungen werden verwendet, um das Canvas-Steuerelement horizontal und vertikal zu verschieben.

```
1 reference
private void OnPreviewMouseWheel(object sender, MouseWheelEventArgs e)
{
    lastMousePositionOnTarget = Mouse.GetPosition(grid);

    if (e.Delta > 0)
        slider.Value += 1;
    if (e.Delta < 0)
        slider.Value -= 1;

    e.Handled = true;
}
```

**Abbildung 25:** Code - *OnPreviewMouseWheel*

Wird aufgerufen, wenn sich das Mausrad bewegt. Wenn es sich nach oben bewegt, wird der Wert des Sliders um 1 erhöht, wenn es sich nach unten bewegt um 1 reduziert.

```
1 reference
private void OnSliderValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    scaleTransform.ScaleX = e.NewValue;
    scaleTransform.ScaleY = e.NewValue;

    var centerOfViewport = new Point(scrollViewer.ViewportWidth / 2,
                                      scrollViewer.ViewportHeight / 2);
    lastCenterPositionOnTarget = scrollViewer.TranslatePoint(centerOfViewport, grid);
}
```

**Abbildung 26:** Code - *OnSliderValueChanged*

Wird aufgerufen, wenn sich der Wert des Schiebereglers ändert. Sie aktualisiert das ZoomLevel basierend auf dem neuen Wert.

```

1 reference
private void OnScrollViewerScrollChanged(object sender, ScrollChangedEventArgs e)
{
    if (e.ExtentHeightChange != 0 || e.ExtentWidthChange != 0)
    {
        Point? targetBefore = null;
        Point? targetNow = null;

        if (!lastMousePositionOnTarget.HasValue)
        {
            if (lastCenterPositionOnTarget.HasValue)
            {
                var centerOfViewport = new Point(scrollViewer.ViewportWidth / 2,
                                                scrollViewer.ViewportHeight / 2);
                Point centerOfTargetNow = scrollViewer.TranslatePoint(centerOfViewport, grid);

                targetBefore = lastCenterPositionOnTarget;
                targetNow = centerOfTargetNow;
            }
        }
        else
        {
            targetBefore = lastMousePositionOnTarget;
            targetNow = Mouse.GetPosition(grid);

            lastMousePositionOnTarget = null;
        }

        if (targetBefore.HasValue)
        {
            double dXInTargetPixels = targetNow.Value.X - targetBefore.Value.X;
            double dYInTargetPixels = targetNow.Value.Y - targetBefore.Value.Y;

            double multiplicatorX = e.ExtentWidth / grid.Width;
            double multiplicatorY = e.ExtentHeight / grid.Height;

            double newOffsetX = scrollViewer.HorizontalOffset - dXInTargetPixels * multiplicatorX;
            double newOffsetY = scrollViewer.VerticalOffset - dYInTargetPixels * multiplicatorY;

            if (double.IsNaN(newOffsetX) || double.IsNaN(newOffsetY)) return;

            scrollViewer.ScrollToHorizontalOffset(newOffsetX);
            scrollViewer.ScrollToVerticalOffset(newOffsetY);
        }
    }
}

```

**Abbildung 27:** Code - *OnScrollViewerScrollChanged*

- *ImageControl.Node*

```

public List<IntNode> IntNodes { get; set; } = [];
public List<AccessPointNode> ApNodes { get; set; } = [];

```

**Abbildung 28:** Code - *ImageControlNode*

In den Listen werden jeweils die *IntNodes* und die *AccessPointNodes* gespeichert.

```

3 references
public void AddNode(Node node)
{
    if (node is IntNode intNode)
    {
        intNode.Id = IntNodes.GetNextId();
        IntNodes.Add(intNode);
    }
    else if (node is AccessPointNode apNode)
    {
        apNode.Id = IntNodes.GetNextId();
        ApNodes.Add(apNode);
    }

    node.NodeClicked += NodeClicked;
    node.StartMoving += OnStartNodeMove;
    node.EndMoving += OnEndNodeMove;

    Canvas.SetLeft(node, node.Left - _nodeWidth / 2);
    Canvas.SetTop(node, node.Top - _nodeWidth / 2);

    node.NodeWidth = _nodeWidth;

    NodeCanvas.Children.Add(node);
}

```

*Abbildung 29: Code - AddNode*

Nimmt eine Node und fügt diese entweder den IntNodes oder ApNodes hinzu. Danach werden alle Events registriert und die Node wird dem Canvas hinzugefügt.

```

2 references
private void AddNode(Node node, MouseButtonEventArgs e)
{
    node.Left = eGetPosition(NodeCanvas).X;
    node.Top = eGetPosition(NodeCanvas).Y;

    AddNode(node);
}

```

*Abbildung 30: Code - AddNode*

Nimmt eine Node und zusätzlich noch einen MouseButtonEventArgs. Setzt die Position von der Node zu der des MouseEvent.

```

1 reference
private void RemoveNode(Node node)
{
    NodeCanvas.Children.Remove(node);

    if (node is IntNode intNode)
    {
        NodeCanvas.Children.RemoveConnectionsWithNode(node);
        IntNodes.Remove(intNode);
        intNode.ConnectedNodes.OfType<IntNode>().ToList().ForEach(x => x.ConnectedNodes.Remove(intNode));
    }
    else if (node is AccessPointNode apNode)
    {
        ApNodes.Remove(apNode);
    }
}

```

**Abbildung 31: Code - RemoveNode**

Entfernt eine Node vom Canvas. Wenn es eine IntNode ist, werden zusätzlich noch alle Verbindungen mit dieser Node gelöscht.

```

1 reference
private void NodeClicked(object sender, Node.NodeEventArgs e)
{
    if (Mode == EditModes.Remove)
    {
        RemoveNode(e.EventNode);
        if (LastSelectedNode == e.EventNode) LastSelectedNode = null;
        return;
    }

    if (LastSelectedNode == null)
    {
        LastSelectedNode = e.EventNode;
        return;
    }

    if (LastSelectedNode is IntNode lsNode
        && e.EventNode is IntNode sNode
        && Mode == EditModes.AddConnection)
    {
        AddConnection(lsNode, sNode);
    }

    LastSelectedNode = e.EventNode;
}

```

**Abbildung 32: Code - NodeClicked**

Wenn eine Node geklickt wird und der Mode Remove ist, wird Remove aufgerufen. Wenn davor noch keine Node ausgewählt war, wird diese Node ausgewählt und es wird zurückgegeben. Wenn die zuletzt ausgewählte Node eine IntNode ist, die geklickte Node eine IntNode und der Mode AddConnec-

tion ist wird eine Verbindung, AddConnection kommt aus dem NodeConnection File, zwischen den Nodes hinzugefügt. Als letztes wird die LastSelectedNode auf die geklickte Node gesetzt.

- ImageControl.NodeConnection

```
2 references
private void AddConnection(IntNode node1, IntNode node2)
{
    if (node1.ConnectedNodes.Contains(node2)) return;
    if (node2.ConnectedNodes.Contains(node1)) return;

    node1.ConnectedNodes.Add(node2);
    node2.ConnectedNodes.Add(node1);
    var connection = new NodeConnection { ConnectionNode1 = node1, ConnectionNode2 = node2 };

    node1.AttatchedConnections.Add(connection);
    node2.AttatchedConnections.Add(connection);

    connection.ConnectionClicked += OnConnectionClicked;

    NodeCanvas.DrawConnectionNode(connection, _nodeWidth);
}
```

*Abbildung 33: Code - AddConnection*

Nimmt zwei Nodes und fügt eine Connection zwischen den beiden hinzu. "DrawConnectionNode" ist eine ExtensionMethod.

```
2 references
public void AddConnection(int id1, int id2)
{
    var node1 = IntNodes.Find(x => x.Id == id1)!;
    var node2 = IntNodes.Find(x => x.Id == id2)!;
    AddConnection(node1, node2);
}
```

*Abbildung 34: Code - AddConnection*

Filtert zwei Nodes mit basierend auf den mitgegebenen Ids heraus. Ruft dann die AddConnection-Methode mit diesen zwei Nodes auf.

2 references

```
private void RemoveNodeConnection(NodeConnection e)
{
    var node1 = (e.ConnectionNode1 as IntNode)!;
    var node2 = (e.ConnectionNode2 as IntNode)!;

    node1.ConnectedNodes.Remove(node2);
    node2.ConnectedNodes.Remove(node1);

    node1.AttatchedConnections.Remove(e);
    node2.AttatchedConnections.Remove(e);

    NodeCanvas.Children.Remove(e);
}
```

Abbildung 35: Code - RemoveNodeConnection

Entfernt die übergebene NodeConnection von dem Canvas und entfernt alle Verbindungen zwischen den zwei Nodes, die die NodeConnection verbindet.

1 reference

```
private void OnConnectionClicked(object? sender, NodeConnection.NodeConnectionEventArgs e)
{
    if (Mode == EditModes.Remove)
    {
        RemoveNodeConnection(e.EventNodeConnection);
    }
}
```

Abbildung 36: Code - OnConnectionClicked

Wenn eine NodeConnection geklickt wird und der Mode Remove ist, entfernt es die NodeConnection.

- ImageControl.NodeMove

```
private List<NodeConnection> _removedConnections = [];
```

Abbildung 37: Code - \_removedConnections

Speichert alle NodeConnections, die beim Beginn des Bewegens gelöscht werden und fügt sie später beim Aufhören wieder hinzu.

```

1 reference
private void OnStartNodeMove(object? sender, Node.NodeEventArgs e)
{
    if (e.EventNode is not IntNode intNode) return;
    var list = intNode.AttatchedConnections;

    while (list.Count != 0)
    {
        _removedConnections.Add(list.First());
        RemoveNodeConnection(list.First());
    }
}

```

**Abbildung 38:** Code - *OnStartNodeMove*

Entfernt alle Verbindungen zwischen der bewegten Node und die damit verbundenen Nodes und fügt diese zur `_removedConnections`-Liste hinzu.

```

1 reference
private void OnEndNodeMove(object? sender, Node.NodeEventArgs e)
{
    _removedConnections.ForEach(x => AddConnection(x.ConnectionNode1.Id, x.ConnectionNode2.Id));
    _removedConnections = [];
    e.EventNode.Left = Canvas.GetLeft(e.EventNode);
    e.EventNode.Top = Canvas.GetTop(e.EventNode);
}

```

**Abbildung 39:** Code - *OnEndNodeMove*

Fügt alle gelöschten Connections wieder hinzu und aktualisiert die Koordinaten der bewegten Node.

### 5.3.2.3 Nodes

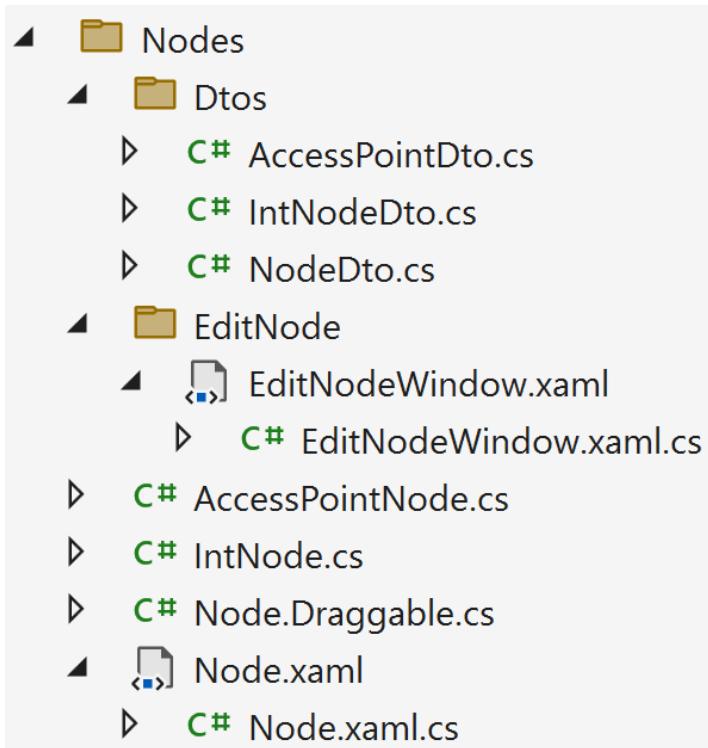


Abbildung 40: Code - Nodes Übersicht

Node ist eine partial Klasse. Die Logik von den einzelnen Funktionen ist in jeweils einem eigenen File.

- Node.xaml

```
<Grid Name="grdNode">
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
    <Ellipse Grid.Row="0" Name="Ellipse" Fill="Green"/>
    <Button Grid.Row="0" Opacity="0" BorderBrush="Transparent"
        Click="OnNodeBtnClicked" Focusable="False" MouseDoubleClick="OnNodeBtnDblClick"/>
    <StackPanel Name="spDescriptions" Grid.Row="1">
        <Label Name="lblNodeName" Content="" VerticalContentAlignment="Center" HorizontalContentAlignment="Center"/>
    </StackPanel>
</Grid>
```

Abbildung 41: Code - Node XAML

```
13 references
public int Id { get; set; }
```

**Abbildung 42: Code - Node Id**

Die Id einer Node.

```
7 references
public double Left { get; set; }
7 references
public double Top { get; set; }
```

**Abbildung 43: Code - NodeCoordinates**

Die Koordinaten einer Node.

```
7 references
public int NodeWidth { get; set; }
```

**Abbildung 44: Code - NodeWidth**

Die Breite einer Node.

```
6 references
public string NodeName { get => lblNodeName.Content.ToString() ?? ""; set => lblNodeName.Content = value; }
```

**Abbildung 45: Code - NodeName**

Der Name einer Node. Get holt den Wert aus dem Label lblNodeName und set setzt den Content des Labels.

```
protected List<string> _propsToEdit = [nameof(NodeName)];
```

**Abbildung 46: Code - \_propsToEdit**

Die Liste aller Namen der Properties die man editieren kann.

```
5 references
public Brush DefaultColor { get; init; } = Brushes.Black;

4 references
public Brush FillColor { get => Ellipse.Fill; set => Ellipse.Fill = value; }
```

*Abbildung 47: Code - DefaultColor/FillColor*

Die Standard-Farbe und die aktuelle Farbe der Node.

```
9 references
public record class NodeEventArgs(Node EventNode);

public EventHandler<NodeEventArgs>? NodeClicked;
```

*Abbildung 48: Code - NodeEventHandler*

Der Event-Handler wird ausgelöst, wenn eine Node geklickt wird.

```
2 references
public Node()
{
    InitializeComponent();
    InitializeDraggable();
}
```

*Abbildung 49: Code - Node ctor*

InitializeDraggable kommt aus dem Draggable-File.

```
1 reference
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    Width = NodeWidth;
    Height = NodeWidth + spDescriptions.Height;

    Ellipse.Width = NodeWidth;
    Ellipse.Height = NodeWidth;
}
```

**Abbildung 50:** Code - *UserControl\_Loaded*

Setzt die Höhe und Breite der UserControl und den Durchmesser der Ellipse.

```
5 references
public virtual void OnNodeBtnClicked(object sender, RoutedEventArgs e) => NodeClicked?.Invoke(this, new NodeEventArgs(this));
```

**Abbildung 51:** Code - *OnNodeBtnClicked*

Löst das Event NodeClicked aus.

```
1 reference
public void EditNode()
{
    var editNodeWindow = new EditNodeWindow(_propsToEdit.ToDictionary(prop => prop, prop => prop.GetType().GetProperties().Single(x => x.Name == prop).GetValue(this)?.ToString()));
    editNodeWindow.ShowDialog();

    if (!editNodeWindow.ApplyChanges) return;

    editNodeWindow.PropValues.ToList().ForEach(prop =>
    {
        GetType().InvokeMember(prop.Key,
            BindingFlags.Instance | BindingFlags.Public | BindingFlags SetProperty,
            Type.DefaultBinder, this, [prop.Value]);
    });
}
```

**Abbildung 52:** Code - *EditNode*

Öffnet ein EditNode-Window mit allen Properties aus \_propsToEdit. Wenn ApplyChanges true ist werden die neuen Werte in die Properties gespeichert.

```
3 references
public virtual void OnNodeBtnDblClick(object sender, MouseButtonEventArgs e)
{
    if (ImageControl.Mode != EditModes.EditNodes) return;
    EditNode();
}
```

**Abbildung 53:** Code - *OnNodeBtnDblClick*

Wird ausgelöst, wenn die Node doppelt geklickt wird. Wenn der Mode EditNodes ist, wird die EditNode-Methode aufgerufen.

```
0 references
public override string ToString() => $"{Id}: {NodeName}";
```

Abbildung 54: Code - Node ToString

ToString der Node Methode, hilfreich für debugging.

- Node.Draggable [7]

```
protected bool _isDragging;
private Point _clickPosition;
```

Abbildung 55: Code - Node Draggable

Variablen für das Bewegen der Node.

```
public EventHandler<NodeEventArgs>? StartMoving;
public EventHandler<NodeEventArgs>? EndMoving;
```

Abbildung 56: Code - EventHandler

Die Events die beim Starten und Beenden vom Bewegen aufgerufen werden.

```
1 reference
public void InitializeDraggable()
{
    MouseDoubleClick += StartDragging;
    PreviewMouseLeftButtonDown += StartDragging;
    PreviewMouseLeftButtonUp += EndDragging;
   MouseMove += MoveDragging;
}
```

Abbildung 57: Code - InitializeDraggable

Alle Events werden zugewiesen.

```
2 references
private void StartDragging(object sender, MouseButtonEventArgs e)
{
    if (ImageControl.Mode != EditModes.MoveNodes) return;

    _isDragging = true;
    var draggableControl = (sender as UserControl)!;
    _clickPosition = e.GetPosition(this);
    draggableControl.CaptureMouse();

    StartMoving?.Invoke(this, new NodeEventArgs(this));
}
```

Abbildung 58: Code - StartDragging

Wenn der Mode nicht MoveNodes ist, wird abgebrochen. Löst das StartMoving-Event aus.

```
1 reference
private void EndDragging(object sender, MouseButtonEventArgs e)
{
    if (ImageControl.Mode != EditModes.MoveNodes) return;

    _isDragging = false;
    var draggable = (sender as UserControl)!;
    draggable.ReleaseMouseCapture();

    EndMoving?.Invoke(this, new NodeEventArgs(this));
}
```

Abbildung 59: Code - EndDragging

Löst das EndMoving-Event aus.

```
1 reference
private void MoveDragging(object sender, MouseEventArgs e)
{
    if (ImageControl.Mode != EditModes.MoveNodes || !_isDragging || sender is not UserControl) return;

    Point currentPosition = e.GetPosition(Parent as UIElement);

    Canvas.SetLeft(this, currentPosition.X - NodeWidth / 2);
    Canvas.SetTop(this, currentPosition.Y - NodeWidth / 2);
}
```

**Abbildung 60:** Code - MoveDragging

Bewegt die Node im NodeCanvas.

- AccessPointNode

Leitet von Node ab.

```
3 references
public string MacAddress { get; set; } = "";
```

**Abbildung 61:** Code - MacAddress

Die Mac-Adresse eines AccessPoints.

```
2 references
public AccessPointNode() : base()
{
    _propsToEdit = [.. _propsToEdit, nameof(MacAddress)];
    DefaultColor = Brushes.Chartreuse;
    FillColor = DefaultColor;
}
```

**Abbildung 62:** Code - APNode

Fügt zu den \_propsToEdit noch die MacAddress-Property hinzu und setzt die Standardfarbe.

- IntNode

Leitet von Node ab.

10 references  
`public List<Node> ConnectedNodes { get; set; } = [];`

*Abbildung 63: Code - ConnectedNodes*

Liste aller verbundenen Nodes.

4 references  
`public List<NodeConnection> AttatchedConnections { get; set; } = [];`

*Abbildung 64: Code - AttatchedConnections*

Liste aller verbundenen Connections.

2 references  
`public IntNode() : base()  
{  
 _propsToEdit = [... _propsToEdit];  
 DefaultColor = Brushes.Black;  
 FillColor = DefaultColor;  
}`

*Abbildung 65: Code - IntNode*

Setzt die Standardfarbe.

- EditNode

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
    <Grid Grid.Row="0" Name="grdPropElements">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
    </Grid>
    <StackPanel Grid.Row="1" Orientation="Horizontal">
        <Button Content="Save" Click="OnSaveClicked"/>
        <Button Content="Cancel" Click="OnCanbleClicked"/>
    </StackPanel>
</Grid>
```

*Abbildung 66: Code - EditNode XAML*

```
private Dictionary<string, string> _properties;
```

*Abbildung 67: Code - \_properties*

Dictionary aller Properties, die mitgegeben worden sind, mit dem Property-Namen als Key und den Werten der Properties als Value.

```
public bool ApplyChanges = false;
public Dictionary<string, string> PropValues = [];
```

*Abbildung 68: Code - ApplyChanges*

ApplyChanges bestimmt ob die Änderungen an den Properties übernommen werden. In PropValues sind die Properties als Key mit den neuen Werten als Value drinnen.

```

1 reference
public EditNodeWindow(Dictionary<string, string> properties)
{
    InitializeComponent();
    _properties = properties;
}

```

**Abbildung 69:** Code – *EditNodeWindow*

```

1 reference
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    PreviewKeyUp += OnPreviewKeyUp;

    _properties.ToList().ForEach(prop =>
    {
        var lblName = new Label { Content = $"{prop.Key}:" };
        var txtValue = new TextBox { Name = prop.Key, Width = 200, Text = prop.Value };

        txtValue.PreviewKeyDown += TxtOnPreviesKeyDown;

        Grid.SetColumn(lblName, 0);
        Grid.SetColumn(txtValue, 1);

        grdPropElements.RowDefinitions.Add(new RowDefinition());
        Grid.SetRow(lblName, grdPropElements.RowDefinitions.Count - 1);
        Grid.SetRow(txtValue, grdPropElements.RowDefinitions.Count - 1);

        grdPropElements.Children.Add(lblName);
        grdPropElements.Children.Add(txtValue);
    });
}

```

**Abbildung 70:** Code - *Window\_Loaded*

Registriert das OnPreviewKeyUp-Event und erstellt für jede Property eine eigene Zeile mit einem Label *lblName* und einer TextBox *txtValue*.

```

1 reference
private void TxtOnPreviesKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Enter)
    {
        int focusRow = Grid.GetRow((UIElement)Keyboard.FocusedElement);

        var nextTextBox = focusRow >= _properties.Count - 1
            ? grdPropElements.Children.OfType<TextBox>().Single(x => Grid.GetRow(x) == 0)
            : grdPropElements.Children.OfType<TextBox>().Single(x => Grid.GetRow(x) == focusRow + 1);
        Keyboard.Focus(nextTextBox);
    }
}

```

**Abbildung 71:** Code - *TxtOnPreviesKeyDown*

Wenn in einer Textbox Enter gedrückt wird, geht der Cursor in die nächste Textbox. Wenn es die unterste Textbox ist, geht der Cursor wieder in die oberste.

```
1 reference
private void CloseWindowWithSave()
{
    ApplyChanges = true;

    _properties.ToList().ForEach(propName =>
    {
        var element = grdPropElements.Children.OfType<TextBox>().Single(x => x.Name == propName.Key);
        PropValues.Add(propName.Key, element.Text);
    });

    Close();
}
```

*Abbildung 72: Code - CloseWindowWithSave*

Setzt ApplyChanges auf true, speichert die Values der geänderten Properties in das PropValue-Dictionary und schließt das Fenster.

```
2 references
private void CloseWindowWithoutSave()
{
    ApplyChanges = false;
    Close();
}
```

*Abbildung 73: Code - CloseWindowWithoutSave*

Schließt das Fenster, ohne zu speichern.

```

1 reference
private void OnSaveClicked(object sender, RoutedEventArgs e)
=> CloseWindowWithSave();

1 reference
private void OnPreviewKeyUp(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Escape)
        CloseWindowWithoutSave();
}

1 reference
private void OnCancelClicked(object sender, RoutedEventArgs e)
=> CloseWindowWithoutSave();

```

*Abbildung 74: Code - OnSave/OnPreviewKeyUp*

Wenn der Button Save geklickt wird, wird CloseWindowWithSave aufgerufen, wenn der Cancel-Knopf geklickt wird oder der Escape-Key wird das CloseWindowWithoutSave aufgerufen.

#### 5.3.2.4 Connection

```

<Line Name="ConnectionLine"
      X1="0"
      X2="{Binding ActualWidth, ElementName=ConnectionUserControll, Mode=OneWay}"
      Y1="0"
      Y2="{Binding ActualHeight, ElementName=ConnectionUserControll, Mode=OneWay}"
      StrokeThickness="3"
      MouseUp="ConnectionLine_MouseUp"/>

```

*Abbildung 75: Code - ConnectionLine XAML*

```

6 references
public Node ConnectionNode1 { get; set; }
6 references
public Node ConnectionNode2 { get; set; }

```

*Abbildung 76: Code - ConnectionNode1/2*

Die beiden Nodes welche die NodeConnection verbindet.

```

private double _x1;
private double _y1;
private double _x2;
private double _y2;

2 references
public double X1
{
    get => _x1;
    set
    {
        _x1 = value;
        UpdateLineDisplay();
    }
}
2 references
public double Y1
{
    get => _y1; set
    {
        _y1 = value;
        UpdateLineDisplay();
    }
}
2 references
public double X2
{
    get => _x2; set
    {
        _x2 = value;
        UpdateLineDisplay();
    }
}
2 references
public double Y2
{
    get => _y2; set
    {
        _y2 = value;
        UpdateLineDisplay();
    }
}

```

**Abbildung 77:** Code - Koordinaten (x1, x2, y1, y2)

Die Koordinaten des Anfangs und Endes der NodeConnection. Wenn ein Value geändert wird, wird UpdateLineDisplay aufgerufen.

```

3 references
public record class NodeConnectionEventArgs(NodeConnection EventNodeConnection);

public EventHandler<NodeConnectionEventArgs> ConnectionClicked;

```

**Abbildung 78:** Code - NodeConnectionEvent

ConnectionClicked wird ausgelöst, wenn eine NodeConnection geklickt wird.

```
1 reference
public Brush Stroke { get => ConnectionLine.Stroke; set => ConnectionLine.Stroke = value; }
```

Abbildung 79: Code - Stroke

Die Farbe der NodeConnection.

```
4 references
private void UpdateLineDisplay()
{
    RenderTransformOrigin = new Point(0, 0);
    var flipTrans = new ScaleTransform();

    double tempWidht = X2 - X1;
    double tempHeight = Y2 - Y1;

    if (tempWidht < 0)
    {
        flipTrans.ScaleX = -1;
    }
    if (tempHeight < 0)
    {
        flipTrans.ScaleY = -1;
    }

    RenderTransform = flipTrans;

    Width = Math.Abs(tempWidht);
    Height = Math.Abs(tempHeight);
}
```

Abbildung 80: Code - UpdateLineDisplay

```
1 reference
private void ConnectionLine_MouseUp(object sender, System.Windows.Input.MouseEventArgs e)
=> ConnectionClicked.Invoke(this, new NodeConnectionEventArgs(this));
```

Abbildung 81: Code - ConnectoinLine\_MouseUp

Löst das ConnectionClicked-Event aus.

### 5.3.2.5 ExtensionMethods

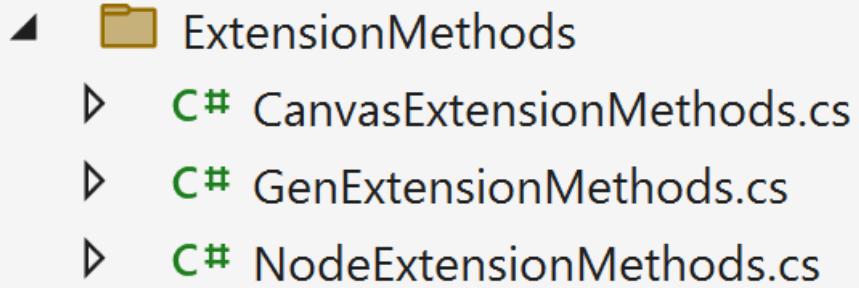


Abbildung 82: Code - ExtensionMethods Übersicht

- CanvasExtensionMethods

```
1 reference
public static void DrawConnectionNode(this Canvas nodeCanvas, NodeConnection connection, double x1, double y1, double x2, double y2, int nodeWidth = 50)
{
    connection.X1 = x1;
    connection.Y1 = y1;
    connection.X2 = x2;
    connection.Y2 = y2;

    connection.Stroke = Brushes.Blue;

    connection.SetValue(Canvas.LeftProperty, x1 + nodeWidth / 2);
    connection.SetValue(Canvas.TopProperty, y1 + nodeWidth / 2);

    nodeCanvas.Children.Insert(0, connection);
}
```

Abbildung 83: Code - CanvasExtentionMethods

Zeichnet die NodeConnection zwischen den Koordinaten.

```
1 reference
public static void DrawConnectionNode(this Canvas nodeCanvas, NodeConnection connection, int nodeWith = 50)
    => DrawConnectionNode(nodeCanvas,
        connection,
        Canvas.GetLeft(connection.ConnectionNode1),
        Canvas.GetTop(connection.ConnectionNode1),
        Canvas.GetLeft(connection.ConnectionNode2),
        Canvas.GetTop(connection.ConnectionNode2),
        nodeWith);
```

Abbildung 84: Code - DrawConnectionNode

Holt sich zuerst die Koordinaten aus dem Canvas und zeichnet dann die NodeConnection.

- GenExtensionMethods

Hier sind ein paar generelle ExtensionMethods.

```
2 references
public static int GetNextId(this IEnumerable<Node> nodes)
=> nodes.Any()
? nodes.Select(x => x.Id).Max() + 1
: 0;
```

Abbildung 85: Code - GetNextId

Holt sich die nächste verfügbare Id aus einer NodeListe. Wenn es noch keine Node gibt, kommt 0 zurück.

```
1 reference
public static WriteableBitmap ToWritableImage(this string base64String)
{
    byte[] byteBuffer = Convert.FromBase64String(base64String);
    using var memoryStream = new MemoryStream(byteBuffer);
    {
        memoryStream.Position = 0;

        var bmpReturn = (Bitmap)System.Drawing.Image.FromStream(memoryStream);

        using var memory = new MemoryStream();
        {
            bmpReturn.Save(memory, ImageFormat.Png);
            memory.Position = 0;

            var bitmapImage = new BitmapImage();
            bitmapImage.BeginInit();
            bitmapImage.StreamSource = memory;
            bitmapImage.CacheOption = BitmapCacheOption.OnLoad;
            bitmapImage.EndInit();
            bitmapImage.Freeze();

            return new WriteableBitmap(bitmapImage);
        }
    }
}
```

Abbildung 86: Code - ToWritableImage

Wandelt einen base64 String zu einer WriteableBitmap um.

```

1 reference
public static void RemoveConnectionsWithNode(this UIElementCollection uieCollection, Node node)
=> uieCollection.OfType<NodeConnection>()
    .Where(x => x.ConnectionNode1 == node || x.ConnectionNode2 == node)
    .ToList()
    .ForEach(uieCollection.Remove);

```

**Abbildung 87: Code - RemoveConnectionsWithNode**

Löscht alle Connections einer Node von einer UIELEMtCollection.

- **NodeExtensionMethods**

```

1 reference
public static IntNodeDto ToIntNodeDto(this IntNode intNode, ImageControl window)
=> new()
{
    Id = intNode.Id,
    Name = intNode.NodeName,
    Left = intNode.Left / window.Image.Width,
    Top = intNode.Top / window.Image.Height,
    ConnectedNodes = intNode.ConnectedNodes.Select(x => x.Id).ToList(),
};

1 reference
public static AccessPointDto ToAccessPointDto(this AccessPointNode apNode, ImageControl window)
=> new()
{
    Id = apNode.Id,
    Name = apNode.NodeName,
    Left = apNode.Left / window.Image.Width,
    Top = apNode.Top / window.Image.Height,
    MacAddress = apNode.MacAddress,
};

```

**Abbildung 88: Code - ToIntNodeDto**

Wandelt eine IntNode oder AccessPointNode zu einem Dto der Klasse um.

```

1 reference
public static List<IntNodeDto> ToNodeDtoList(this IEnumerable<IntNode> intNodes, ImageControl window)
=> intNodes.Select(x => x.ToIntNodeDto(window)).ToList();

1 reference
public static List<AccessPointDto> ToNodeDtoList(this IEnumerable<AccessPointNode> apNodes, ImageControl window)
=> apNodes.Select(x => x.ToAccessPointDto(window)).ToList();

```

**Abbildung 89: Code - ToNodeDtoList**

Wandelt eine Liste von IntNodes oder AccessPoints zu einer Liste von Dtos dieser Klasse um.

```

1 reference
public static IntNodeToIntNode(this IntNodeDto intNodeDto, ImageControl window)
=> new()
{
    Id = intNodeDto.Id,
    NodeName = intNodeDto.Name,
    Left = intNodeDto.Left * window.Image.Width,
    Top = intNodeDto.Top * window.Image.Height,
    ConnectedNodes = [],
};

1 reference
public static AccessPointNode ToApNode(this AccessPointDto apNodeDto, ImageControl window)
=> new()
{
    Id = apNodeDto.Id,
    NodeName = apNodeDto.Name,
    Left = apNodeDto.Left * window.Image.Width,
    Top = apNodeDto.Top * window.Image.Height,
    MacAddress = apNodeDto.MacAddress,
};

```

**Abbildung 90:** Code – *ToIntNode/ToApNode*

Wandelt ein IntNodeDto oder ein AccessPointNodeDto zu einer normalen Klasse um.

```

1 reference
public static List<IntNode> ToNodeList(this IEnumerable<IntNodeDto> intNodes, ImageControl window)
=> intNodes.Select(x => x.ToIntNode(window)).ToList();

1 reference
public static List<AccessPointNode> ToNodeList(this IEnumerable<AccessPointDto> intNodes, ImageControl window)
=> intNodes.Select(x => x.ToApNode(window)).ToList();

```

**Abbildung 91:** Code - *ToNodeList*

Wandelt eine Liste von IntNodeDtos oder AccessPointDtos zu einer Liste von normalen Klassen um.

## 5.4 Backend

### 5.4.1 Nuggets

Folgende Nuggets wurden verwendet:

 <b>CliWrap</b> by Tyrrrz	3.6.6
Library for interacting with external command-line interfaces	
 <b>GrueneisR.RestClientGenerator</b> by Robert Gruenies	1.2.2
Middleware based on Swagger to create a http file for Rest Client using swagger.json	
 <b>MathNet.Spatial</b> by Christoph Ruegg, Johan Larsson	0.6.0
Math.NET Spatial, providing methods and algorithms for geometry computations in science, engineering and every day use. Supports .Net Framework 4.6.1 or higher and .Net Standard 2.0 or higher on Windows, Linux and Mac.	
 <b>Microsoft.AspNetCore.OpenApi</b> by Microsoft	8.0.3
.NET Provides APIs for annotating route handler endpoints in ASP.NET Core with OpenAPI annotations.	
 <b>Microsoft.VisualStudio.Web.CodeGeneration.Design</b> by Microsoft	8.0.0
.NET Code Generation tool for ASP.NET Core. Contains the dotnet-aspnet-codegenerator command used for generating controllers and views.	8.0.2
 <b>Swashbuckle.AspNetCore</b> by Swashbuckle.AspNetCore	6.5.0
Swagger tools for documenting APIs built on ASP.NET Core	

**Abbildung 92: Backend – Nuggets**

## 5.4.2 Klassen/Methoden/Variablen

```
1 reference
public class PositionController : ControllerBase
{
    private readonly MapService _mapService;
    private readonly PositionService _posService;

    0 references
    public PositionController(MapService mapService, PositionService posService)
    {
        _mapService = mapService;
        _posService = posService;
    }

    [HttpGet("spf")]
    0 references
    public List<int> GetShortestPath(int id1, int id2)
    => _posService.GetShortestPath(id1, id2);

    [HttpPost("getPosition")]
    0 references
    public string CalculateCenterPoint([FromBody] Dictionary<string, string> requestBody)
    {
        return _posService.GetPosition(requestBody);
    }
}
```

Abbildung 93: Code - PositionController

Der PositionController übernimmt die Aufgabe, den kürzesten Weg von A nach B zu finden und die ID's jener Nodes zurückzugeben. Ebenfalls bestimmt er aus den MAC-Adressen und Distanzen zu den umliegenden APs die aktuelle Position.

```
3 references
public class PositionService(MapService mapService)
{
    private readonly MapService _mapService = mapService;

    1 reference
    AStarAlgorythm AStarAlgorythm { get; set; } = new();

    1 reference
    public List<int> GetShortestPath(int id1, int id2) ...

    1 reference
    internal string GetPosition(Dictionary<string, string> requestBody) ...
    3 references
    private Point2D[] CalculateCircleIntersections(Point2D center1, double radius1, Point2D center2, double radius2, double factor) ...
    3 references
    public bool AreCirclesEncapsulated(Point2D center1, double radius1, Point2D center2, double radius2) ...
    1 reference
    private Point2D CalculateTriangleCenter(Point2D point1, Point2D point2, Point2D point3) ...
}
```

Abbildung 94: Code – PositionService

- AStarAlgorythm
- GetPosition wird mit dem "RequestBody" aufgerufen
- Der Requestbody ist ein JSON mit MAC-Adressen und Distanzen zu gewissen APs
- triedAPCount wird auf 3 gesetzt
- der Requestbody wird in KeyValueCollection gesplittet
- diese werden in ein Dictionary gespeichert
- Dictionary wurde als Speichermethode gewählt, da manche APs mehrere Signale abgeben, welche gelesen werden, jedoch selbe Position und MAC-Adresse haben
- in eine neue Liste sortedAPs, werden die drei APs gespeichert, welche die geringste Distanz haben
- Die APs vom Endgerät werden mit den am Backend gespeicherten APs, zu einem combinedAccess-Point Objekt verbunden

```

1 reference
public List<SpfNode> FindShortestPath(SpfNode startNode, SpfNode targetNode)
{
    var openSet = new List<SpfNode> { startNode };
    var closedSet = new HashSet<SpfNode>();

    startNode.GScore = 0;
    startNode.FScore = Heuristic(startNode, targetNode);

    while (openSet.Count > 0)
    {
        var currentNode = GetLowestFScoreNode(openSet);
        if (currentNode == targetNode)
        {
            return ReconstructPath(currentNode);
        }
        openSet.Remove(currentNode);
        closedSet.Add(currentNode);

        foreach (var neighbor in currentNode.Neighbors)
        {
            if (closedSet.Contains(neighbor))
            {
                continue;
            }
            var tentativeGScore = currentNode.GScore + 1;

            if (!openSet.Contains(neighbor))
            {
                openSet.Add(neighbor);
            }
            else if (tentativeGScore >= neighbor.GScore)
            {
                continue;
            }
            neighbor.CameFrom = currentNode;
            neighbor.GScore = tentativeGScore;
            neighbor.FScore = neighbor.GScore + Heuristic(neighbor, targetNode);
        }
    }
    return null;
}

```

Abbildung 95: Code – FindShortestPath

Die FindShortestPath Methode wurde mithilfe des A\* Algorythmus programmiert. [9]

```

List<CombinedAccessPoint> combinedAPs = sortedAPs
    .Select(ap =>
{
    var matchingAccessPoint = allAccessPoints.FirstOrDefault(accessPoint => accessPoint.MacAddress == ap.Key);

    return new CombinedAccessPoint
    {
        Id = matchingAccessPoint?.Id ?? 0,
        Left = matchingAccessPoint?.Left ?? 0.0,
        Top = matchingAccessPoint?.Top ?? 0.0,
        MacAddress = ap.Key,
        Distance = ap.Value
    };
})
.ToList();

```

**Abbildung 96:** Code – CombinedAPs

Anhand dieser Daten werden drei Point2D Objekte erstellt. [10]

```

Point2D center1 = new Point2D(combinedAPs[0].Left, combinedAPs[0].Top);
double radius1 = combinedAPs[0].Distance;

Point2D center2 = new Point2D(combinedAPs[1].Left, combinedAPs[1].Top);
double radius2 = combinedAPs[1].Distance;

Point2D center3 = new Point2D(combinedAPs[2].Left, combinedAPs[2].Top);
double radius3 = combinedAPs[2].Distance;

```

**Abbildung 97:** Code - Point2D

Wenn nicht genügend APs in der Nähe sind, wird eine Exception geworfen.

Dann wird überprüft, ob sich die Kreise ineinander befinden, falls ja, wird einer davon durch den nächstgelegenen AP ersetzt.

```

while (circlesChanged)
{
    circlesChanged = false;
    if (triedAPCount - 1 > combinedAPs.Count)
    {
        throw new InvalidOperationException("Position can not be calculated");
    }

    if (AreCirclesEncapsulated(center1, radius1, center2, radius2))
    {
        center1 = new Point2D(combinedAPs[triedAPCount].Left, combinedAPs[triedAPCount].Top);
        radius1 = combinedAPs[0].Distance;
        triedAPCount++;
        circlesChanged = true;
    }

    if (AreCirclesEncapsulated(center3, radius3, center2, radius2))
    {
        center2 = new Point2D(combinedAPs[triedAPCount].Left, combinedAPs[triedAPCount].Top);
        radius2 = combinedAPs[0].Distance;
        triedAPCount++;
        circlesChanged = true;
    }

    if (AreCirclesEncapsulated(center1, radius1, center3, radius3))
    {
        center3 = new Point2D(combinedAPs[triedAPCount].Left, combinedAPs[triedAPCount].Top);
        radius3 = combinedAPs[0].Distance;
        triedAPCount++;
        circlesChanged = true;
    }
}

```

*Abbildung 98: Code - circlesChanged*

```

3 references
public bool AreCirclesEncapsulated(Point2D center1, double radius1, Point2D center2, double radius2)
{
    double distanceBetweenCenters = Math.Sqrt(Math.Pow(center2.X - center1.X, 2) + Math.Pow(center2.Y - center1.Y, 2));
    bool isCircle1Encapsulated = distanceBetweenCenters + radius1 <= radius2;
    bool isCircle2Encapsulated = distanceBetweenCenters + radius2 <= radius1;
    return isCircle1Encapsulated || isCircle2Encapsulated;
}

```

*Abbildung 99: Code – AreCirclesEncapsulated*

Falls die Kreise zu klein sind, sodass sie keine Schnittpunkte haben, werden sie vergrößert.

```

double factor = 1;
double distance12 = center1.DistanceTo(center2);
double distance23 = center2.DistanceTo(center3);
double distance13 = center1.DistanceTo(center3);

if (radius1 <= 0) radius1 = 0.1;
if (radius2 <= 0) radius2 = 0.1;
if (radius3 <= 0) radius3 = 0.1;

while ((distance12 > radius1 + radius2) || (distance13 > radius2 + radius3) || (distance23 > radius1 + radius3))
{
    factor *= 1.1;
    radius1 *= factor;
    radius2 *= factor;
    radius3 *= factor;
}

```

**Abbildung 100:** Code - radius\*factor

Die Schnittpunkte der Kreise werden berechnet. Zwischen diesen Punkten wird ein Dreieck aufgespannt, von dem der Mittelpunkt berechnet wird.

```

Point2D[] intersectionPoints12 = CalculateCircleIntersections(center1, radius1, center2, radius2, factor);
Point2D[] intersectionPoints23 = CalculateCircleIntersections(center2, radius2, center3, radius3, factor);
Point2D[] intersectionPoints13 = CalculateCircleIntersections(center1, radius1, center3, radius3, factor);

Point2D triangleCenter = CalculateTriangleCenter(intersectionPoints12[0], intersectionPoints23[0], intersectionPoints13[1]);

return triangleCenter.ToString();

```

**Abbildung 101:** Code - CalculateCircleIntersections

```

private Point2D[] CalculateCircleIntersections(Point2D center1, double radius1, Point2D center2, double radius2, double factor)
{
    double distance = center1.DistanceTo(center2);

    double a = (radius1 * radius1 - radius2 * radius2 + distance * distance) / (2 * distance);

    double h = radius1 * radius1 - (a * a);
    if (h < 0) h = h * -1;
    h = Math.Sqrt(h);
    Point2D p = center1 + a * (center2 - center1) / distance;

    double x3 = p.X + h * (center2.Y - center1.Y) / distance;
    double y3 = p.Y - h * (center2.X - center1.X) / distance;
    double x4 = p.X - h * (center2.Y - center1.Y) / distance;
    double y4 = p.Y + h * (center2.X - center1.X) / distance;

    return new Point2D[] { new Point2D(x3, y3), new Point2D(x4, y4) };
}

```

**Abbildung 102:** Code – CalculateCircleIntersections

```

private Point2D CalculateTriangleCenter(Point2D point1, Point2D point2, Point2D point3)
{
    double centerX = (point1.X + point2.X + point3.X) / 3;
    double centerY = (point1.Y + point2.Y + point3.Y) / 3;

    return new Point2D(centerX, centerY);
}

```

**Abbildung 103:** Code - CalculateTriangleCenter

- MapService

```
private string ContentPath = "Content\\\";
```

*Abbildung 104: Code - ContentPath*

Der Pfad in dem der Content liegt.

```
private List<IntNodeDto> _intNodes = [];

4 references
public List<IntNodeDto> IntNodes
{
    get => _intNodes;
    set
    {
        _intNodes = value;
        File.WriteAllText(@$"{{ContentPath}}\intnodes.json", JsonSerializer.Serialize(IntNodes));
    }
}
```

*Abbildung 105: Code – intNodes*

Beinhaltet und speichert beim Setzen die im Karteneditor definierten IntNodes.

```
2 references
public List<IntNodeDto> TestIntNodes { get; set; } = [];
```

*Abbildung 106: Code - TestIntNodes*

Speichert IntNodes, die zum Testen verwendet werden.

```
private List<AccessPointDto> _accessPoints = [];
3 references
public List<AccessPointDto> AccessPoints
{
    get => _accessPoints;
    set
    {
        _accessPoints = value;
        File.WriteAllText(@$"{{ContentPath}}\apnodes.json", JsonSerializer.Serialize(AccessPoints));
    }
}
```

*Abbildung 107: Code - FullProp AccessPoints*

Beinhaltet und speichert beim Setzen die im Karteneditor definierten AccessPoints.

```
private string _imgBase64 = "";
4 references
public string ImgBase64
{
    get => _imgBase64;
    set
    {
        _imgBase64 = value;
        File.WriteAllText(@$"{{ContentPath}}\base64", ImgBase64);
    }
}
```

Abbildung 108: Code - FullProp ImgBase64

Beinhaltet und speichert beim Setzen die als Base64 codierte Grafik.

```
0 references
public MapService()
{
    InitializeTestData();
}
```

Abbildung 109: Code - MapService ctor

1 reference

```
private void InitializeTestData()
{
    var intNode1 = new IntNodeDto() { Id = 0, Top = 0.1, Left = 0.1, };
    var intNode2 = new IntNodeDto() { Id = 1, Top = 0.3, Left = 0.2, };
    var intNode3 = new IntNodeDto() { Id = 2, Top = 0.5, Left = 0.7, };
    var intNode4 = new IntNodeDto() { Id = 3, Top = 0.9, Left = 0.9, };
    var intNode5 = new IntNodeDto() { Id = 4, Top = 0.7, Left = 0.5, };
    var intNode6 = new IntNodeDto() { Id = 5, Top = 0.2, Left = 0.8, };
    var intNode7 = new IntNodeDto() { Id = 6, Top = 0.4, Left = 0.3, };
    var intNode8 = new IntNodeDto() { Id = 7, Top = 0.6, Left = 0.6, };
    var intNode9 = new IntNodeDto() { Id = 8, Top = 0.2, Left = 0.1, };
    var intNode10 = new IntNodeDto() { Id = 9, Top = 0.4, Left = 0.4, };
    var intNode11 = new IntNodeDto() { Id = 10, Top = 0.6, Left = 0.7, };

    intNode1.ConnectedNodes.Add(intNode2.Id);
    intNode2.ConnectedNodes.Add(intNode1.Id);
    intNode2.ConnectedNodes.Add(intNode3.Id);
    intNode2.ConnectedNodes.Add(intNode6.Id);
    intNode3.ConnectedNodes.Add(intNode2.Id);
    intNode3.ConnectedNodes.Add(intNode4.Id);
    intNode4.ConnectedNodes.Add(intNode3.Id);
    intNode4.ConnectedNodes.Add(intNode9.Id);
    intNode4.ConnectedNodes.Add(intNode10.Id);
    intNode5.ConnectedNodes.Add(intNode6.Id);
    intNode6.ConnectedNodes.Add(intNode5.Id);
    intNode6.ConnectedNodes.Add(intNode7.Id);
    intNode6.ConnectedNodes.Add(intNode2.Id);
    intNode7.ConnectedNodes.Add(intNode6.Id);
    intNode8.ConnectedNodes.Add(intNode10.Id);
    intNode9.ConnectedNodes.Add(intNode4.Id);
    intNode10.ConnectedNodes.Add(intNode8.Id);
    intNode10.ConnectedNodes.Add(intNode11.Id);
    intNode11.ConnectedNodes.Add(intNode10.Id);

    TestIntNodes = [
        intNode1,
        intNode2,
        intNode3,
        intNode4,
        intNode5,
        intNode6,
        intNode7,
        intNode8,
        intNode9,
        intNode10,
        intNode11,
    ];
}
```

Abbildung 110: Code - InitializeTestData

Initialisiert die TestDaten.

```

1 reference
public List<SpfNode> GetNodesAsSpfList(List<IntNodeDto> nodeList)
{
    var spfNodes = nodeList.Select(x => new SpfNode
    {
        Id = x.Id,
        X = x.Left,
        Y = x.Top,
    }).ToList();

    spfNodes.ForEach(spfNode =>
    {
        var connectedNodes = spfNodes.Where(node => nodeList.Find(x => x.Id == spfNode.Id).ConnectedNodes.Contains(node.Id)).ToList();
        spfNode.Neighbors = connectedNodes;
    });
    return spfNodes;
}

```

**Abbildung 111:** Code - *GetNodesAsSpfList*

Wandelt die IntNodeDto-Liste in eine SpfNode-Liste um.

- ImageCrlController

```

[HttpGet("nodes")]
0 references
public List<IntNodeDto> GetNodes() => _mapService.IntNodes.ToList();

```

**Abbildung 112:** Code - *ImageCrlController*

Gibt die IntNodes-Liste zurück.

```

[HttpPost("mapData")]
0 references
public IActionResult PostMapData([FromBody]string jsonMapData)
{
    this.Log();
    var mapData = (MapDataDto)JsonSerializer.Deserialize(jsonMapData, typeof(MapDataDto))!;

    this.Log();
    Console.WriteLine(mapDataToJson());
    _mapService.IntNodes = mapData.IntNodes;
    _mapService.AccessPoints = mapData.AccessPoints;

    return Ok(jsonMapData);
}

```

**Abbildung 113:** Code - *PostMapData*

Speichert die im Karteneditor definierten Nodes.

```
[HttpGet("img")]
0 references
public string GetImg() => _mapService.ImgBase64;
```

Abbildung 114: Code - GetImg

Stellt die Base64-Grafik zur Verfügung.

- LoadContentBackgroundService

```
private readonly IServiceProvider _serviceProvider = serviceProvider;

0 references
protected override Task ExecuteAsync(CancellationToken stoppingToken)
{
    using var scope = _serviceProvider.CreateScope();
    var mapService = scope.ServiceProvider.GetRequiredService<MapService>();

    string contentPath = "Content\\\";

    List<IntNodeDto> intNodeDtos = JsonSerializer.Deserialize<List<IntNodeDto>>(File.ReadAllText(${contentPath}intnodes.json));
    List<AccessPointDto> apDtos = JsonSerializer.Deserialize<List<AccessPointDto>>(File.ReadAllText(${contentPath}apnodes.json));
    string base64 = File.ReadAllText(${contentPath}\\base64");

    mapService.ImgBase64 = base64;
    mapService.IntNodes = intNodeDtos;
    mapService.AccessPoints = apDtos;

    return Task.Run(() => { }, stoppingToken);
}
```

Abbildung 115: Code - ExecuteAsync

Lädt den am Server gespeicherten Content ein.

## 5.5 Android-App

```
5 usages
public class AccessPoint {
    2 usages
    private String macAddress;
    2 usages
    private float left;
    2 usages
    private float top;

    no usages
    public AccessPoint(String macAddress, float left, float top) {
        this.macAddress = macAddress;
        this.left = left;
        this.top = top;
    }

    no usages
    public String getMacAddress() { return macAddress; }

    1 usage
    public float getLeft() { return left; }

    1 usage
    public float getTop() { return top; }
}
```

Abbildung 116: Code – AccessPoint

Die AccessPoint Klasse hat drei Variable, "macAddress" um "AccessPoint" eindeutig zu identifizieren, "left" und "top" sind die relative Position des APs.

### 5.5.1 Base64Handler

```
2 usages
public class Base64Handler {
    1 usage
    private static final String FILE_NAME = "ImageBase64.txt";
    2 usages
    private static final int SCALE_FACTOR = 3;

    1 usage
}    public static String readBase64FromFile(Context context) {...}

    1 usage
}    public static Bitmap decodeBase64(String base64String) {...}
}
```

Abbildung 117: Code - Base64Handler

Der Base64Handler ist dafür zuständig, aus einer Base64-Datei ein Bild zu erstellen. Die Variable FILE\_NAME verweist dabei auf ein Asset (ersichtlich Screenshot).

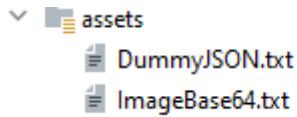


Abbildung 118: Code – assets

Der SCALE\_FACTOR wird benötigt, um das Bild auch ersichtlich zu machen, da es sonst viel zu klein wäre.

Die Methode readBase64FromFile liest den Base64 String von dem File aus den "assets" aus.

```
1 usage
public static String readBase64FromFile(Context context) {
    StringBuilder stringBuilder = new StringBuilder();
    try {
        AssetManager assetManager = context.getAssets();
        InputStream inputStream = assetManager.open(FILE_NAME);
        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
        String line;
        while ((line = reader.readLine()) != null) {
            stringBuilder.append(line).append("\n");
        }
        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return stringBuilder.toString();
}
```

Abbildung 119: Code - `readBase64FromFile`

Die Methode `decodeBase64` macht aus einem Base64 String eine Bitmap.

```
1 usage
public static Bitmap decodeBase64(String base64String) {
    byte[] decodedBytes = Base64.decode(base64String, Base64.DEFAULT);
    Bitmap originalBitmap = BitmapFactory.decodeByteArray(decodedBytes, offset: 0, decodedBytes.length);

    int newWidth = originalBitmap.getWidth() * SCALE_FACTOR;
    int newHeight = originalBitmap.getHeight() * SCALE_FACTOR;
    Bitmap scaledBitmap = Bitmap.createScaledBitmap(originalBitmap, newWidth, newHeight, filter: true);

    return scaledBitmap;
}
```

Abbildung 120: Code - `decodeBase64`

## 5.5.2 MainActivity

Variablen:

```
public class MainActivity extends AppCompatActivity {  
    2 usages  
    private Timer timer;  
    8 usages  
    private List<NodeData> nodes;  
    3 usages  
    private List<AccessPoint> accessPoints;  
    5 usages  
    private Bitmap mapImage;  
    5 usages  
    private int screenWidth;  
    8 usages  
    NavigationView customView;  
    5 usages  
    private int screenHeight;  
    2 usages  
    private OkHttpClient client;  
    5 usages  
    String baseUrl = "https://8f1c-77-242-69-32.ngrok-free.app";  
    5 usages  
    private WifiManager wifiManager;  
    1 usage  
    private boolean test = true;  
    2 usages  
    private Context context;  
    3 usages  
    private Spinner RoomSpinner;  
    5 usages  
    private NodeData selectedNode;
```

**Abbildung 121: Code - MainActivity Variablen**

Bei der Variable baseUrl muss beachtet werden, dass sie jedes Mal erneuert werden muss, wenn Ngrok neu gestartet wird.

Methoden:

```
@Override
protected void onCreate(Bundle savedInstanceState) {...}
1 usage
private void populateSpinner() {...}
1 usage
private void startPeriodicScan() {...}
1 usage
private void scanForAccessPoints() {...}
1 usage
private void callEndpoint(String parameters) {...}
1 usage
private double calculateDistance(int frequency, int rssi) {...}
1 usage
private class DownloadDataTask extends AsyncTask<String, Void, List<String>> {...}
1 usage
private String downloadJsonData(String jsonUrl) {...}
2 usages
private void handleJsonData(String nodejsonData, String accessPointjsonData, String imageBase64Data) {...}
1 usage
private Bitmap decodeBase64Image(String base64Image) {...}
1 usage
private List<NodeData> parseNodeData(String jsonData) throws JSONException {...}
1 usage
private int getScreenWidth() {...}
1 usage
private int getScreenHeight() {...}
1 usage
private List<AccessPoint> parseAccessPointData(String accessPointjsonData) throws JSONException {...}
1 usage
private class AsyncTaskForBackendAccess extends AsyncTask<Void, Void, String> {...}
```

Abbildung 122: Code - MainActivity Methoden

#### 5.5.2.1 OnCreate

Initialisiert Variablen, startet den Scan für naheliegende APs, holt sich Daten vom Backend

#### 5.5.2.2 startPeriodicScan

Startet nach zehn Sekunden mit einem zehn Sekunden Intervall; es wird periodisch nach umliegenden APs gesucht [10]

#### 5.5.2.3 scanForAccessPoints

Ersucht den User um Zugriff auf WLAN-Daten und Position und erzeugt einen JSON-String daraus, dieser wird als Parameter an callEndpoint übergeben.

#### 5.5.2.4 callEndpoint

```

private void callEndpoint(String parameters) {
    Map<String, String> data = new HashMap<>();
    data.put("parameters", parameters);

    new AsyncTaskForBackendAccess(context, data).execute();
}

```

**Abbildung 123:** Code - `callEndpoint`

#### 5.5.2.5 calculateDistance

```

private double calculateDistance(int frequency, int rssi) {
    double referenceSignalLevel = -45.0;
    double propagationExponent = 2.7;
    double distance = Math.pow(10, (referenceSignalLevel - rssi) / (10 * propagationExponent));

    distance = Math.sqrt(distance * distance - 1);
    double frequencyAdjustment = 27.55 - (20 * Math.log10(frequency));
    distance = distance * Math.pow(10, (frequencyAdjustment - referenceSignalLevel) / (20 * propagationExponent));

    return distance;
}

```

**Abbildung 124:** Code – `calculateDistance`

Berechnet die mit einer vordefinierten Formel die Distanz zu einem AP (die Formel ist für eine bestimmte Marke von APs vordefiniert), jedoch muss beachtet werden, dass diese selten akkurat ist, da Wände oder ähnliche Störfaktoren das Ergebnis verfälschen.

#### 5.5.2.6 downloadJsonData

Schickt einen get-Request zu dem entsprechenden Endpunkt.

#### 5.5.2.7 handleJsonData

Teilt die JSON-Daten, den entsprechenden Listen in der `customView` zu.

#### 5.5.2.8 decodeBase64Image, parseNodeData, parseAccessPointData

Wandeln JSON in verwendbare Daten um.

#### 5.5.2.9 DownloadDataTask

Get-Request an das Backend.

#### 5.5.2.10 AsyncTaskForBackendAccess

Post-Request an das Backend und Weiterverarbeitung der Daten.

```
private NodeData setNearestNode(float x, float y) {
    NodeData nearestNode = null;
    double minDistance = Double.MAX_VALUE;

    for (NodeData node : nodes) {
        double nodeX = node.getRelativeX() / screenWidth;
        double nodeY = node.getRelativeY() / screenHeight;

        double distance = Math.sqrt(Math.pow(nodeX - x, 2) + Math.pow(nodeY - y, 2));
        if (distance < minDistance) {
            minDistance = distance;
            nearestNode = node;
        }
    }
    return nearestNode;
}
```

Abbildung 125: Code – setNearestNode

```
@Override
protected void onPostExecute(String s) {
    Log.d(TAG, msg: "Response: " + s);
    s = s.replaceAll(regex: "[^\\d.,]", replacement: "");
    Toast.makeText(mContext, text: "POST request completed", Toast.LENGTH_SHORT).show();
    String[] coordinates = s.split(regex: ",");

    if (coordinates.length == 2) {
        float x = Float.parseFloat(coordinates[0].trim()) * screenWidth;
        float y = Float.parseFloat(coordinates[1].trim()) * screenHeight;

        customView = findViewById(R.id.customView);
        customView.setorangeCirclePosition(x, y);
        customView.setPathBetweenLines(nodes, mPath);
    }
}
```

Abbildung 126: Code – onPostExecute

### 5.5.3 NavigationView

Variablen:

```
4 usages
public class NavigationView extends View {
    8 usages
    private List<NodeData> nodes;
    3 usages
    private List<AccessPoint> accessPoints;
    3 usages
    private Bitmap mapImage;
    1 usage
    private float touchX;
    1 usage
    private float touchY;
    4 usages
    private float orangeCircleX = -1;
    4 usages
    private float orangeCircleY = -1;
    2 usages
    private ScaleGestureDetector scaleGestureDetector;
    2 usages
    private GestureDetector gestureDetector;
    4 usages
    private Matrix matrix;
    7 usages
    private float scaleFactor = 1.0f;
    3 usages
    private List<NodeData> nodesSPF;
    6 usages
    private List<Integer> spfPath;
    1 usage
    private NodeData selectedNode;
```

Abbildung 127: Code - NavigationView Variablen

## Methoden:

```
no usages
} public NavigationView(Context context) {...}
1 usage
} private void init() {...}
1 usage
} public void setNodes(List<NodeData> nodes) {...}
1 usage
} public void setAccessPoints(List<AccessPoint> accessPoints) {...}
2 usages
} public void setMapImage(Bitmap mapImage) {...}
@Override
} protected void onDraw(Canvas canvas) {...}
@Override
} public boolean onTouchEvent(MotionEvent event) {...}
2 usages
} private void drawRedCircle(Canvas canvas, float x, float y) {...}
1 usage
} private void drawNode(Canvas canvas, NodeData nodeData) {...}
1 usage
} private void drawEdges(Canvas canvas, NodeData nodeData) {...}
1 usage
} private NodeData findNodeById(int nodeId) {...}
2 usages
} public void setSelectedNode(NodeData pselectedNode) { this.selectedNode = pselectedNode; }
1 usage
} public void setPathBetweenLines(List<NodeData> pnodes, List<Integer> mPath) {...}
1 usage
} public void drawPathBetweenLines(Canvas canvas) {...}
1 usage
} private class GestureListener extends GestureDetector.SimpleOnGestureListener{...}
2 usages
} public void setorangeCirclePosition(float x, float y) {...}
1 usage
} private void draworangeCircle(Canvas canvas) {...}
1 usage
} private class ScaleListener extends ScaleGestureDetector.SimpleOnScaleGestureListener {...}
1 usage
} private void drawLineToNearestNode(Canvas canvas) {...}
```

**Abbildung 128: Code - NavigationView Methoden**

### 5.5.3.1 init

Initialisiert scaleGestureDetector, gesturDetector und matrix.

### 5.5.3.2 setNodes, setAccessPoint, setMapImage, setSelectedNode, setPathBetweenLines

#### SetMethoden

### 5.5.3.3 onDraw

Ruft alle Draw Methoden auf und zeichnet den Canvas neu.

#### 5.5.3.4    onTouchEvent

Ruft das onTouchEvent von scaleGestureDetector und gestureDetector auf [10]

#### 5.5.3.5    drawRedCircle, drawNode, DrawEdges, drawPathBetweenLines, drawOrangeCircle, drawLineToNearestNode

Zeichenmethoden.

#### 5.5.3.6    setOrangeCirclePosition

Positioniert den orangenen Kreis (aktueller Standpunkt).

#### 5.5.3.7    GestureListener

Erlaubt das Bewegen der Karte durch Scrollen.

#### 5.5.3.8    ScaleListener

Erlaubt das Zoomen der Karte, durch das Bewegen zweier Finger.

## 5.5.4 NodeData

```
28 usages
public class NodeData {
    2 usages
    private int id;
    2 usages
    private double relativeX;
    2 usages
    private double relativeY;
    2 usages
    private List<Integer> connectedNodes;
    2 usages
    private String name;

    1 usage
}   public NodeData(int id, double relativeX, double relativeY, List<Integer> connectedNodes, String name) {...}
    1 usage
    public String getName(){return name;}

}   public int getId() { return id; }

    7 usages
}   public double getRelativeX() { return relativeX; }

    7 usages
}   public double getRelativeY() { return relativeY; }

    1 usage
}   public List<Integer> getConnectedNodes() { return connectedNodes; }
}
```

Abbildung 129: Code – NodeData

## 6 Ergebnis

### 6.1 Karten-Editor

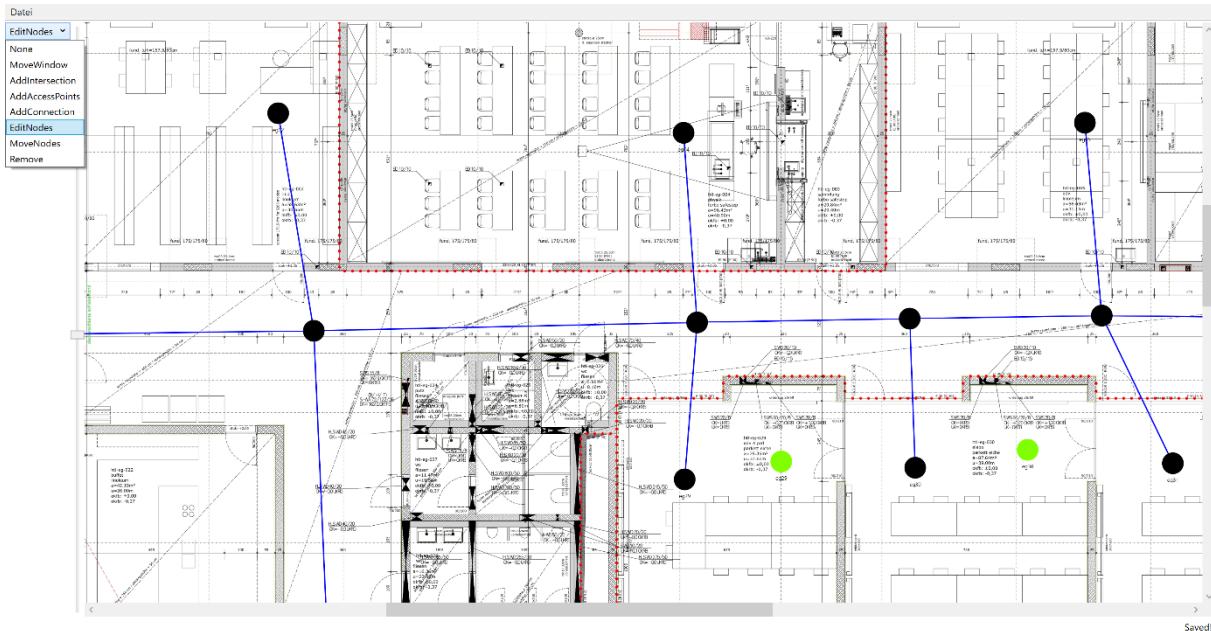


Abbildung 130: Karten-Editor

In dem Screenshot des Karteneditors haben wir versucht, mit einem Blick klarzumachen, was unsere Applikation kann. Man kann nach Belieben eine Grafik öffnen und sie bearbeiten.

Weiters sieht man links welche Funktionen unsere Karteneditor Applikation genau bietet:

- None: Sie können nichts außer scrollen.
- MoveWindow: Sie können die Grafik bewegen.
- AddIntersection: Sie können Knotenpunkte hinzufügen.
- AddAccessPoints: Sie können APs hinzufügen.
- AddConnection: Sie können eine Verbindung hinzufügen.
- EditNodes: Sie können APs und Knotenpunkte bearbeiten.
- MoveNodes: Sie können APs und Knotenpunkte verschieben.
- Remove: Sie können jegliches Item löschen.

## 6.2 Backend

Unser Backend übernimmt erfolgreich die Aufgabe der Kommunikation zwischen all unseren Bestandteilen. Die Karten-Daten von der Karteneditor-App können an das Backend geschickt werden. Dort werden sie verarbeitet und der Android-App brauchbar zur Verfügung gestellt. Weiters übernimmt es erfolgreich die Berechnungen für den SPF und die Positionsbestimmung, welches viele Ressourcen auf dem Endgerät unserer Nutzer erspart. Es speichert den Base64 String, der zur Grafik-Übertragung verwendet wird, und die Koordinaten der Nodes und APs.

## 6.3 Android-App

Unsere Android-App greift auf unser Backend zu und holt sich die Positionen der Nodes und holt sich eine Grafik (den Gebäudeplan) als Base64 String. Mit diesen Ressourcen kann die Android-App die Nodes auf der Karte einzeichnen und entsprechend verbinden. Dann kann durch Eingabe eines Zielraumes die Navigation gestartet werden. Die aktuelle Position des Benutzers wird auf der Karte gesetzt. Durch einen SPF-Algorithmus im Backend wird der kürzeste Weg von dem aktuellen Standort zu dem Zielraum berechnet. Dann wird durch visuelle Anzeige navigiert.

Die Android App besteht, wie im Screenshot ersichtlich aus folgenden Klassen:

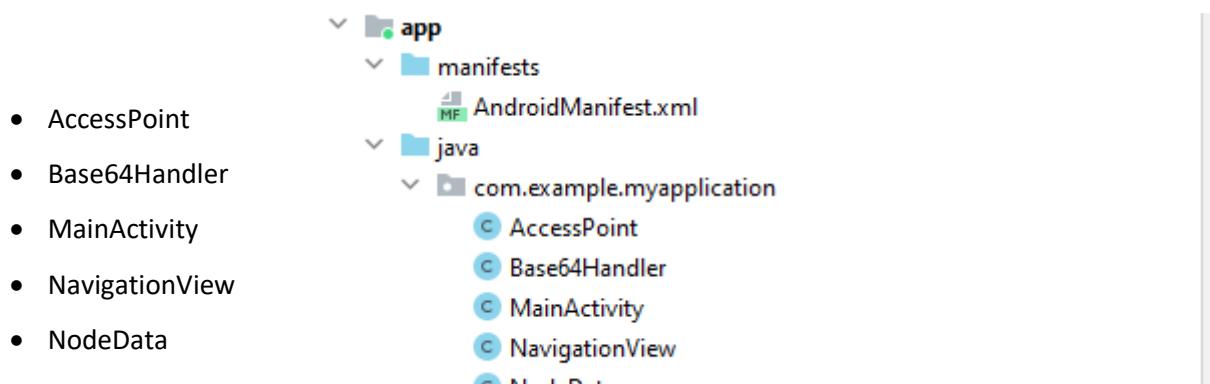


Abbildung 131: Code - app Übersicht

## 6.4 Ablauf/Funktion

### 6.4.1 Karteneditor Ablauf

- Programm öffnen
- Grafik öffnen
- Alle vorhandenen Knotenpunkte und APs werden geladen

- Bearbeitung der Grafik (Knotenpunkte und APs hinzufügen, entfernen, ändern, etc.)
- Speichern/Programm schließen
- (An den Server schicken ist für Testzwecke vorerst ausgeschaltet)

#### **6.4.2 Backend Funktionen**

- Kommunikation mit Karteneditor/Android App
- Speicherung der relevanten Variablen (base64 String für Grafik und Koordinaten Knotenpunkte, APs)
- Berechnung SPF und Position

#### **6.4.3 Android App Ablauf**

- Die App wird gestartet und initialisiert
- Die Daten von den Wegpunkten und des Bildes werden vom Backend geholt
- Die Wegpunkte werden eingezeichnet und verbunden
- Ein Timer wird gestartet, der periodisch nach umliegenden APs scannt
- Nach dem Scan werden die gefundenen MAC-Adressen und Distanzen ans Backend geschickt
- Das Backend berechnet den Standort und schickt diesen zurück.
- Das mobile Endgerät, ruft das Backend erneut auf, um den Kürzesten Weg zum Ziel zu erfragen
- Sobald eine Antwort kommt, wird dieser Weg auf der Karte eingezeichnet

### **6.5 Conclusio**

Die Entwicklung und Implementierung eines Machbarkeitsprototypen, im Kontext eines Proof of Concepts, für eine Indoor-Navigations-App stellt einen bedeutenden Schritt in der Realisierung einer Lösung für die Navigation in geschlossenen Räumen für Fill dar. Durch die Kombination von Android-App, Backend und Karteneditor konnten wir einen funktionsfähigen Prototypen erstellen.

Unsere Ergebnisse zeigen einen klaren Ablauf: Die App wird erfolgreich gestartet und initialisiert, wobei die erforderlichen Daten von den Wegpunkten und dem Kartenbild vom Backend bezogen werden. Die Wegpunkte können vom User präzise eingezeichnet und miteinander verbunden werden, um eine zuverlässige Routenführung zu gewährleisten. Ein Timer steuert den regelmäßigen Scan nach umliegenden Access Points, deren Informationen anschließend an das Backend gesendet wer-

den. Dort erfolgt die Berechnung des Standorts, der dann zurück an das mobile Endgerät gesendet wird.

Des Weiteren ermöglicht die App die Abfrage des kürzesten Weges zum Ziel, indem sie erneut mit dem Backend interagiert. Sobald die Antwort vorliegt, wird der optimale Weg auf der Karte visualisiert, was eine effektive Navigation unterstützt.

Insgesamt unterstreichen unsere Ergebnisse die Machbarkeit einer Indoor-Navigationslösung und zeigen das Potenzial für weitere Entwicklungen und Anwendungen in diesem Bereich auf.

## 7 Resümee

### 7.1 Probleme/Problemlösungsstrategien

In diesem Abschnitt werden wir die Probleme analysieren, auf die wir bei der Planung und Entwicklung der Diplomarbeit gestoßen sind und die Lösungen, die wir erarbeitet oder gefunden haben.

Problem	Lösung
Kommunikation zwischen Backend am Rechner und Android-App am Handy	Verwendung von Ngrok
Bei Prototypen wir viel herumprobiert. Dadurch entsteht oft ein sehr unübersichtlicher/unverständlicher Code.	Neuaufsetzung mit erfahrenem Wissen
Unerfahrenheit in Grafikverarbeitung in C# und Android	Viel einlesen und recherchieren
Ngrok Probleme:  405 not allowed  307 forwarded	405: falscher Datentyp übergeben  307: app.UseHttpsRedirection(); auskommennieren
GET requests auf unser Backend kamen nicht an; Frameworks funktionierten nicht wie geplant	Hierbei haben uns die Folien von Herr Prof. Sickinger (3.JG) sehr weitergeholfen. (anbei bei Digitaler Abgabe) [13]

Android lässt nur 3 Scans alle 2 Minuten zu	Wi-Fi scan throttling deaktivieren (in den Entwicklereinstellungen)
Bildübertragung	Workaround 1; Base64 String verschicken, ist aber leider sehr unperfomant  Workaround 2; hardcoded, da Machbarkeitsprototyp

## 7.2 Lessons Learned

### 7.2.1 Michael Aigner

Ich habe gelernt, dass es oft von Vorteil ist, nach Problemen zu "googlen". Es nimmt oft deutlich mehr Zeit in Anspruch, wenn man versucht gewisse Probleme selbst zu lösen. Weiters habe ich mir mitgenommen, dass strukturierter Code, vor allem bei langlebigen Projekten, sehr wichtig ist. Außerdem habe ich gelernt, dass es durchaus sinnvoll ist, ein neues Feature in einem Sandbox-Projekt auszutesten, bevor man es in die Main-Anwendung implementiert.

### 7.2.2 Sebastian Raith

Ich habe mir mitgenommen, alten Code nicht gleich über Bord zu werfen, selbst wenn er nicht funktioniert. Man weiß nie, ob man ihn nicht doch noch einmal einsehen will. Weiters habe ich gelernt, dass Demo-Projekte eine super Hilfestellung sein können. Besonders in Bereichen, wo man sich noch nicht so gut auskennt. Einige Probleme lassen sich auch durch ein Neustarten der IDE lösen (Keystroke-lock). Als Letztes habe ich mir mitgenommen, nie wieder mit Android Studio zu arbeiten.

### 7.2.3 Felix Weigert

Ich habe bei diesem Projekt gelernt, wie wichtig eine gute Kommunikation zwischen allen Parteien ist. Als "Kommunikationsbeauftragter" der Gruppe habe ich viel über Organisation etc. gelernt. Außerdem ist mir bei diesem Projekt wieder klar geworden, wie wichtig es ist, alte Projekte nicht wegzurüfen. Oft Wochen später wäre es hilfreich gewesen, wenn man sich einen Code noch einmal ansehen hätte können.

## 8

## Verwendete Lösungswege mit Begründung/Erklärung

Ngrok	Ngrok war die empfohlene Lösung, um mit unserer Android-App, auf des (localhost) Backend zugreifen zu können. Ngrok tunnelt das Backend auf eine public IP und kann daher von unserer App angesteuert werden.
Base64 String	Base64 Strings war unsere Lösung wie wir ein Bild als String verschicken kann. Da wir das Bild von einer C# zu einer Android-App übermitteln wollten, brauchten wir eine einheitliche Größe => String. Einziges Problem ist, dass diese Lösung sehr unperformant ist. Für das Vorzeichen wurde das Bild zusätzlich hardcodiert.
A* Algorithmus	Ein SPF-Algorythmus, für unsere Navigations-App. Im Gegensatz zu Djekstra ist der A* besser geeignet, da er mit den Distanzen als Gewichtung arbeitet.
MathNet.Spatial	Ein Nuget, welches es erlaubt, geometrische Formen im 2-dimensionalen Koordinatensystem, zu verwenden. Dies ist Grundlage, dass man mit diesen Formen Berechnungen für die aktuelle Position des Benutzers durchführen kann.
Rest-Schnittstelle	Best-Practice für Kommunikation zwischen Frontend und Backend. Vorheriger Lösungsansatz "Sockets" war unperformant und in unserem Fall komplizierter als eine Rest-Schnittstelle.

## **9 Vorschläge/Empfehlungen Weiterentwicklung**

- Base64 String durch performantere Lösung ersetzen
- UI/UX überarbeiten
- Timer von Scanintervall auf < 3 Sekunden stellen
- Für IOS müsste man „von Vorne“ Anfangen, da IOS einen eigenen Lösungsweg für Indoor-Navigation anbietet

## **10 Statement des Auftraggebers**

### **Statement des Auftraggebers**

Das weitläufige und stetig wachsende Betriebsareal der Fill Gesellschaft m.b.H. stellt vor allem zu Beginn oftmals eine Herausforderung für Kunden, Lieferanten und neue Mitarbeiter dar. In einer Zeit, in der Effizienz, Produktivität und Komfort in der Arbeitsumgebung immer mehr an Bedeutung gewinnen, wurde auch im Unternehmen Fill das Ziel gesetzt, eine innovative Lösung zu entwickeln, um diese Navigationsherausforderungen zu bewältigen.

Aigner Michael, Raith Sebastian und Weigert Felix haben sich im Rahmen ihrer Diplomarbeit mit dem Thema der internen Navigation unter Anwendung von WLAN-Access-Points auseinandergesetzt und eine Navigations-App entwickelt, die es ermöglicht, indoor ohne Abhängigkeit von einem GPS-Signal zu bestimmten Zielorten geführt zu werden. Beispielsweise können so neue Mitarbeiter einfacherweise die kürzesten Wege zu Besprechungsräumen finden.

Wir bedanken uns bei Aigner Michael, Raith Sebastian und Weigert Felix für die gelungene Umsetzung des Projektes und freuen uns auf eine mögliche weitere Zusammenarbeit.

Ing. Fabian Wilflingseder Projektmanagement Digitalisierung

## **11 Danksagung**

Ein besonderer Dank gilt in erster Linie unserem Betreuer Herrn Prof. Hable und unserem Ansprechpartner bei der Firma Fill, Herrn Wilflingseder. Durch sie wurde diese Diplomarbeit erst möglich und sie unterstützten uns stets mit Rat und Tat.

Weiter wollen wir uns bei allen Professoren bedanken, die uns bei diversen Problemen und Fragen weitergeholfen haben. Prof. Grüneis und Prof. Planberger gaben uns bei diversen Programmierfragen den richtigen Input. Prof. Jebinger inspirierte uns beim Designen von unserem Flyer und dem Plakat.

Natürlich gilt es auch unseren Klassenkameraden zu danken. Elias Schröcker und Florian Augustin boten stets eine 'helfende Hand an' und gaben uns Anregung, wenn wir sie benötigten. Fabian Gaisböck und Maximilian Gritsch haben ebenfalls ein Dankeschön verdient für ihre Unterstützung.

Ein besonderes Dankeschön möchten wir unserer gesamten Klasse 5AHIF 2023/24 aussprechen. Ein derart angenehmes und familiäres Klima war ein wichtiger Schlüssel zum Erfolg.

## **Quellen und Literaturverzeichnis**

- [1] „Fill Gurten,“ [Online]. Available: <https://www.fill.co.at/de>. [Zugriff am 20 03 2024].
- [2] YouTube, „YouTube,“ [Online]. Available: <https://youtu.be/ySN5Wnu88nE>. [Zugriff am 31 03 2024].
- [3] „Visual Studio,“ [Online]. Available: <https://visualstudio.microsoft.com/de/>. [Zugriff am 20 03 2024].
- [4] „Android Studio Wiki,“ [Online]. Available: [https://de.wikipedia.org/wiki/Android\\_Studio](https://de.wikipedia.org/wiki/Android_Studio). [Zugriff am 19 03 2024].
- [5] „AboutGeoGebra,“ [Online]. Available: <https://www.geogebra.org/about?lang=de>. [Zugriff am 20 03 2024].
- [6] „ngrok,“ [Online]. Available: <https://ngrok.com/>. [Zugriff am 20 03 2024].
- [7] K. Stumpf, „Code Project,“ [Online]. Available: <https://www.codeproject.com/Articles/97871/WPF-simple-zoom-and-drag-support-in-a-ScrollViewer>. [Zugriff am 31 03 2024].
- [8] C. Sunwold, „Stack Overflow,“ [Online]. Available: <https://stackoverflow.com/questions/1495408/how-to-drag-a-usercontrol-inside-a-canvas>. [Zugriff am 31 03 2024].
- [9] GeeksForGeeks, „GeeksForGeeks,“ [Online]. Available: <https://www.geeksforgeeks.org/a-search-algorithm/>. [Zugriff am 31 03 2024].
- [10] „csharphelper,“ [Online]. Available: [http://csharphelper.com/howtos/howto\\_trilateration.html](http://csharphelper.com/howtos/howto_trilateration.html) . [Zugriff am 31 03 2024].
- [11] „stackoverflow,“ [Online]. Available: <https://stackoverflow.com/questions/32016811/how-to-perform-a-query-every-30-seconds-in-android> . [Zugriff am 31 03 2024].
- [12] „github,“ [Online]. Available: <https://github.com/fayyaztech/android-canvas-zoom-and-draw-over-image/tree/master> . [Zugriff am 31 03 2024].

[13] P. Sickinger, „praesMain“.

# Abbildungsverzeichnis

Abbildung 3: SPF - SPF Prototyp-App.....	12
Abbildung 4: Grundlegende Planung .....	16
Abbildung 5: Code - Karteneditor Übersicht .....	20
Abbildung 6: Code - Karteneditor CommunicatorService .....	21
Abbildung 7: Code – MapDataDto .....	21
Abbildung 8: Code - MessageService.....	22
Abbildung 9: Code – SaveService.....	23
Abbildung 10: Code - MainWindow XAML.....	23
Abbildung 11: Code – MainWindow .....	24
Abbildung 12: Code - EditorLib Übersicht .....	24
Abbildung 13: Code - Enum EditModes .....	25
Abbildung 14: Code – ImageControl.....	25
Abbildung 15: Code - ImageControl XAML.....	26
Abbildung 16: Code - EditModes .....	26
Abbildung 17: Code - FullProp Image .....	27
Abbildung 18: Code - _nodeWidth .....	27
Abbildung 19: Code - FullProp lastSelectedNode .....	27
Abbildung 20: Code - ImageControl.....	28
Abbildung 21: Code - OnNodeCanvasMouseUp.....	28
Abbildung 22: Code - InitializeDisplaceEvents .....	29
Abbildung 23: Code - MouseEventArgs.....	29
Abbildung 24: Code - StartMovingCanvas.....	30
Abbildung 25: Code - EndMovingCanvas .....	30
Abbildung 26: Code - OnMouseMove.....	30
Abbildung 27: Code - OnPreviewMouseWheel .....	31
Abbildung 28: Code - OnSliderValueChanged .....	31
Abbildung 29: Code - OnScrollViewerScrollChanged.....	32
Abbildung 30: Code - ImageControlNode .....	32
Abbildung 31: Code - AddNode .....	33
Abbildung 32: Code - AddNode .....	33
Abbildung 33: Code - RemoveNode.....	34
Abbildung 34: Code - NodeClicked .....	34
Abbildung 35: Code - AddConnection.....	35
Abbildung 36: Code - AddConnection.....	35

Abbildung 37: Code - RemoveNodeConnection .....	36
Abbildung 38: Code - OnConnectionClicked .....	36
Abbildung 39: Code - _removedConnections.....	36
Abbildung 40: Code - OnStartNodeMove .....	37
Abbildung 41: Code - OnEndNodeMove .....	37
Abbildung 42: Code - Nodes Übersicht.....	38
Abbildung 43: Code - Node XAML .....	38
Abbildung 44: Code - Node Id .....	39
Abbildung 45: Code - NodeCoordinates.....	39
Abbildung 46: Code - NodeWidth.....	39
Abbildung 47: Code - NodeName .....	39
Abbildung 48: Code - _propsToEdit .....	39
Abbildung 49: Code - DefaultColor/FillColor .....	40
Abbildung 50: Code - NodeEventHandler .....	40
Abbildung 51: Code - Node ctor .....	40
Abbildung 52: Code - UserControl_Loaded.....	41
Abbildung 53: Code - OnNodeBtnClicked .....	41
Abbildung 54: Code - EditNode .....	41
Abbildung 55: Code - OnNodeBtnDblClick .....	41
Abbildung 56: Code - Node ToString.....	42
Abbildung 57: Code - Node Draggable.....	42
Abbildung 58: Code - EventHandler.....	42
Abbildung 59: Code - InitializeDraggable .....	42
Abbildung 60: Code - StartDragging .....	43
Abbildung 61: Code - EndDragging .....	43
Abbildung 62: Code - MoveDragging .....	44
Abbildung 63: Code - MacAddress.....	44
Abbildung 64: Code - APNode .....	44
Abbildung 65: Code - ConnectedNodes .....	45
Abbildung 66: Code - AttatchedConnections .....	45
Abbildung 67: Code - IntNode .....	45
Abbildung 68: Code - EditNode XAML .....	46
Abbildung 69: Code - _properties.....	46
Abbildung 70: Code - ApplyChanges.....	46
Abbildung 71: Code – EditNodeWindow.....	47
Abbildung 72: Code - Window_Loaded .....	47
Abbildung 73: Code - TxtOnPreviesKeyDown .....	47

Abbildung 74: Code - CloseWindowWithSave.....	48
Abbildung 75: Code - CloseWindowWithoutSave .....	48
Abbildung 76: Code - OnSave/OnPreviewKeyUp .....	49
Abbildung 77: Code - ConnectionLine XAML.....	49
Abbildung 78: Code - ConnectionNode1/2 .....	49
Abbildung 79: Code - Koordinaten (x1, x2, y1, y2) .....	50
Abbildung 80: Code - NodeConnectionEvent.....	50
Abbildung 81: Code - Stroke.....	51
Abbildung 82: Code - UpdateLineDisplay.....	51
Abbildung 83: Code - ConnectoinLine_MouseUp.....	51
Abbildung 84: Code - ExtensionMethods Übersicht.....	52
Abbildung 85: Code - CanvasExtentionMethods .....	52
Abbildung 86: Code - DrawConnectionNode .....	52
Abbildung 87: Code - GetNextId.....	53
Abbildung 88: Code - ToWriteableImage.....	53
Abbildung 89: Code - RemoveConnectionsWithNode.....	54
Abbildung 90: Code -ToIntNodeDto.....	54
Abbildung 91: Code - ToNodeDtoList.....	54
Abbildung 92: Code –ToIntNode/ToApNode.....	55
Abbildung 93: Code - ToNodeList .....	55
Abbildung 94: Backend – Nuggets.....	56
Abbildung 95: Code - PositionController .....	57
Abbildung 96: Code – PositionService .....	57
Abbildung 97: Code – FindShortestPath .....	59
Abbildung 98: Code – CombinedAPs .....	60
Abbildung 99: Code - Point2D .....	60
Abbildung 100: Code - circlesChanged .....	61
Abbildung 101: Code – AreCirclesEncapsulated.....	61
Abbildung 102: Code - radius*factor .....	62
Abbildung 103: Code - CalculateCircleIntersections.....	62
Abbildung 104: Code – CalculateCircleIntersections .....	62
Abbildung 105: Code - CalculateTriangleCenter.....	62
Abbildung 106: Code - ContentPath .....	63
Abbildung 107: Code – intNodes .....	63
Abbildung 108: Code - TestIntNodes .....	63
Abbildung 109: Code - FullProp AccessPoints .....	63
Abbildung 110: Code - FullProp ImgBase64 .....	64

Abbildung 111: Code - MapService ctor .....	64
Abbildung 112: Code - InitializeTestData .....	65
Abbildung 113: Code - GetNodesAsSpfList .....	66
Abbildung 114: Code - ImageCrlController .....	66
Abbildung 115: Code - PostMapData.....	66
Abbildung 116: Code - GetImg .....	67
Abbildung 117: Code - ExecuteAsync .....	67
Abbildung 118: Code – AccessPoint.....	68
Abbildung 119: Code - Base64Handler .....	69
Abbildung 120: Code – assets.....	69
Abbildung 121: Code - readBase64FromFile .....	70
Abbildung 122: Code - decodeBase64 .....	70
Abbildung 123: Code - MainActivity Variablen.....	71
Abbildung 124: Code - MainActivity Methoden .....	72
Abbildung 125: Code - callEndpoint.....	73
Abbildung 126: Code – calculateDistance.....	73
Abbildung 127: Code – setNearestNode.....	74
Abbildung 128: Code – onPostExecute .....	74
Abbildung 129: Code - NavigationView Variablen.....	75
Abbildung 130: Code - NavigationView Methoden .....	76
Abbildung 131: Code – NodeData .....	78
Abbildung 132: Karten-Editor.....	79
Abbildung 133: Code - app Übersicht .....	80