Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Fachbereich 4: Informatik, Kommunikation und Wirtschaft

Master Thesis

Erstellung und Umsetzung eines Mixed-Reality-Konzeptes zur Echtzeitlageermittlung und Visualisierung von Personenpositionsdaten unter Verwendung von 3D Open Data Stadtmodellen

Vorgelegt von: Rico Herlt
Matrikelnummer: 546587
am: 21.03.2017

Zum Erlangen des akademischen Grades

MASTER OF SCIENCE

(M.Sc.)

Studiengang: Angewandte Informatik

Erstbetreuer: Prof. Dr.-Ing. Thomas Schwotzer
Zweitbetreuer: Dipl.-Kfm.(FH) Norman Uhlmann, M. Eng.

SELBSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

„Erstellung und Umsetzung eines Mixed-Reality-Konzeptes zur Echtzeitlageermittlung und Visualisierung von Personenpositionsdaten unter Verwendung von 3D Open Data Stadtmodellen“

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 20.03.2017

Unterschrift

ZUSAMMENFASSUNG

In dieser Arbeit ist ein Prototyp entstanden, der in der Lage ist, in Echtzeit Positionsdaten von Objekten und Personen in einem Mixed Reality Umfeld darzustellen, welches in das bestehende Projekt DOWSER als zusätzliches Hilfsmittel eingebunden werden kann. Konkret wurden die Echtzeitpositionsdaten in Form von Hologrammen von dreidimensionalen Modellen wie Plätze, Straßen oder Vierteln auf der HoloLens visualisiert. Speziell wurden Open Data Stadtmodelle aus Berlin verwendet. Als Anwender wurden die Rollen Einsatzleitung und Einsatzkräfte (bewegliche Objekte) identifiziert. Die Einsatzleitung erhält durch die Ergebnisse dieser Arbeit die Möglichkeit im Rahmen des Mixed-Reality-Ansatzes 3D-Informationen zusammen mit Echtzeitinformationen auf der HoloLens zu visualisieren. Um das zu ermöglichen, wurde im Rahmen der Arbeit erfolgreich ein Kommunikationskonzept mit Schnittstellen entworfen und implementiert.

Die erhobenen Daten werden neben der Live-Darstellung für eine spätere Auswertung gespeichert. Diese Arbeit bietet einen positiven Beitrag im Rahmen der Digitalisierungsstrategie von Behörden und Organisationen mit Sicherheitsaufgaben. Im Rahmen der Arbeit stellte sich heraus, dass die vom Land Berlin bereitgestellten Open Data Stadtmodelle nicht direkt eingebunden werden können, da diese durch Schnittstellenvorgabe nur mit erheblicher Wartezeit und in einer unzureichenden Qualität abgerufen werden können, die sich nicht für den dierekten Einsatz mit der HoloLens eignet. Für die Visualisierung wurde zusätzlich die HereIAm App entwickelt, die auch unabhängig vom Projektrahmen anderweitig zur Erfassung von Positionsdaten genutzt werden kann, aber auch für die Weiterentwicklung dieses Mixed-Reality-Ansatzes.

Als Programmiersprache wurde ausschließlich C# genutzt.

ABKÜRZUNGSVERZEICHNIS

BOS ..... Behörden und Organisationen mit Sicherheitsaufgaben
CIL ..... Common Intermediate Language
CityGML ..... City Geography Markup Language
CLR ..... Common Language Runtime
FTP ..... File Transfer Protocol
GLONASS ..... Globalnaja nawigazionnaja sputnikowaja
GPS ..... Global Positioning System
GSM ..... Global System for Mobile Communications
HTML ..... Hypertext Markup Language
HTTP ..... Hypertext Transfer Protocol
IDE ..... Integrated Development Environment
IETF ..... Internet Engineering Task Force
LBS ..... Location-based Services
LOD2 ..... Level of Detail 2
LTE ..... Long Term Evolution
MAC ..... Media Access Control
MQTT ..... Message Queue Telemetry Transport
MVVM ..... Model View ViewModel
NAVSTAR-GPS ..... Navigational Satellite Timing and Ranging – Global Positioning System
NFC ..... Near Field Communication
RAM ..... Random Access Memory
RFC ..... Requests for Comments
SDK ..... Software Development Kit
SQL ..... Structured Query Language
SSL ..... Secure Sockets Layer
TLS ..... Transport Layer Security
UMTS ..... Universal Mobile Telecommunications System
URL ..... Uniform Resource Locator
UWP ..... Universal Windows Platform
VR ..... Virtual Reality
WLAN ..... Wireless Local Area Network
XAML ..... Extensible Application Markup Language
XML ..... Extensible Markup Language

INHALTSVERZEICHNIS

Selbständigkeitserklärung ..... **I**

Zusammenfassung ..... **II**

Abkürzungsverzeichnis ..... **III**

1 Einführung in die Thematik **1**

1.1 Projektumfeld 1

1.2 Problemstellung 2

1.3 Zielsetzung 3

1.4 Abgrenzung 4

1.5 Vorgehensweise und Methodik 5

2 Grundlagen **6**

2.1 Einsatz 6

2.2 Verwendete Frameworks 7

2.2.1 Microsoft .NET Framework *7*

2.2.2 Mono Framework *8*

2.2.3 Xamarin Framework *9*

2.2.4 Microsoft .NET Core Framework *9*

2.2.5 Unity *10*

2.3 Open Data 11

2.4 Mixed Reality 11

2.5 Positionsbestimmung 18

2.6 Echtzeitübertragung 22

3 Konzept **34**

3.1 Anwendungsbereich 34

3.2 Beispielszenario: Einsatz am Gendarmenmakt 35

3.3 Rahmenbedingungen 36
3.3.1 Technische Möglichkeiten 36
3.3.2 Rechtliche Möglichkeiten 40
3.3.3 Gesellschaftliche Akzeptanz 41
3.4 Anforderungen 46
3.4.1 Anforderungen aus den Rahmenbedingungen 47
3.4.2 Funktionale Anforderungen 49
3.4.3 Nicht-funktionale Anforderungen 51
3.5 Konzept 51
3.5.1 Ideales Konzept 52
3.5.2 Konzeptüberführung 59
4 Implementierung 60
4.1 Technischer Aufbau 60
4.2 Entwicklungswerkzeuge 62
4.3 Beschreibung der Komponenten 63
4.3.1 Open Data Stadtmodelle von Berlin 63
4.3.2 HoloCommander – Unity vs. UWP 66
4.3.3 PositionHub 73
4.3.4 Here I Am 76
4.4 Bewertung der Ergebnisse 79
4.4.1 Open Data Stadtmodelle 79
4.4.2 HoloCommander 81
4.4.3 Here I Am 82
4.4.4 PositionHub 82
5 Fazit und Ausblick 83
Quellenverzeichnis VII

Abbildungsverzeichnis ..... **XVI**

Tabellenverzeichnis ..... **XVII**

Listings ..... **XVII**

1 Einführung in die Thematik

Gegenstand dieser Arbeit ist die Erstellung und Umsetzung eines Mixed-Reality-Konzeptes zur Echtzeitlageermittlung und Visualisierung von Personenpositionsdaten unter Verwendung von 3D Open Data Stadtmodellen.

Bereits am 22.05.2016 rief die Berlin Partner und die Technologie Stiftung Berlin erstmalig zu einem Hackathon mit dem Titel „#Berlin3D - Hacking Berlin's City Model“ auf (Land Berlin; BerlinOnline Stadtportal GmbH & Co. KG, 2015 a), um mit dem am gleichen Tag veröffentlichten dreidimensionalen Modell des Landes Berlin zu experimentieren. Dies wurde durch die Senatsverwaltung für Wirtschaft, Technologie und Forschung als erster flächendeckender LOD2 Datenbestand von allen Gebäuden des Landes Berlin, unter anderem im CityGML-Format , veröffentlicht. Die darin enthaltenen 3D-Modelle der Gebäude sind texturiert und basieren auf der ALK (Automatisierte Liegenschaftskarte) (Land Berlin, BerlinOnline Stadtportal GmbH & Co. KG, 2015 b) um in Form eines Programmierwettbewerbs mit den Daten zu experimentieren.

1.1 Projektumfeld

Das Projekt wird als Teil des DOWSER-Projektes der h3ko Vertriebsgesellschaft realisiert. Das primäre Ziel des DOWSER-Projekts ist die Erstellung einer mobilen Anwendung, die DOWSER-APP, mit der Behörden und Organisationen mit Sicherheitsaufgaben (BOS) (Wikipedia, 2016 a) ein Hilfsmittel zur Verfügung gestellt bekommen. Damit soll es möglich sein, dass zum Beispiel Feuerwehren oder das Technische Hilfswerk die Möglichkeit bekommen, kommunen- und länderübergreifend Wasserentnahmestellen wie Hydranten oder Brunnen zu katalogisieren und anzeigen zu lassen, um im Ernstfall, bereits auf dem Weg zum Einsatzort, einen Überblick über die nächstgelegene Löschwasserentnahmestelle zu erhalten. Zusätzlich gibt es weitere Teilprojektbereiche, wobei ein

Abbildung 1.1: DOWSER-Logo (h3ko, 2017)

Bereich mit dem Namen DOWSER-AR darauf abzielt, Augmented Reality einzusetzen, um Einsatzkräften die Arbeit zu erleichtern, indem zusätzliche Umgebungsinformationen direkt in die Schutzvisiere der Helme der Einsatzkräfte projiziert werden. In diesen Bereich fällt ebenfalls das umzusetzende Mixed-Reality-Konzept, wobei dieses Konzept eher die Anwendung bei der Einsatzleitung finden wird.

Darüber hinaus beschäftigen sich die Mitglieder der Projektgruppe DOWSER - Tech mit der Entwicklung einer Kommunikationslösung, die es ermöglichen soll, die Übermittlung von GPS (Wikipedia, 2016 a) Positionsdaten von Einsatzkräften an die Einsatzleitung aufrecht zu erhalten, selbst wenn nationale Mobilfunknetze wie GSM, UMTS oder LTE aufgrund von örtlicher Nichtverfügbarkeit (kein Mobilfunkmast in Reichweite) oder aufgrund eines Netzausfalls bedingt durch Katastrophenfälle nicht funktionieren. Die Projektgruppe DOWSER-API beschäftigt sich damit, die gesammelten Daten im Sinne von Open Data in Form einer einheitlichen Schnittstelle zur Verfügung zu stellen.

1.2 Problemstellung

Sobald es durch Alarmierungen oder im Rahmen von Prävention zu einem Einsatz im Bereich der Äußeren oder Inneren Sicherheit, der humanitären Hilfe oder dem Katastrophenschutz kommt, werden Einsatzkräfte als Angehörige von Behörden und Organisationen mit Sicherheitsaufgaben beauftragt. Dazu gehören Polizei, Feuerwehr, Rettungsdienst, Technisches Hilfswerk und sonstige herangezogenen Personen, deren Aufgabe die Gefahrenabwehr oder Schadensbekämpfung ist (Wikipedia, 2016 a). Dabei unterstehen alle Einsatzkräfte einer Einsatzleitung, die sich um eine zielgerichtete Koordination und Führung aller Einsatzkräfte kümmert. Um das zu erreichen, ist es unerlässlich, der Einsatzleitung zu jeder Zeit so viele Informationen wie möglich über das aktuelle Geschehen am Einsatzort zur Verfügung zu stellen. Da die Einsatzleitung sich nicht immer am Ort des Einsatzes befindet, werden für einen besseren Überblick sehr häufig Karten des Einsatzortes verwendet, auf denen Taktische Zeichen platziert wer-

den. Taktische Zeichen sind wiederum Symbole, die Fahrzeuge, Gebäude, Einrichtungen und Einsatzkräfte unverwechselbar darstellen. Die genaue Position der Taktischen Zeichen auf der Karte und die tatsächliche Position der Einsatzkräfte und -fahrzeuge wird derzeit in verbaler Abstimmung mit Personen am Einsatzort abgeglichen.

Dieser verbale Austausch von Informationen über genaue Standorte ist langsam, zeitverzögert und fehleranfällig ist. Die Dokumentation eines Einsatzes findet handschriftlich statt.

1.3 Zielsetzung

Ziel dieser Arbeit ist es, einen Prototyp zu entwickeln, der in der Lage ist, in Echtzeit Positionsdaten von Objekten und Personen in einem Mixed Reality Umfeld darzustellen, welches in das bestehende Projekt DOWSER als zusätzliches Hilfsmittel eingebunden werden kann. Konkret sollen die Echtzeitpositionsdaten in Form von Hologrammen von dreidimensionalen Modellen wie Plätzen, Straßen oder Vierteln auf der HoloLens visualisiert werden. Speziell sollen hier die Daten der Open Data Stadtmodelle am Beispiel Berlins verwendet werden. Die Anwender des Programms, in der Rolle einer Einsatzleitung einer Behörde oder Organisationen mit Sicherheitsaufgaben (BOS), sollen mit Hilfe von Mixed-Reality in der Lage sein, in der Einsatzzentrale einen Überblick über die sich am Einsatzort bewegenden Einsatzkräfte und Objekte, wie zum Beispiel Fahrzeuge, zu bekommen. Um das zu ermöglichen, werden diverse Informationen benötigt. Zum einen muss die Beschaffenheit des Einsatzortes in Form von dreidimensionalen Modellen bekannt sein. Darüber hinaus muss die aktuelle Position der nachzuverfolgenden Objekte und Positionen ermittelt und der Anwendung zur Verfügung gestellt werden. Mit all diesen Informationen soll die Einsatzleitung im Ernstfall in der Lage sein, einen Echtzeiteindruck über die aktuellen Positionen von Personen und beweglichen Objekten zu erhalten, um somit die Auswirkungen über zu treffende Entscheidungen besser abschätzen zu können. Darüber hinaus ist es denkbar, dass die Einsatzleitung dadurch die Möglichkeit bekommt, durch die In-

teraktion mit dem dreidimensionalen Modell des Einsatzortes die aktuellen Geschehnisse aus verschiedenen Blickwinkeln zu betrachten, parallel stattfindende Ereignisse besser zu beobachten und die Erkenntnisse in den Entscheidungsprozess mit einfließen zu lassen.

Abschließend wird geprüft, ob es mithilfe von Mixed-Reality-Geräten und der automatischen Ortung von Einsatzkräften und -fahrzeugen möglich ist. Darüber hinaus sollen die Positionsdaten für eine spätere Auswertung des Einsatzgeschehens gespeichert werden.

Zusätzlich soll verdeutlicht werden, dass sich C# als Programmiersprache schon lange nicht mehr nur für die Entwicklung von Desktopanwendungen eignet, sondern durch verschiedene Frameworks und Laufzeitumgebungen flexibel eingesetzt werden kann.

1.4 Abgrenzung

Der im Titel der Thesis vorkommende Begriff „Lageermittlung“ bezieht sich nicht auf die Bestimmung der Raumlage, also der Ausrichtung eines Objektes im dreidimensionalen Raum, sondern die Positionsbestimmung von Objekten, wie im Duden unter 1a definiert als „Stelle, wo etwas (in Bezug auf seine Umgebung) liegt“ (Duden, o.J.).

Der Kontext dieser Arbeit fokussiert sich auf das Gebiet der Bundesrepublik Deutschland, insbesondere auf die Open Data Initiative des Landes Berlin, welche dreidimensionale Modelle für ganz Berlin zur Verfügung stellt. Die prototypische Implementierung wird sich nur auf einige wenige Beispielorte beziehen. Zusätzlich wird innerhalb des Modells die Position der beweglichen Objekte und Einsatzkräfte dargestellt, um der Einsatzleitung, die sich oftmals nicht unmittelbar vor Ort befindet, einen besseren Überblick über das Geschehen zu verschaffen und so eine bessere Lageeinschätzung zu ermöglichen.

Die prototypische Umsetzung der Mixed-Reality-Darstellung der dreidimensiona-

len Stadtmodelle wird mit Hilfe der Microsoft HoloLens (Microsoft, 2016 c) realisiert, da derzeit keine vergleichbaren Geräte frei erhältlich sind, die in der Lage sind, Mixed Reality darzustellen.

1.5 Vorgehensweise und Methodik

Um die grundlegenden Voraussetzungen für den Einstieg in die Thematik zu schaffen, werden zunächst anhand vorhandener Fachliteratur die Grundlagen der Thematik erläutert und zusätzlich mit Erfahrungen aus einem speziellen Anwendungsfalles kombiniert.

Aufgrund des neuartigen Technologieansatzes von Microsoft insbesondere in Bezug auf Mixed-Reality, wird auf die Rahmenbedingungen und Umsetzungsmöglichkeiten des Konzeptes mit der HoloLens eingegangen. Die Umsetzung wird mithilfe Microsoft .NET Plattform in der Programmiersprache C# erfolgen. Das betrifft sowohl die Smartphone-Umsetzung für iOS und Android, eine Web API und die HoloLens-Anwendung an sich.

Im Konzept wird eine Lösung skizziert, welche aufgrund der hergeleiteten Anforderungen letztlich überprüft wird.

Für Leser, die das Lesen digitaler Medien bevorzugen, befindet sich die vollständige Arbeit samt Internetquellen auf einer CD-ROM im Anhang.

2 GRUNDLAGEN

Diese Thesis basiert auf Grundlagenwissen zu verschiedenen Themengebieten. Zum einen werden Einsätze im Rahmen von Behörden und Organisationen mit Sicherheitsaufgaben definiert, denn dieses Konzept soll durch in einem Einsatz beteiligte Personen angewendet werden. Im Anschluss werden die während der Umsetzung verwendeten Frameworks näher erläutert, um deren Entstehung in Bezug auf die gemeinsame Programmiersprache C# aufzuzeigen. Danach erfolgt eine Einführung in Open Data, um in die Thematik der Veröffentlichung von Daten von Behörden und Verwaltungsapparaten einzuleiten und auch um zu zeigen, wie Open Data in Bezug zur Mixed Reality gebracht werden kann. Mixed Reality wird im darauffolgenden Kapitel definiert. Darüber hinaus wird eine Abgrenzung zur realen und der virtuellen Realität durchgeführt um aufzuzeigen, worin sich diese Realitäten unterscheiden. Anschließend wird in das Thema der Positionsbestimmung eingeleitet und beschrieben, mit welchen technischen Möglichkeiten Geräte derzeit in der Lage sind, ihre eigene Position zu ermitteln. Im abschließenden Abschnitt zur Echtzeitübertragung wird zuerst der Begriff „Echtzeit“ in einen technischen Bezug eingeordnet und anschließend werden technische Möglichkeiten dargestellt, um die von den Geräten ermittelte eigene Position möglichst schnell und effizient zu übertragen.

2.1 Einsatz

Behörden und Organisationen mit Sicherheitsaufgaben sind in Deutschland für die Gefahrens- und Schadensbekämpfung und -abwehr zuständig. Dabei kann der Gefährdungsgrad eines Schadensereignisses oder einer Gefahrenlage auch während des Einsatzes erst entstehen, wie zum Beispiel bei einem Großbrand oder einem Hochwasser, aber auch ursächlich abgeschlossen sein, wie bei einem Erdbeben oder einem Zugunglück. Unabhängig davon erfordern auch kleinere Einsätze sehr häufig erhebliche technische und organisatorische Maßnahmen.

Die Einsatzleitung hat die Aufgabe, alle Maßnahmen zur Gefahrenabwehr und

zur Schadensbegrenzung zu veranlassen und dabei Einsatzkräfte möglichst wirkungsvoll an den meist unbekannten Einsatzorten einzusetzen. Um das zu ermöglichen muss die Einsatzleitung fähig sein, die aktuelle Lage schnell zu erfassen, sie zu beurteilen um möglichst schnell eine Entscheidung zur Handlung zu treffen und diese anschließend als Einsatzbefehl den Einsatzkräften zu kommunizieren. Dabei hängt der Erfolg des Einsatzes wesentlich vom reibungslosen Funktionieren der Einsatzleitung ab. (Feuerwehr-Dienstvorschrift 100, 1999 S. 7)

2.2 Verwendete Frameworks

In der Umsetzung dieser Arbeit kommen verschiedene Frameworks zum Einsatz. Alle dieser Frameworks bieten die Möglichkeit, in der Programmiersprache C# zu entwickeln. Die Besonderheit dieser Frameworks besteht darin, dass sie neben dem Compiler verschiedene Laufzeitumgebungen für verschiedene Plattformen mitbringen. Dadurch ist es möglich, mobile verteilte Systeme in nur einer Programmiersprache zu entwickeln, von der Smartphone Anwendung für iOS und Android über Webanwendungen, die neben Windows auch unter Linux und Apple OSX lauffähig sind, bis hin zu Universal Windows Anwendungen, die auf Mikrokontrollern, wie dem Raspberry Pi, oder Mixed-Reality-Brillen, wie der HoloLens, lauffähig sind.

2.2.1 Microsoft .NET Framework

Das Microsoft .NET Framework ist eine Plattform zur Entwicklung und Ausführung von Anwendungen. Es besteht sowohl aus einer Laufzeitumgebung (Common Language Runtime, kurz CLR), mit deren Hilfe die für das .NET Framework entwickelten Anwendungen ausgeführt werden, als auch aus einer umfangreichen Sammlung von Klassenbibliotheken, die grundlegende Funktionalitäten und Hilfswerkzeuge für Entwickler bereitstellt (Wikipedia, 2017 d). Alle .NET-Anwendungen werden unabhängig von ihrer Programmiersprache in eine Zwischensprache (Common Intermediate Language, kurz CIL), ähnlich dem bekannten Java Byte-Code, übersetzt (Wikipedia, 2016 c) und während der Ausführung von einem Just-in-Time-Compiler in einen nativen Maschinencode übersetzt (Wikipedia, 2016 d).

Aktuell liegt das .NET Framework in der Version 4.7 als Vorabversion in Windows 10 Insider Builds vor. Bei neueren Microsoft-Windows-Client-Versionen bildet das Framework einen festen Bestandteil des Betriebssystems und ist in der Grundeinstellung für Anwender bereits vorinstalliert, sodass eine separate Installation nicht erforderlich ist. (Wikipedia, 2017 d)

2.2.2 Mono Framework

Mono ist eine quelloffene Implementierung des .NET Frameworks von Microsoft (siehe vorheriger Abschnitt). Aktuell ist die Version 4.6 Service Release 2 (4.6.2.16) verfügbar (Mono, o. J. b). Durch das Mono Framework wird die Implementierung plattformunabhängiger Software auf Basis der Common Language Infrastructure unter der Verwendung der Programmiersprache C# ermöglicht. Als Entwicklungsumgebung steht MonoDevelop für Windows, Linux und OSX zur Verfügung. (Mono, o. J. b)

Das Mono Framework unterstützt sehr viele Komponenten des Microsoft .NET Frameworks der Version 4.5. Nicht unterstützt werden Windows Presentation Foundation (WPF) und Windows Workflow Foundation (WWF). Für die Windows Communication Foundation (WCF) wird nur eingeschränkt Unterstützung angeboten. Darüber hinaus unterstützt der Mono Compiler die Version 6.0 der Programmiersprache C#. (Mono, o. J. a)

Als Microsoft Mitte 2000 das .NET Framework vorstellte, kam Miguel de Icaza, damaliger Projektmitarbeiter der Firma Ximian, auf die Idee, eine Version des .NET Frameworks für Linux zu entwickeln. Gut ein Jahr später, am 19.Juli 2001, wurde Mono als quelloffenes Projekt gestartet. Als im August 2003 Novell die Firma Ximian übernahm, fungierte Novell als Hauptsponsor für das Projekt. Durch die Übernahme von Novell selbst durch die Attachmate-Gruppe im Aprill 2011, wurde die Rolle des Hauptsponsors durch das von Miguel de Icaza neu gegründete Unternehmen Xamarin übernommen (Wikipedia, 2017 c), welches sein Xamarin Framework für die native mobile Cross-Plattform-Entwicklung auf der Basis des Mono Frameworks anbot. Nach der Übernahme Xamarins als Tochtergesellschaft durch Microsoft im Jahr 2016 wurde Microsoft folglich

Hauptsponsor des Projektes. (Mono, o. J. a)

2.2.3 Xamarin Framework

Von der Gründung des Unternehmens Xamarin wurde am 16. Mai 2011 von Miguel De Icaza (ebenfalls der Gründer von Mono) auf seinem persönlichen Blog berichtet. (Icaza, 2011) Dabei wurde angekündigt, dass es nun eine kommerzielle Version des Mono Frameworks kompatibel für die OSX, iOS und Android Plattformen geben soll.

Im Februar 2013 war es dann nach dem Xamarin Release 2.0 (Friedman, 2013) das erste Mal möglich, mithilfe der IDE Xamarin Studio, welches ein Fork der Open Source IDE Mono Develop darstellt, native Software für OSX, iOS, Android in einer einheitlichen Entwicklungsumgebung in der Programmiersprache C# zu entwickeln. Gleichzeitig wurden Erweiterungen für die Integration in das Microsoft Visual Studio veröffentlicht, mit deren Hilfe es erstmals möglich war, native mobile Cross-Plattform-Anwendungen mit einer einheitlichen Anwendungslogik für Android, iOS und Windows Phone in C# zu entwickeln.

Heute sind nach der Übernahme von Xamarin durch die Firma Microsoft im März 2016 die Entwicklungswerkzeuge für Xamarin Projekte fest im Visual Studio ab Version 2015 integriert.

2.2.4 Microsoft .NET Core Framework

Das .NET Core Framework wurde unter Koordination von Microsoft entwickelt und stellt eine quelloffene Plattform innerhalb der .NET Plattform dar. Es dient zur Ausführung von Programmen und ist im Gegensatz zum .NET Framework neben Windows auch unter Linux und OSX lauffähig und somit als plattformunabhängiges Framework entworfen worden. Es ist seit November 2016 in der Version 1.1 verfügbar und wird rasant weiterentwickelt. Ein großes Ziel, das es zu erreichen gibt, ist die Angleichung der API an das .NET Framework, um .NET

Core langfristig als plattformunabhängigen Hauptzweig innerhalb der .NET Technologie werden zu lassen. (Wikipedia, 2017 a)

2.2.5 Unity

Unity stellt eine plattformunabhängige Umgebung der Firma Unity Technologies dar. Die Entwicklungsumgebung namens Unity Editor ist für Windows, Linux und OSX verfügbar. Mit ihr lassen sich Spiele und andere interaktive 3D-Grafikprogramme für verschiedenste Spielkonsolen (Wii, Playstation, Xbox), PC-Plattformen (Windows, OSX, Linux), mobile Geräte (iOS, Android, Windows) und Webbrowser (WebGL) entwickeln.

Die in Unity erstellten Mechanismen können über sogenannte Skripte programmatisch erweitert werden. Die Skripte selbst basieren auf dem Mono Framework und können neben UnityScript (sehr hohe Ähnlichkeit zu JavaScript) auch in C# implementiert werden. Die Implementierung und das Debugging kann in beliebigen IDEs erfolgen, wie zum Beispiel dem Visual Studio unter Windows oder dem MonoDevelop unter Windows, Linux und OSX.

Neben der plattformübergreifenden Erstellung von Spielen und 3D-Programmen unterstützt Unity auch das Einbinden von plattformabhängigen C++ Bibliotheken, um performanzkritische Erweiterung zu implementieren.

Unity bietet eine offene Softwarearchitektur, die auf Socket-Kommunikation und der .NET Kompatibilität über Mono und Erweiterungen in C++ basiert. Dies führte dazu, dass in den vergangenen Jahren viele Erweiterungen, die über die reine Spieleentwicklung hinaus gehen, für Unity entwickelt wurden. Dazu zählt die Kompatibilität vieler Virtual Reality Systeme, wie zum Beispiel Moddle VR, Oculus Rift, HTC Vive, aber auch zu vieler Augmented Reality Systeme, wie Qualcomm Vuforia, Metaio oder der Microsoft HoloLens. (Wikipedia, 2017 b)

2.3 Open Data

Open Data beschreibt ein Prinzip für die freie Nutzbarkeit und Verfügbarkeit von öffentlichen Daten. Neben der Nutzbarkeit werden jedoch weitere wichtige Punkte in diesem Zusammenhang beschrieben, wie der Zugang, die Weiterverbreitung, die Einschränkungen des Einsatzzweckes und viele weitere Kriterien. (Prof. Dr. Jörn von Lucke, 2010 S. 2) Prof. Dr. Jörn von Lucke beschreibt Open Data in einem Gutachten über frei verfügbare Daten des öffentlichen Sektors wie folgt: „Offene Daten sind sämtliche Datenbestände, die im Interesse der Allgemeinheit der Gesellschaft ohne jedwede Einschränkung zur freien Nutzung, zur Weiterverbreitung und zur freien Weiterverwendung frei zugänglich gemacht werden“. (Prof. Dr. Jörn von Lucke, 2010 S. 3)

Die Senatsverwaltung für Wirtschaft, Technologie und Forschung in Berlin beauftragte den Fraunhofer Fokus im Rahmen einer Studie ein Konzept, Pilotsystem und Handlungsempfehlungen für das Vorhaben, der Entwicklung einer Berliner Open Data-Strategie, zu erarbeiten. Diese Studie wurde im Januar 2012 veröffentlicht. Obwohl im Vergleich „internationaler Beispiele für die Bereitstellung offener Verwaltungsdaten“ (Dr. Nils Barnickel, 2012 S. 37) die Erfassung von LOD2 (Level of Detail 2) genannt wird, umfasst die Studie keine konkreten Pläne zur Erstellung eines 3D-Stadtmodells von Berlin (Dr. Nils Barnickel, 2012). Trotz allem veröffentlichte die Senatsverwaltung für Wirtschaft, Technologie und Forschung am 22.05.2016 erstmalig einen flächendeckenden LOD2 Datenbestand von allen Gebäuden des Landes Berlin im CityGML-Format. Die darin enthaltenen 3D-Modelle der Gebäude sind texturiert und basieren auf der ALK (Automatisierte Liegenschaftskarte) (Land Berlin, BerlinOnline Stadtportal GmbH & Co. KG, 2015 b).

2.4 Mixed Reality

Als Ivan E. Sutherland bereits vor fast 50 Jahren (Sutherland, 1968) erstmals die Idee von Verschmelzung wirklicher und virtueller Realität mittels eines auf dem Kopf tragbaren Bildschirms auf der American Federation of Information Processing Societies (kurz AFIPS) Fall Joint Computer Conference in San Francisco

vorstellte, beschrieb er sie wie folgt:

„Die fundamentale Idee hinter dreidimensionalen Bildschirmen besteht darin, dem Anwender ein perspektivisches Bild zu präsentieren, welches sich ändert, wenn er sich bewegt. Das retinale Bild von Objekten, die wir [durch unser Auge] sehen, ist im Grunde genommen zweidimensional.“ (Sutherland, 1968)

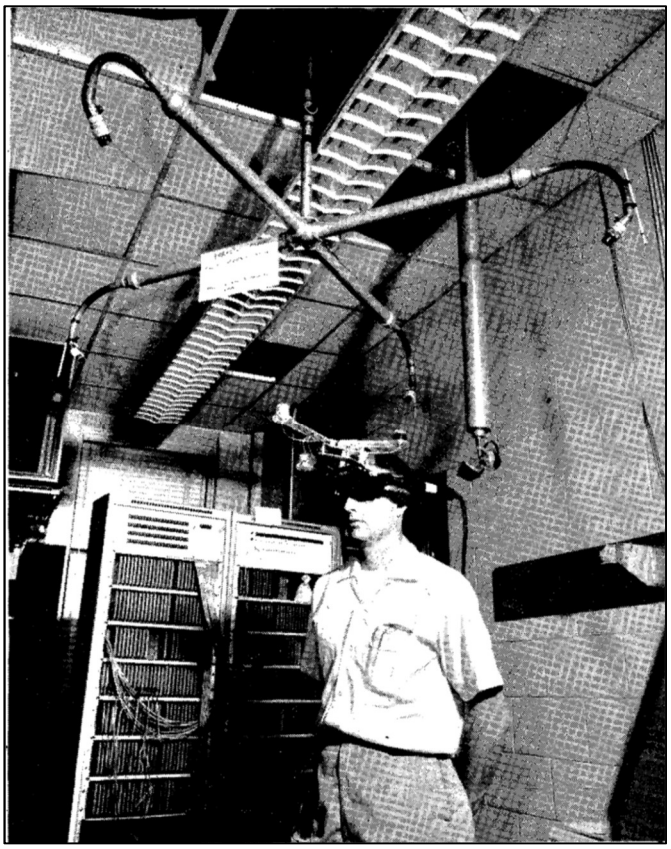

Abbildung 2.2: Prototyp Headmounted Display (Sutherland, 1968 S. 758)

Sutherland beschreibt in seinem Paper „A head-mounted three dimensional display“ den Versuch, wirkliche und virtuelle Realität mit Hilfe von vor den Augen montierten Miniatur-CRT-Bildschirmen darzustellen. Um Kopfbewegungen und somit die Blickrichtung und die Position des Kopfes im realen Raum des Anwenders zu erkennen, wurden zwei Sensoren verwendet. Zum einen wurde ein Ultraschallsensor in der Brille untergebracht und zum anderen kam ein mechanischer

Positionssensor, der fest mit dem Kopf des Anwenders und der Decke des Raumes verbunden war, zum Einsatz, um die genaue Berechnung zwischen Rotation und Translation des Kopfes im Verhältnis zum Raum zu berechnen.

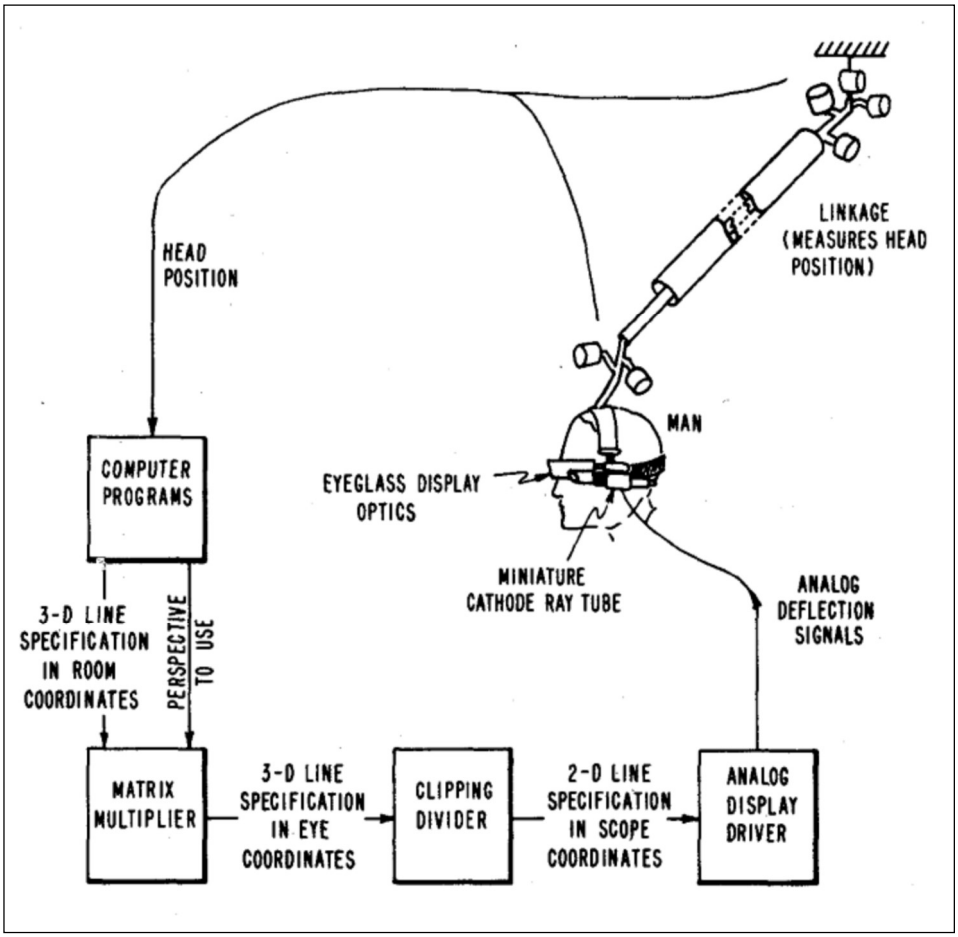

Abbildung 2.3: Funktionsweise Headmounted Display (Sutherland, 1968 S. 760)

Obwohl das Thema Mixed Reality nicht neu ist, hat es nie wirklich an Aktualität verloren. Es ist heute dank der Fortschritte in der Computertechnik präsenter denn je. Neben den vielen wissenschaftlichen Studien und Forschungen zu diesem Thema finden auch regelmäßig wissenschaftliche Konferenzen statt, wie die internationale VAMR (Virtual, Augmented and Mixed Reality), die als Teil der internationalen HCI (Human Computer Interaction) Konferenz in diesem Sommer in Vancouver, Kanada zum 9. Mal stattfinden wird. (HCI International, o. J.). Um eine klare Unterscheidung zwischen der realen, erweiterten und virtuellen Umgebung zu schaffen und somit eine Klassifizierung von Anwendungen sowie Gerä-

ten zu ermöglichen, beschrieben Paul Milgram und Fumino Kishino in der wissenschaftlichen Arbeit „A Taxonomy of Mixed Reality Visual Displays“ (Milgram, et al., 1994) den fließenden Übergang zwischen der reinen realen und der reinen virtuellen Realität und brachten erstmals die zusammenfassende Bezeichnung Mixed Reality hervor, die von nun an alle Umgebungen beschreiben soll, die eine Mischung dieser beiden Realitäten darstellt.

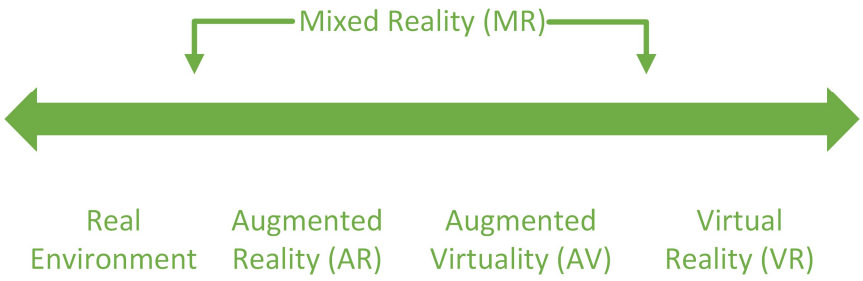

Abbildung 2.4: Vereinfachte Darstellung des Virtualitätskontinuums nach Milgram und Kishino

Bevor eine Klassifizierung der Übergänge zwischen den Realitäten möglich ist, ist eine Definition der Begriffe „Virtuell“ und „Real“ erforderlich. Milgram und Kishino definieren Objekte beider Umgebungen nach drei verschiedenen Aspekten:

- “Echte Objekte sind alle Objekte, die eine wirkliche objektive Existenz besitzen” (Milgram, et al., 1994 S. 6)
- „Virtuelle Objekte sind alle Objekte, die im Wesentlichen oder im Effekt existieren, aber nicht förmlich oder in Wirklichkeit.“ (Milgram, et al., 1994 S. 7)

Nach dem ersten Gesichtspunkt müssen reale Objekte direkt angesehen oder aus einer musterhaften Objektvorlage von einem Bildschirm dargestellt werden können. Damit virtuelle Objekte hingegen angesehen werden können, müssen sie zuvor simuliert werden. Dabei kommt eine Beschreibung oder ein Modell dieses Objektes zum Einsatz. (Milgram, et al., 1994 S. 7).

Als zweites Kriterium wird die Bildqualität von Objekten unterschieden. Betrachtet man reale Objekte indirekt, also zum Beispiel durch einen Bildschirm, müssen

zuvor Daten über dieses Objekt aufgenommen (bzw. abgetastet) werden. Diese Aufnahme könnte durch eine Videokamera, einen Laser oder einen Ultraschallsensor geschehen. Anschließend werden die ermittelten Daten mithilfe eines Anzeigegerätes, wie zum Beispiel einem Bildschirm, künstlich rekonstruiert dargestellt. Virtuelle Objekte hingegen können nicht aufgenommen (bzw. angetastet) werden, da sie nicht existieren. Bei der Darstellung von virtuellen Objekten ist lediglich die visuelle Rekonstruktion des Bildes aufgrund von Daten möglich. Demnach muss ein Bild eines Objekts was sehr real aussieht, nicht zwangsläufig real sein. Als dritter Aspekt zur Unterscheidung zwischen realen und virtuellen Bildern wird die Leuchtkraft von Objekten betrachtet. Ein reales Objekt strahlt eine gewisse Leuchtkraft an dem Ort aus, an dem es sich befindet, ganz gleich ob dieses Objekt direkt oder durch einen Bildschirm betrachtet wird. Ein virtuelles Objekt hingegen weist an dem Ort, an dem es sich zu befinden scheint, keine eigene Leuchtkraft auf, so wie es zum Beispiel bei Hologrammen oder Spiegelbildern der Fall ist. (Milgram, et al., 1994 S. 7)

Nach dieser Definition sehen wir in einer Mixed Reality Umgebung Bilder von virtuellen Objekten eingebettet in einer realen dreidimensionalen Umgebung. Die Objekte selbst erscheinen transparent, was bedeutet, dass die dahinterliegenden Objekte nicht komplett überdeckt werden.

Obwohl SEGA bereits im Jahr 1991 die erste öffentlich erhältliche, auf dem Kopf tragbare Virtual Reality Brille vorstellte, kam das Projekt auf Grund von Problemen bei der Entwicklung nicht wirklich über den Prototypen-Status hinaus (Horowitz, 2004). Erst 2012 kam die aus einem Kickstarter Projekt finanzierte Oculus Rift als erste frei erhältliche VR-Brille auf den Markt (Oculus, 2012) und wurde ein großer Erfolg. Bis jedoch das erste Gerät marktreif war, das in der Lage war, echte Mixed-Reality-Umgebungen zu erzeugen, sind weitere 4 Jahre vergangen. Erst im Frühjahr 2016 brachte Microsoft laut eigenen Angaben mit der HoloLens das erste unabhängige Gerät auf den Markt, das in der Lage ist, digitale Inhalte in der realen Umgebung anzuzeigen. (Microsoft, 2016 c)

„Microsoft HoloLens ist der erste eigenständige holographische Computer, der

dir ermöglicht, mit deinen Digitalen Inhalten und mit Hologrammen in deiner Umgebungswelt zu interagieren.“ (Microsoft, 2016 c)

Im Rahmen dieser Arbeit war es möglich eine der ersten offiziell verfügbaren HoloLenses zu erhalten. Da bisher keine anderen Geräte auf dem freien Markt erhältlich sind, die in der Lage sind Mixed-Reality in Form von Hologrammen ins Sichtfeld des Anwenders zu projizieren, wird die prototypische Umsetzung des Konzeptes auf Basis der Microsoft HoloLens realisiert. Die HoloLens ist als Entwickler-Edition seit November 2016 in Deutschland erhältlich. Mit den bisher erhältlichen Augmented Reality und Virtual Reality Geräten war es bisher ausschließlich möglich sich in einer dieser beiden Welten zu begeben. Mit diesem Gerät ist es nunmehr erstmals möglich, mit Personen, Plätzen und Objekten aus der echten physischen Welt und der virtuellen Welt zusammen in einer verschmolzenen Umgebung zu interagieren. (Microsoft, 2016 b). Dabei ist besonders auffällig, dass die ins Sichtfeld der Anwender projizierten Hologramme im Gegensatz zu Augmented Reality Umgebungen nicht starr ins Blickfeld des Anwenders geheftet sind (vergl. Head-up-Display), sondern vielmehr durch die HoloLens an den Ort projiziert werden, wo sich das Objekt tatsächlich zu befinden scheint. Dem Anwender vermittelt dies ein stärkeres Gefühl dafür, dass das Objekt tatsächlich real existiert. Unterstützt wird das visuelle Empfinden durch eine räumliche Akustik (spartial sound), ausgegeben durch die Lautsprecher, die sich beim Tragen des Gerätes auf dem Kopf direkt über den Ohren befinden. Die HoloLens ist in der Lage, die Position und Blickrichtung des Trägers in Verbindung mit den virtuellen Objekten im Raum zu berechnen. Somit ist es möglich, dass man die Position der virtuellen Objekte im Raum nicht nur visuell, sondern auch akustisch wahrnehmen kann.

Die HoloLens ist mit den folgenden Spezifikationen ausgestattet:

HoloLens Device Specifications	
Optics See-through holographic lenses (waveguides) 2 HD 16:9 light engines Automatic pupillary distance calibration Holographic Resolution: 2.3M total light points Holographic Density: >2.5k radiants (light points per radian)	**Power** Battery Life 2-3 hours of active use Up to 2 weeks of standby time Fully functional when charging Passively cooled (no fans)
Sensors 1 IMU 4 environment understanding cameras 1 depth camera 1 2MP photo / HD video camera Mixed reality capture 4 microphones 1 ambient light sensor	**Processors** Intel 32 bit architecture with TPM 2.0 support Custom-built Microsoft Holographic Processing Unit (HPU 1.0)
Human Understanding Spatial sound Gaze tracking Gesture input Voice support	**Memory** 64GB Flash 2GB RAM
Input / Output / Connectivity Built-in speakers Audio 3.5mm jack Volume up/down Brightness up/down Power button Battery status LEDs Wi-Fi 802.11ac Micro USB 2.0 Bluetooth 4.1 LE	**OS and Apps** Windows 10 Windows Store
What you need to develop Windows 10 PC able to run Visual Studio 2015 and Unity	**Weight** 579g

Tabelle 3.1: Gerätespezifikationen Microsoft HoloLens (Microsoft, 2016 a)

2.5 Positionsbestimmung

Die Freigabe von NAVSTAR-GPS, für den zivilen Einsatz im Jahr 2000 (International Committee on Global Navigation Satellite Systems, 2016 S. 6) sorgte für einen regelrechten Boom im Einsatz von Navigationsgeräten und Navigationsdiensten. Bei GPS handelt es sich um ein Satellitennavigationssystem, welches die Bestimmung der eigenen Position auf der Erde ermöglicht. Ursprünglich wurde GPS als Kurzform von NAVSTAR GPS verwendet. Mittlerweile ist diese Abkürzung sowohl umgangssprachlich als auch fachsprachlich so etabliert, dass unter diesem Begriff verschiedenste Satellitennavigationssysteme zusammengefasst werden.

Aktuell arbeiten mehr als vier Satellitennavigationssysteme. Die bekanntesten sind wohl NAVSTAR-GPS aus den USA, GLONASS aus der russischen Föderation und das europäische Galileo-Programm. (International Committee on Global Navigation Satellite Systems, 2016 S. 3-55)

Name	**Herkunft**	**Satellitenanzahl**	**Bahnhöhe**
NAVSTAR-GPS	USA	27	20.200 km
GLONASS	Russische Föderation	24	19.100 km
Gallileo	Europäische Union	18 (30 geplant)	23.260 km
Beidou	China	14 (35 geplant)	21.528 km

Tabelle 3.2: Übersicht über die bekanntesten Satellitennavigationssysteme

Obwohl sich alle dieser Satellitensysteme in der Satellitenanzahl, Bahnhöhe im Orbit und in der verwendeten Funktechnologie, wie dem Frequenzband oder dem verwendeten Übertragungsverfahren (Frequenzmultiplex, Zeitmultiplex, Codemultiplex) unterscheiden, funktionieren sie prinzipiell sehr ähnlich:

Jeder Satellit, ausgestattet mit einer sehr genauen Uhr (meist Atomuhr), überträgt stetig seine aktuelle Position über der Erde zusammen mit der aktuellen Uhrzeit. Sobald ein GPS-Empfangsgerät das Signal von drei Satelliten gleichzeitig empfängt, ist es in der Lage, durch Berechnung der Signallaufzeit eines jeden eintreffenden Signals die Entfernung zum sendenden Satelliten zu bestimmen. Aus den Entfernungen zum Satelliten und der zusätzlich bekannten Position im Orbit wird wiederum durch eine mathematische Berechnung (Trilateration) der eigene Standort im zweidimensionalen Raum ermittelt. Sobald ein Signal eines vierten Satelliten hinzukommt, ist es möglich, die Position im dreidimensionalen Raum zu berechnen (Multilateration). Als Ergebnis kann zusätzlich zum Längen- und Breitengrad der Position auch die Höhe über dem Meeresspiegel ermittelt werden. Die Verfügbarkeit der Signale von mehr als vier Satelliten steigert die Genauigkeit der ermittelten Position, denn durch äußere Einflüsse in der Atmosphäre können Laufzeitfehler der Signale, verursacht durch Ionosphäreneinflüsse, hervorgerufen werden.

Moderne Smartphones unterstützen meist mehr als eins dieser Satellitensysteme. Das iPhone 7 von Apple unterstützt den Empfang von NAVSTAR GPS und GLONASS (Apple, o. J. c). Samsung Geräte der S7 Serie unterstützen zusätzlich zu den beiden noch das Beidou-System (Samsung, o. J.). Das Galileo-System wird aktuell von keinem der gängigen Smartphones unterstützt. Die Ursache wird damit begründet sein, dass das System seine Dienste offiziell erst seit dem 15. Dezember 2016 anbietet. (Europäische Kommission, 2016). Viele bestehende GPS-Empfänger können jedoch über ein Firmware-Update die Kompatibilität zu Galileo erreichen, denn sowohl NAVSTAR GPS als auch Galileo nutzen, nach einem Kompromiss zwischen der Europäischen Union und der USA, zugunsten maximaler Kompatibilität, mit dem „open service“-Signal, das gleiche Datenformat. (International Committee on Global Navigation Satellite Systems, 2016 S. 24)

Neben der Positionsbestimmung per GPS stehen mobilen Geräten heutzutage einige alternative Möglichkeiten zur Verfügung. Diese Möglichkeiten sind vor allem dort notwendig, wo der Empfang von GPS aus technischen Gründen nicht

sichergestellt bzw. realisiert werden kann. Dies ist vor allem in urbanen Umgebungen oder innerhalb von Gebäuden der Fall. Die Notwendigkeit für die Positionsbestimmung hat sich aus der Erstellung von standortbezogenen Diensten (englisch Location-based Services, LBS) ergeben. So existieren zum Beispiel Anwendungen, die dem Benutzer interessante Orte in der Nähe zeigen (Points of Interest) oder die in der Lage sind anzuzeigen, wo sich die eigenen Freunde gerade aufhalten. Darüber hinaus gibt es Anwendungen, die mithilfe von LBS in Form von Geofencing einen imaginären geografischen Zaun bilden können um zu erkennen, ob der Anwender mit seinem mobilen Gerät einen bestimmten Bereich verlässt, um auf diesem Weg standortbasierte Aktionen auszulösen, wie zum Beispiel Erinnerungen, (Apple, o. J. a) wie in diesem Anwendungsfall: Sobald ich nach Hause komme, möchte ich daran erinnert werden die Waschmaschine einzuschalten. Ein zweites Szenario könnte eine Diebstahlwarnanlage sein, die dem Eigentümer eines Fahrzeuges eine Nachricht per E-Mail oder SMS zukommen lässt, sobald das Fahrzeug unbemerkt aus der Garage entfernt wird.

Heutzutage bestehen für mobile Geräte verschiede Möglichkeiten, um den eigenen Standort unabhängig von Satellitennavigationssysteme zu bestimmen. Bereits Ende der 1990er war es der Federal Communications Commission (FCC) der USA möglich, Positionen von Mobiltelefonen aus den Informationen des Netzbetreibers zu bestimmen, sofern es sich beim Telefonat um ein sogenannten E-911- Anruf (Emergency 911) handelte. Das europäische Pendant, die E-112-Anrufe, wurden im Jahr 2003 von der Europäischen Kommission initiiert. (Gentile, et al., 2013 S. 10). Diese Art der Lokalisierung funktionierte ursprünglich über die Ermittlung der Zell-Id des Sendemastes der Mobilfunkanbieter. Jedes Mobiltelefon baut eine Verbindung zu einer nahegelegenen Basisstation auf. Es befindet sich dabei in einer von mehreren Funkzellen, die die Basisstation ausstrahlt. Da die Position der Basisstation und der Zelle bekannt ist, kann darauf auf die Position des Mobiltelefons geschlossen werden. Die Größe einer Mobilfunkzelle kann von weniger als 100 Metern bis hin zu 35 Kilometer variieren, abhängig davon wie hoch die Dichte der Teilnehmer an einem bestimmten Ort sind (Gentile, et al., 2013 S. 150), eignet sich die Bestimmung der Position durch dieses Verfahren nur bedingt. Daraufhin sind weitere Verfahren erstellt worden, die

eine bessere Schlussfolgerung auf den eigenen Standort ermöglichen.

Diese Verfahren lassen sich grundsätzlich basierend auf den verwendeten Informationen einteilen. Dazu zählen Umgebungsinformationen (Zell-Id), Distanzmessungen zu bekannten Sendestationen, Richtungsmessung der eintreffenden Funksignale und Prüfung und Bildung eines Standortes basierend auf vorherigen Signalstärkemessungen. (Gentile, et al., 2013 S. 142)

Verfahren basierend auf	**Beispiele / eingesetzte Algorithmen**
Umgebungsinformationen	- Zell-ID von Mobilfunkzellen
Distanzmessungen	- Empfangene Signalstärke (Received signal strength, RSS) - Gemessene Ankunftszeit (Time of arrival, TOA) - Absolute Ankunftszeit (Time difference of arrival, TDOA)
Richtungsmessungen	- Ankunftswinkel (Angle of arrival, AOA)
Informationsprüfungen	- Bildung eines „Fingerabdrucks“ basierend auf vergangene Signalstärkemessungen

Tabelle 3.3: Verfahren zur Positionsbestimmung

Neben der Betrachtung von GPS- oder Mobilfunksignalen kommen seit einiger Zeit alternative Signalquellen in Betracht. Die Firma Skyhook aus Boston entwickelt seit 2003 mit großem Erfolg Lösungen, um eine Positionsbestimmung mithilfe von stationären WLAN-Netzen zu ermöglichen (Skyhook, o.J.). Um das zu realisieren wurde eine Datenbank erstellt, in der die ausgestrahlten MAC-Adressen von frei verfügbaren WLAN Access Points aus Büros und Wohnungen in Verbindung mit ihrem physischen Standort gespeichert wurden. Als Ergebnis sind mobile Geräte in der Lage, mit den Informationen über die WLAN-Netze und

durch Distanzmessverfahren wie RSS bis auf wenige Meter genau den eigenen Standort zu bestimmen. (Gentile, et al., 2013 S. 2)

Neben der Positionsbestimmung auf Basis von Funksignalen besteht heutzutage zusätzlich die Möglichkeit den Ort eines Netzwerkgerätes über die zugewiesene IP-Adresse zu ermitteln, die dem Netzwerkadapter zugewiesen wurde. Dabei handelt es sich jedoch nicht um die IP im LAN, sondern um die öffentliche WAN-Adresse des Internet-Gateways, was in den Meisten Fällen der Router sein dürfte (Koch, et al., 2013 S. 1).

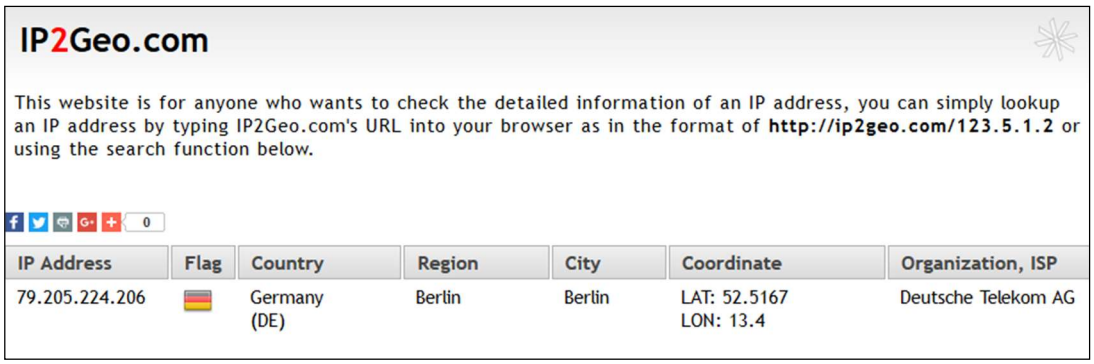

Abbildung 2.5: Ergebnis einer Positionsermittlung durch IP2Geo.com

Dieses Verfahren wird eingesetzt, um auf mehrere Kilometer (ca. 25 km -100 km) genaue Angaben über die Stadt oder die Region zu machen, aus der der Netzwerkaufruf erfolgt ist. Diese Positionsbestimmung ist in den meisten Fällen nicht genau genug, um eine Position konkret zu lokalisieren und daher für das aktuelle Vorhaben ungeeignet.

2.6 Echtzeitübertragung

Damit ein System geschaffen werden kann, welches in der Lage ist, Positionen von entfernten Objekten oder Positionen auszuwerten und darzustellen, muss ein mobiles Objekt seine durch eine, der im vorherigen Abschnitt erläuterten Positionsbestimmungsverfahren selbst ermittelte, Position in einer beliebigen Art und Weise zur darstellenden Informationsverarbeitungseinheit übertragen. In dem Moment, indem ein mobiles Objekt (A) die Position von einem anderen Objekt (B) feststellt, indem das zu lokalisierende, mobile Objekt (A) seine eigene

Position ermittelt und diese über einen beliebigen Kanal zurück an Objekt (B) überträgt, handelt sich nicht länger nur um eine Positionsbestimmung, sondern um eine Ortung vom mobilen Objekt (A), durchgeführt von Objekt (B). (Schnabel, o.J.) Der Rückkanal kann in diesem Fall ein beliebiges technisches Kommunikationsmedium sein. In der Praxis kommen sehr häufig kabellose Internetverbindungen wie WLAN oder das Mobilfunknetz zum Einsatz. Es wären jedoch außerdem auch die ebenfalls kabellosen Technologien Bluetooth oder RFID bzw. NFC denkbar, sofern die Reichweite für den jeweiligen Anwendungsfall ausreichend ist.

Damit die ermittelten Positionen möglichst schnell zur Verfügung stehen und verarbeitet werden können, ist eine Kommunikation zwischen dem mobilen, zu ortenden Objekt und der Informationsverarbeitungseinheit in „Echtzeit“ nötig. Um herauszufinden, wie lange die Kommunikation dauern darf um dem Kriterium Echtzeit zu entsprechen, müssen wir den Begriff Echtzeit definieren und dabei auch die nach aktuell anerkannten Erkenntnissen aus der Physik betrachten. Demzufolge breiten sich elektromagnetische Wellen, zu dem auch das Licht gehört, mit der sogenannten Lichtgeschwindigkeit aus.

Signalleiter	**Verkürzungsfaktor (ca.)**
Offene Bandleitung (Wikipedia, 2016 b)	95 - 99 %
Koaxialkabel (ITWissen, o.J.)	77 %
Verdrillte Kabel (ITWissen, o.J.)	60 %
Lichtwellenleiter (ITWissen, o.J.)	67 %

Tabelle 3.4: Verkürzungsfaktoren verschiedener Signalleiter

Die Lichtgeschwindigkeit ist nach Definition im Vakuum mit 299.792.458 Meter pro Sekunde am größten und nimmt abhängig vom vorhandenen Leitermaterial,

in dem sich das Licht ausbreitet, ab. Für die Ausbreitung von Signalen wird ebenfalls die Lichtgeschwindigkeit angenommen, ebenfalls abhängig vom Leitermaterial und bei Lichtwellenleitern sogar von der Form der Leitung. (Wikibooks, 2015). Ausgedrückt wir diese Abhängigkeit durch den Verkürzungsfaktor (VKF bzw. NVP vom engl. Nominal Velocity of Propagation) im Vergleich zur Lichtgeschwindigkeit im Vakuum, welcher somit eine Kennzahl für Leitungen darstellt. (Wikipedia, 2016 b)

Der Verkürzungsfaktor wird durch verschiedene Einflüsse hervorgerufen, wie zum Beispiel der Dispersion bei Lichtwellenleitern oder Kapazität und Induktion bei metallischen Kabeln. (ITWissen, o.J.). Bei der Funkübertragung wird ebenfalls von der Lichtgeschwindigkeit für Signallaufzeiten ausgegangen, da sich in diesem Fall die Signale in Form von Radiowellen ausbreiten. Hierbei wird die Laufzeit stark von der Beschaffenheit der Luft sowie weiterer atmosphärischer Einflüsse und Störungen abgebremst wird. (Wikibooks, 2010)

„In real-time computing the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced“ (Stankovic, 1988)

Nach Stankovics Definition ist das Ergebnis der Echtzeitberechnung auch dann falsch, wenn es zwar logisch korrekt wäre, jedoch nicht rechtzeitig zur Verfügung steht. Demnach handelt es sich bei Echtzeit-Computer um Systeme, die speziell darauf ausgelegt sind ihre Berechnungen innerhalb eines definierten Zeitrahmens zu erfüllen, welcher genau definiert sein muss. Es geht also dabei in erster Linie nicht um Hochleistungsberechnungen, sondern darum, gezielt zeitliche Vorgaben einzuhalten. Hochleistungsberechnungen tragen jedoch positiv dazu bei und erhöhen den durchschnittlichen Datendurchsatz des Echtzeitsystems (Jonsson, 2015 S. 11). Man unterscheidet zwischen weiche und harte Echtzeitsysteme, gemessen an den Auswirkungen, die mit einer Zeitüberschreitung einhergeht. Demnach werden Überschreitungen, die fatale Folgen, wie weitreichende ökonomische Schäden oder Gefährdung von Menschenleben mit sich bringen, den harten Echtzeitsystemen zugeordnet. Dazu zählen zum Beispiel

Raketensteuerungen, Autopiloten und automatische Fahrzeugsteuerungen (Buckl, 2013 S. 26). Weiche Echtzeitsysteme wiederum verschlechtern die vom Benutzer wahrgenommene Qualität der Anwendung, haben jedoch keine weitreichenden Auswirkungen auf die Umwelt oder Gefährdung von Menschenleben. Zu den weichen Echtzeitsystemen zählen Anwendungen wie Computerspiele, Video-Streaming oder Virtual Reality (Jonsson, 2015 S. 13).

Da das zu entwickelnde Konzept eine verteilte Anwendung darstellt, deren Mobilteile in Form von Smartphones über eine Schnittstelle zum Mobilfunknetzwerk verfügen, wird die Echtzeitkommunikation auf Basis vom TCP/IP-Protokoll umgesetzt werden. TCP (Transmission Control Protocol) ist in der RFC 793 (Postel, 1981) definiert und neben dem UDP (User Datagram Protocol), definiert in RFC 768 (Postel, 1980) ein wichtiger Bestandteil der Internetprotokollfamilie auf Ebene der Transportschicht nach dem OSI-Referenzmodell sowie dem TCP/IP-Referenzmodell, welches als vierschichtiges Modell eine Vereinfachung des siebenschichtigen OSI-Referenzmodells darstellt. (Holtkamp, 2002 S. 14). Beide Protokolle beschreiben die Kommunikation auf Basis des Internet Protokolls, sie unterscheiden sich jedoch in der Art der Antwortquittierung. Während beim UDP sowohl die Übermittlung als auch das Eintreffen in der gesendeten Reihenfolge von Datenpaketen vom Sender zum Empfänger nicht garantiert ist, existiert beim TCP ein Verfahren, bei dem eine Übermittlungsbestätigung vom Empfänger zum Sender übertragen wird, sobald das Datenpaket empfangen wurde. Für den Fall, dass Pakete auf dem Weg zum Empfänger verloren gehen, werden diese erneut gesendet.

Üblicherweise findet die Kommunikation im Internet nach dem Client-Server-Interaktionsmodell statt. Ein Server nimmt Anforderungen entgegen, verarbeitet sie und liefert anschließend das Ergebnis zurück. Demnach ist ein Server jedes Programm, welches einen bestimmten Dienst (englisch service) anbietet. Ein laufendes Programm wird zu einem Client, wenn es Anforderungen an einen Server stellt und auf dessen Antwort wartet (Comer, 2011 S. 387).

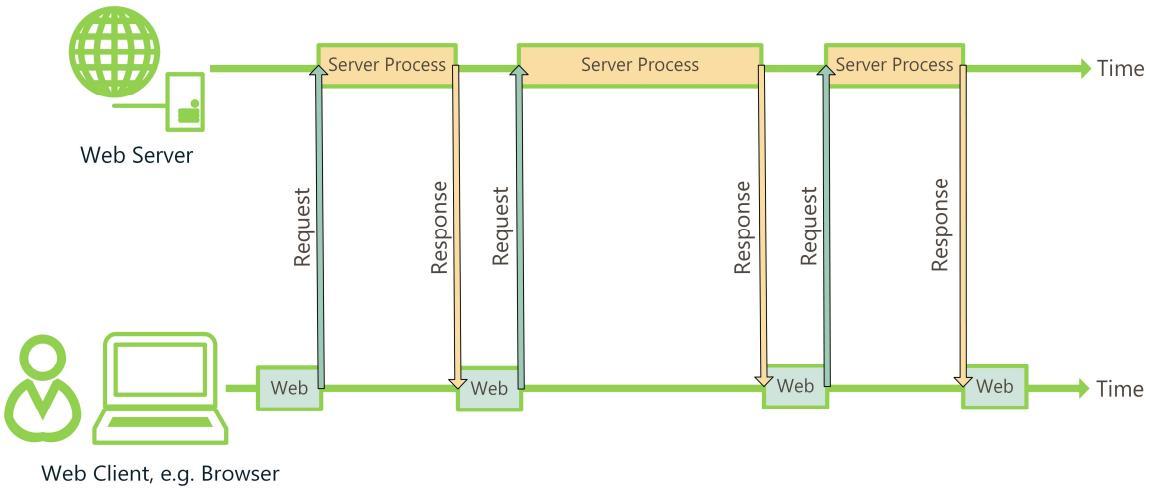

Abbildung 2.6: Server-Client-Interaktionsmodell

In Echtzeitanwendungen ist dieses Vorgehen jedoch nicht immer sinnvoll, denn je nach Anwendungsfall kann es sein, dass Informationen direkt nach ihrem Eintreffen verarbeitet und zum Client übertragen werden sollen. Um dieses Vorhaben zu realisieren, gibt es verschiedenste Konzepte. Generell unterscheidet man hierbei zwischen Polling und Pushing. Beim Polling wiederholt der Client in zyklischen Abständen immer wieder seine Anfragen an den Server um zu prüfen, ob neue Ergebnisse vorhanden sind.

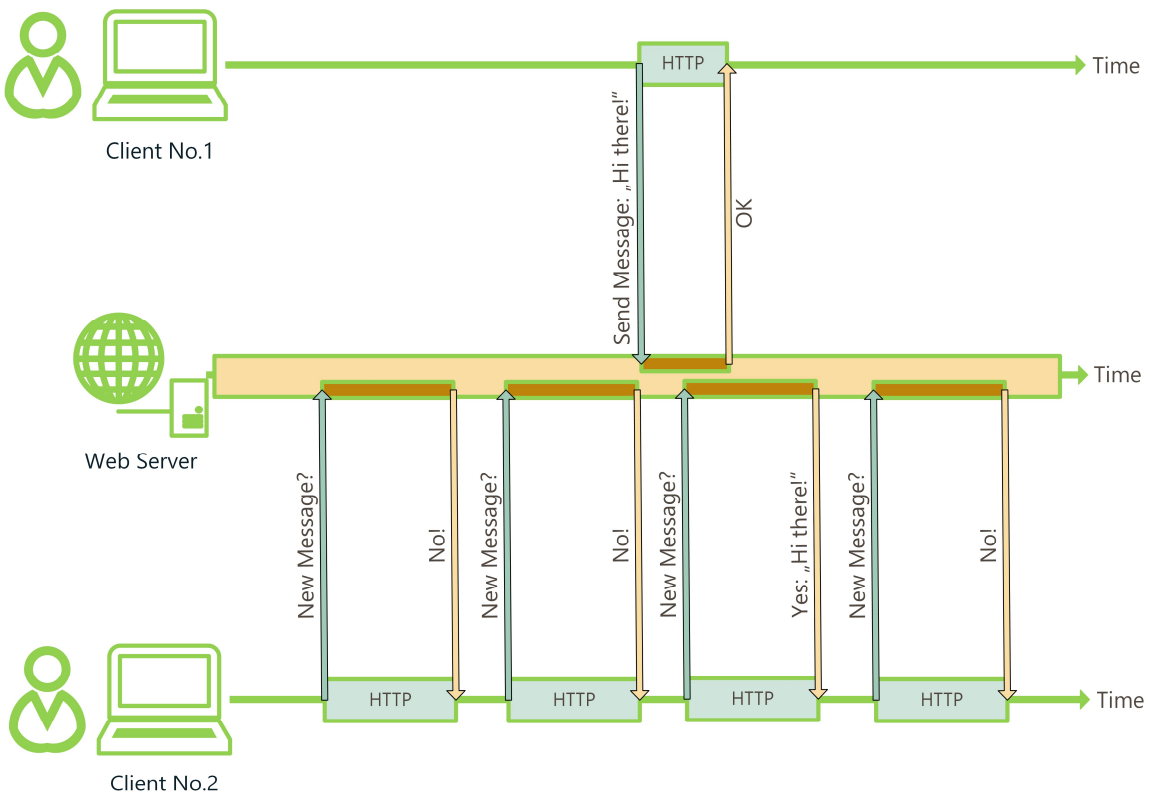

Abbildung 2.7: Polling-Verfahren

Der Server bestätigt jede Anfrage entweder mit dem neuen Ergebnis, oder mit der Info, dass es keine neuen Ergebnisse gibt. Beim Pushing wiederum scheint es so, als würde der Server entgegengesetzt handeln und erst dann eine Verbindung mit der Clientanwendung aufbauen, wenn neue Ergebnisse für die Clientanwendung vorhanden sind. Dies ist jedoch sowohl nach Definition des Server-Client-Prinzips als auch auf Grund von technischen und sicherheitsrelevanten Aspekten nicht immer möglich, denn obwohl es Peer-To-Peer-Lösungen gibt, die gleichzeitig sowohl Server- als auch Clientanwendung sind, sind in der Praxis Clients häufig nicht direkt im Netz erreichbar, sondern befinden sich hinter Firewalls, NATs oder Proxies. (Aguilar, 2014 S. 9)

Long Polling ist ein Push-Verfahren, das sehr ähnlich wie das bereits beschriebene Polling-Verfahren funktioniert, allerdings wurde das Vorgehen etwas abgeändert, um eine bessere Performanz durch höhrere Effizienz mit geringerer Latenz zu erreichen. Das ist möglich, indem der Client zwar wie beim normalen Polling-Verfahren regelmäßig beim Sever nach Updates fragt, allerdings vom Server

keine Antwort bekommt, solange nicht tatsächlich ein Update vorhanden ist. Demnach gibt es zwei Möglichkeiten, wie die Verbindung zwischen Server und Client beendet werden kann:

1. Der Server sendet eine Antwort, falls ein Update verfügbar ist.
2. Die Verbindung zwischen Server und Client wird aufgrund einer Zeitüberschreitung der Verbindung client- oder serverseitig getrennt.

In beiden Fällen würde der Client direkt im Anschluss eine neue Long-Polling Verbindung zum Server aufbauen, um erneut Updates zu empfangen.

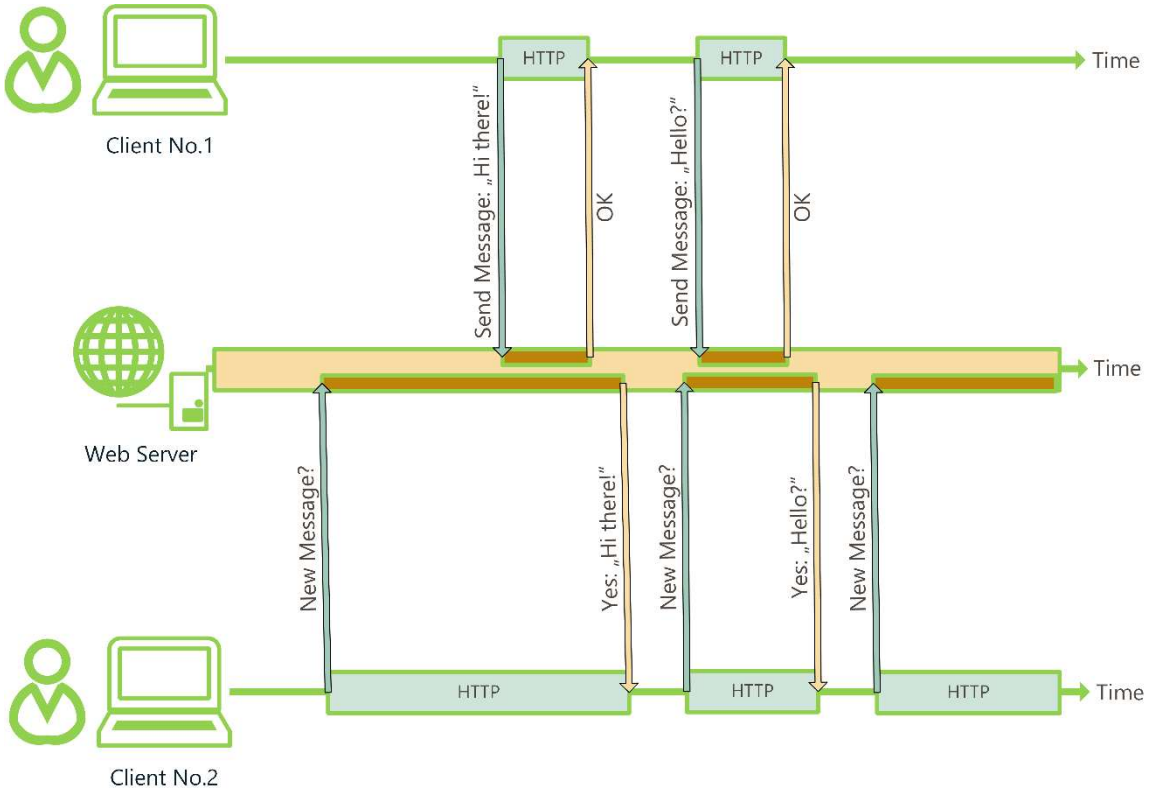

Abbildung 2.8: Long-Polling-Verfahren

Falls gleichzeitig Daten vom Client zum Server gesendet werden müssten, würde der Client eine neue HTTP-Verbindung zum Server aufbauen. Der eigentliche Vorteil bei diesem Verfahren ist die geringe Latenz. Die Übertragung in Echtzeit wird dadurch erreicht, dass bereits eine geöffnete Verbindung zwischen Server

und Client besteht, die direkt für die Antwortnachricht vom Server zum Client genutzt werden kann. Zudem wird gegenüber dem normalen Polling die Anzahl der Verbindungen reduziert und es besteht weiteres Optimierungspotential, indem man mehrere gleichzeitig eintreffende Updates gebündelt, also innerhalb einer einzigen Antwortnachricht, an den Client überträgt. Da beim Long-Polling lediglich die Möglichkeiten vom HTTP-Protokoll genutzt werden, ist kein zusätzliches Framework erforderlich. Daher sollte jeder HTTP-Client in der Lage sein, dieses Push-Verfahren zu implementieren (Aguilar, 2014 S. 13-14)

Server-Sent Events, auch bekannt als „Event Source API“ ist ein vom W3 Konsortium verabschiedeter Standard, der Bestandteil von HTML 5 ist. Die Funktionsweise ist ähnlich wie beim Long-Polling. Der Client stellt eine Anfrage an den Server, diese wird so lange offengehalten, bis der Server ein Update an den Client senden kann. Beim Senden der Antwortnachricht an den Client wird allerdings die Verbindung mit dem Server nicht geschlossen, sondern weiterhin offengehalten. Das ist möglich, weil die Antwort ähnlich wie in einem Live-Video-Stream in Form eines scheinbar endlosen Datenstroms gesendet wird, der jedoch nur dann Updates enthält, wenn diese tatsächlich eingetroffen sind.

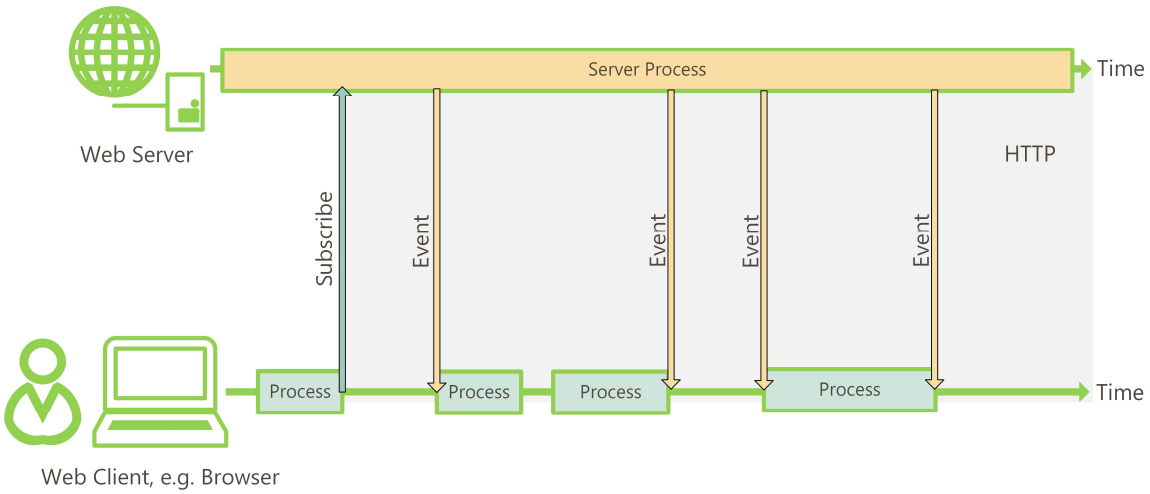

Abbildung 2.9: Server-Sent Events-Verfahren

Die Antwortnachricht vom Server enthält den Content-Type „text/event-stream“, damit der Client erkennen kann, dass es sich bei der Nachricht um Server-Sent Events handelt. Aktuell unterstützen sehr viele Browser (abgesehen vom Internet

Explorer und einige mobile Webbrowser) diesen Content-Type und somit dieses Push-Verfahren, da es ebenfalls auf dem HTTP-Standard basiert und keiner besonderen Implementierung bedarf. Zusätzlich zum Client muss jedoch die vorhandene Netzwerk-Infrastruktur (Proxies, Firewalls, etc.) in der Lage sein, diese Nachrichten korrekt zu interpretieren, um diesen Standard zu unterstützen, da Nachrichten aufgrund der Übertragung als Datenstrom nicht wie bei traditionellen HTTP-Nachrichten gepuffert übertragen werden und keine Verbindungsabbrüche aufgrund von Zeitüberschreitungen entstehen dürfen. Hervorzuheben bei diesem Verfahren ist es, dass es sich nur um eine unidirektionale Verbindung vom Server zum Client handelt. Für die Kommunikation vom Client zum Server müsste eine zweite HTTP-Verbindung aufgebaut werden. (Aguilar, 2014 S. 11-12)

Ein wiederum anderes Push-Verfahren stellt der sogenannte „Forever Frame“ dar. Er wird in eine Anwendung auf Basis eines <IFRAME>-Tags innerhalb des HTML-Markups realisiert und funktioniert ähnlich, wie die bereits erwähnten Server-Sent Events. iFrames werden hauptsächlich im Internet Browser ihre Anwendung finden, da sie im HTML definiert und vom Browser interpretiert werden. Der iFrame baut eine permanente Verbindung (daher der Begriff „Forever“) zur definierten Ressource auf. Diese Verbindung wird genau wie beim vorherigen Verfahren offengehalten und ermöglicht so eine Interaktion zwischen Server und Client via HTML und JavaScript.

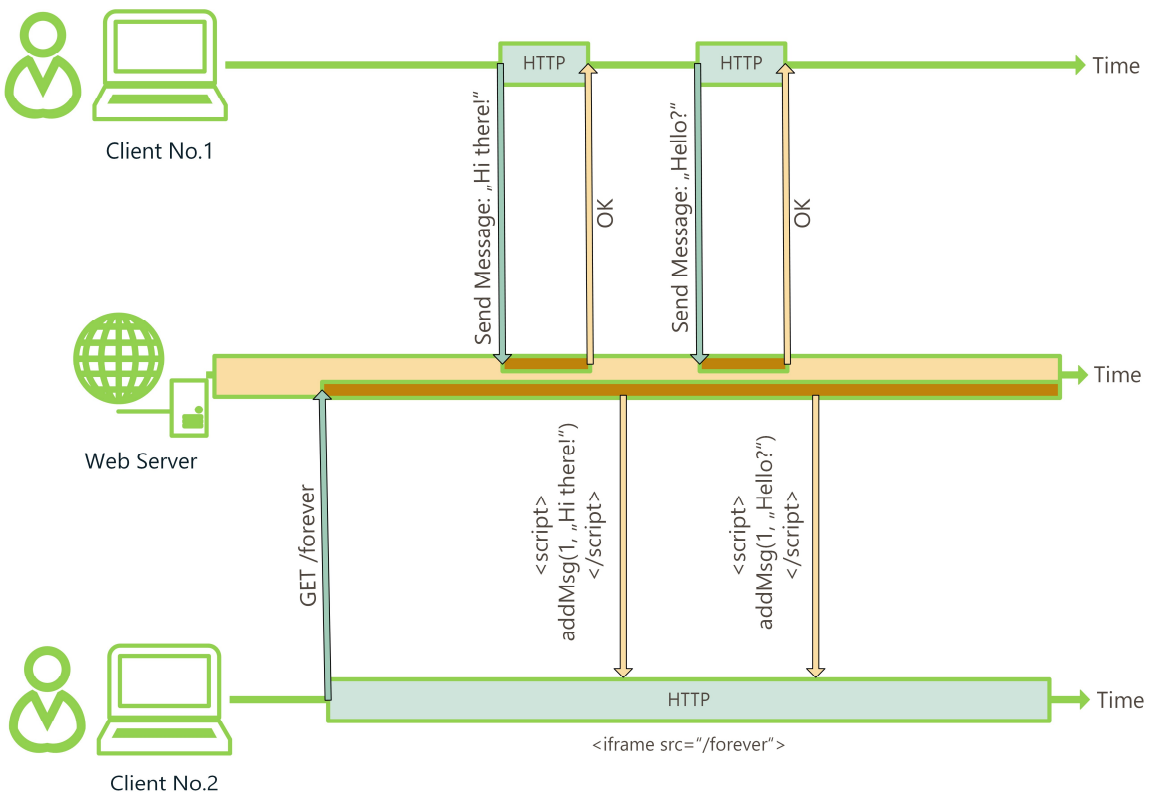

Abbildung 2.10: Forever-Frame-Verfahren

Zu beachten gibt es, dass auch hierbei die Netzwerkinfrastruktur das gepufferte übertragen des Datenstroms unterstützen muss und die Verbindungen nicht aufgrund von Zeitüberschreitungen unterbrechen darf. Darüber hinaus muss das Deallokieren des verwendeten Speichers in der Client-Anwendung berücksichtigt werden, da alle ankommenden Updates vom Server vorübergehen im Client gehalten werden und es so sehr schnell zu einer ungeplanten großen Auslastung des RAMs kommen kann. (Aguilar, 2014 S. 14-15)

Ein weiterer vom W3 Konsortium verabschiedeter Standard für Push-Verfahren stellt das Websocket-Protokoll dar, an dem ebenfalls die IETF gearbeitet hat. Er ist definiert in der RFC 6455 (Fette, et al., 2011) und beschreibt ein Protokoll, dass eine dauerhafte und bidirektionale Verbindung zwischen Client und Server ermöglicht. Da die Implementierung dieses Standards sehr komplex ist, ist sie nicht universell anwendbar. Allein der Fakt, dass die bidirektionale Verbindung zwischen Server und Client auf Basis vom HTTP-Protokoll realisiert wird, macht es zwingend erforderlich, dass Websockets zusätzlich von der vorhandenen

Netzwerkinfrastruktur unterstützt werden.

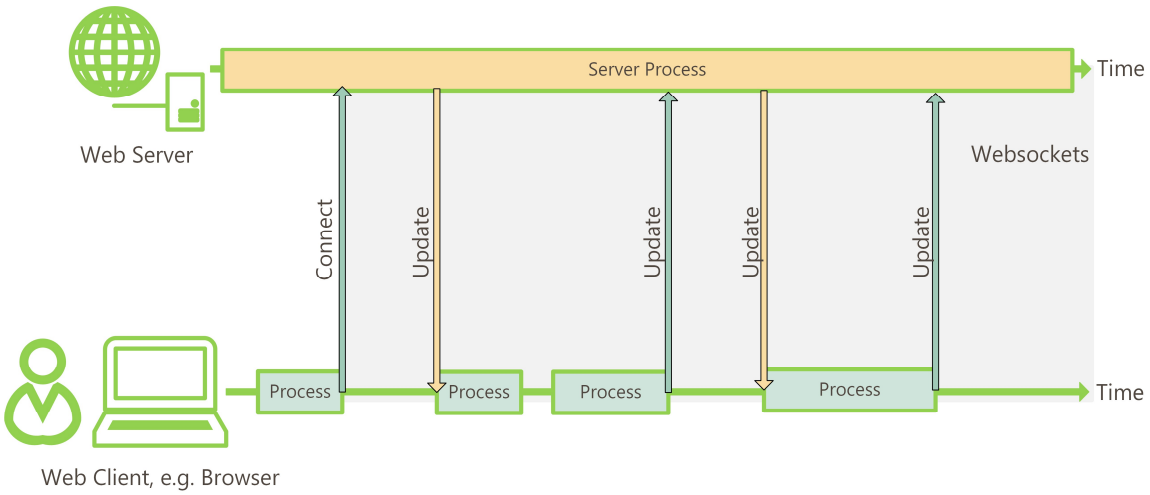

Abbildung 2.11: Kommunikation nach dem Websocket Protokoll

Aus Entwickler-Sicht können Websocket-Verbindungen sehr einfach aufgebaut werden, indem man eine URL zum Server, beginnend mit dem ws://-Schema angibt. Diese Technologie wird als die zukunftsträchtigste Technologie für Echtzeit-Push-Anwendungen gesehen. (Aguilar, 2014 S. 10-11)

Mit Ausnahme der Websockets basieren alle vorgestellten Verfahren auf dem HTTP-Protokoll, welches in der RFC 2616 spezifiziert ist (Fielding, et al., 1999). Da jedoch das ebenfalls im Jahr 1999 (Stanford-Clark, o.J.) erstmals erscheinende MQTT-Protokoll in der letzten Zeit gerade aufgrund der aufkommenden „Internet der Dinge“-Bewegung und der damit verbundenen Maschine-Zu-Maschine-Kommunikation sehr omnipräsent ist, wird es in diesem Zusammenhang ebenfalls kurz vorgestellt. Beim MQTT-Protokoll handelt es sich in der Version 3.1.1. (Banks, et al., 2015) um einen im November 2014 ernannten OASIS (Organization for the Advancement of Structured Information Standards) Standard (OASIS, 2014). Das Protokoll zeichnet sich durch das Subscribe-Publish-Pattern, ebenfalls basierend auf bidirektionalen TCP- bzw. Websocket-Verbindungen (Banks, et al., 2015), aus. Gerade weil es im Gegensatz zum HTTP-Protokoll aufgrund von möglichst kurz kodierten Befehlen weniger Overhead mit sich bringt und durch die bidirektionale Verbindung speziell für die Anwendung von Push-Verfahren konzipiert ist, ist es besonders für die Kommunikation zwischen

Mikrocontrollern in Sensornetzwerken geeignet. Beim MQTT existiert ein Broker, der als Mittelsmann agiert und alle eintreffenden Daten vom Sender (dem Publisher) zu jedem Empfänger (Subscriber) weiterleitet, der sich für ein bestimmtes Thema (Topic) registriert hat, um Updates dafür zu erhalten.

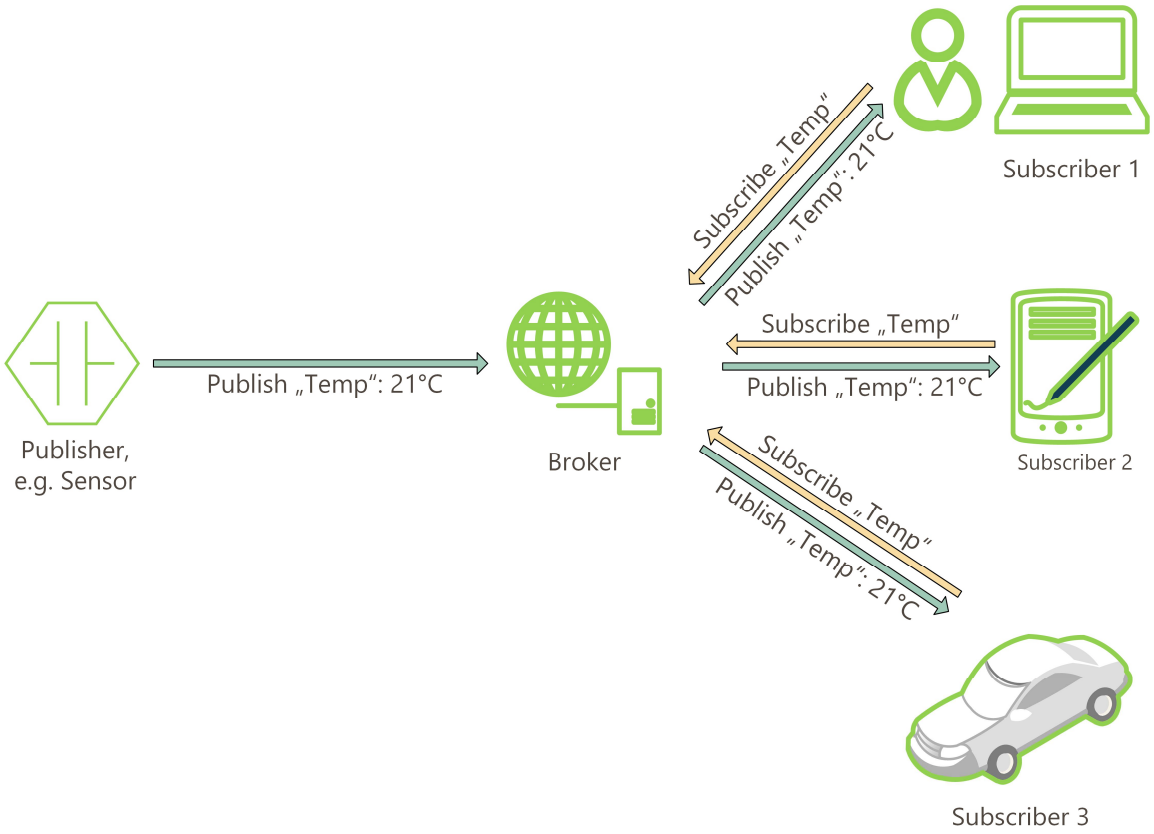

Abbildung 2.12: MQTT-Protokoll mit Subscibe-Publish-Verfahren

Spätestens nachdem Facebook seine neue Instant Messenger App namens „Messenger", ebenfalls basierend auf dem MQTT-Protokoll, angekündigt hat (Zhang, 2011), steht fest, dass dieses Protokoll nicht länger nur Kleinstcomputern und Mikrocontrollern vorenthalten bleibt, sondern auch immer mehr Einzug in dem Bereich der Smartphones, Notebooks oder Schreibtisch-Computern findet. Seitdem MQTT auf Basis vom Websocket-Protokoll arbeiten kann, steht sogar dem Einsatz im Webbrowser nichts mehr im Wege, sofern dies vom verwendeten Browser unterstützt wird.

3 Konzept

Dieses Kapitel beschäftigt sich ausschließlich mit der theoretischen Konzeption auf der Grundlage des in Kapitel zwei ermittelten Technologieansatzes sowie eines in diesem Kapitel benannten Beispielszenarios. Zunächst wird analysiert, ob und wie die Entwicklung überhaupt möglich ist und welche Anforderungen ein Prototyp dabei erfüllen müsste. Anschließend werden die Mittel und Wege erarbeitet, mit denen ein Prototyp umgesetzt werden kann.

3.1 Anwendungsbereich

Die Einsatzleitung einer Feuerwehr unterteilt sich neben dem Einsatzleiter klassischerweise in vier Sachgebiete. Dazu zählen Personal und Innerer Dienst (S1), Lage (S2), Einsatz (S3) und Versorgung (S4). Jedes dieser Sachgebiete ist dabei für die Erfüllung eigener Aufgaben verantwortlich. Bei Bedarf können weitere Sachgebiete eingerichtet werden, insbesondere für Presse- und Medienarbeit (S5) und Information- und Kommunikationswesen (S6). Je nach aktueller Schadenslage können noch Fachberater und Verbindungspersonen hinzugezogen werden (Feuerwehr-Dienstvorschrift 100, 1999 S. 16-17).

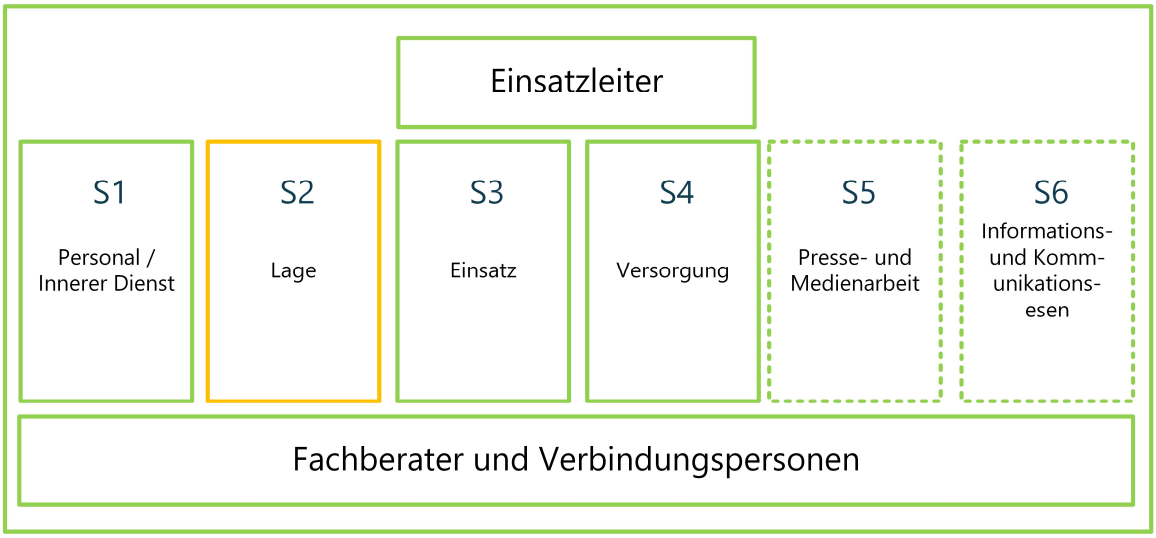

Abbildung 3.13: Zusammensetzung einer Einsatzleitung

Das zu entwickelnde Konzept gliedert sich in das Sachgebiet Lage (S2) ein, da es der Einsatzleitung einen dreidimensionalen Überblick über die Einsatzlage geben soll.

3.2 Beispielszenario: Einsatz am Gendarmenmakt

Um sich die Anwendung des Konzptes besser vorstellen zu können, wird folgendes (frei erfundenes) Einsatzszenario als Beispiel gegeben:

Im Konzerthaus Berlin kommt es während einer Aufführung des Berliner Orchesters im Großen Saal zu einem Feuer mit sehr starker Rauchentwicklung. Aufgrund der zeitgleich stattfindenden Ausstellung im Deutschen Dom und einer Trauung im Französichen Dom befinden sich sehr viele Zivilisten auf und um den Gendarmenmarkt. Die Feuerwehr rückt an und und startet mit der Evakuuierung des Konzerthauses Berlin und der Brandbekämpfung. Zusätzlich rückt Polizei an und sperrt die umliegenden Straßen für den öffentlichen Verkehr. Da viele Zuschauer des Konzertes eine Rauchvergiftung erleiden, wird entschieden vom Rettungsdienst Zelte für die Erstversorgung in einer Nebenstraße errichten zu lassen.

Ein Einsatzleiter-Stab wird eingerichtet. Die Einsatzleitung befindet sich dahernicht vor Ort, sondern in der Leitstelle, hat jedoch die Aufgabe durch gezielte Befehlsgebung alle Hauptaktionen zu koordinieren. Um das zu ermöglichen, werden die Einsatzkräfte und mobilen Einsatzgeräte, wie zum Beispiel mobile Straßensperren, mit Transpondern ausgestattet, die in der Lage sind, ihre aktuelle Position zur Leitstelle zu übertragen. Die Einsatzleitung im Sachgebiet Lage (S2) hat mehrere HoloLenses zur Verfügung, um sich ein direktes Bild des Geschehens vor Ort machen zu können.

3.3 Rahmenbedingungen

Im Folgenden werden die technischen Möglichkeiten aufgrund des in Kapitel 3.2 beschriebenen Beispielszenarios, sowie die rechtlichen Rahmenbedingungen skizziert.

3.3.1 Technische Möglichkeiten

Im folgenden Kapitel werden die technischen Möglichkeiten betrachtet, um die Positionsdaten in einem dreidimensionalen Modell einer Stadt darzustellen.

3.3.1.1 Open Data 3D Stadtmodelle

Im Rahmen dieser Arbeit sollen die im Zusammenhang einer Open Data Initiative erstellten, frei zugänglichen 3D-Modelle von Städten verwendet werden, um diese in einem Mixed-Reality-Umfeld darzustellen. Um keine proprietären Formate in Bezug auf die Modelle zu schaffen und eine maximale Kompatibilität zu ermöglichen wird die Möglichkeit geschaffen, die Modelle beliebiger Orte, die bereits aufgrund der Open Data Initiative geschaffen wurden, einzubinden. Um das zu erreichen muss das Modell in einem einheitlichen Format vorliegen bzw. möglichst einfach in dieses Format überführt werden können. So stellt CityGML zum Beispiel ein offenes, standardisiertes Modell zum Austausch von Daten dreidimensionaler Städte und Landschaften in Form von XML dar (3D Geoinformation group at TU Delft, o. J.). Viele Städte bieten bereits solche 3D-Modelle an, dazu zählen zum Beispiel Dresden (Landeshauptstadt Dresden, 2017), Rotterdam (Rotterdam Open Data Store, o. J.). Hamburg (Transparenzportal Hamburg, 2014) und Berlin (Berlin Partner für Wirtschaft und Technologie GmbH, 2015 a).

Die beispielhafte Umsetzung in dieser Arbeit beschränkt sich auf das Land Berlin, kann aber bei Bedarf durch das Einbinden von Modellen von Orten anderer Bundesländer beliebig erweitert werden. Allerdings ist zu beachten, dass aufgrund des EU-Ansatzes die Zuständigkeiten bei den einzelnen Bundesländern liegt. Daher stehen verschiedenste Daten in unterschiedlicher Qualität, Datenformaten und Schnittstellen zur Verfügung.

3.3.1.2 Positionsbestimmung von Personen und Objekten

Um die Positionsbestimmung von Personen und Objekten im Rahmen dieser Arbeit zu simulieren, soll eine prototypische Anwendung erstellt werden, die die Ortung der Person ermöglicht, der das Smartphone gehört bzw. des Objektes, in dem das Smartphone angebracht ist. Diese Anwendung soll auf die gängigen Smart-Phone-Plattformen iOS (Marktanteil 2016 bei 21,6%), Android (Marktanteil 2016 bei 75,6%) und Windows (Marktanteil 2016 bei 2,5%) lauffähig sein, um eine möglichst breite Masse an Smartphones zu unterstützen und damit für möglichst viele Einsatzkräfte nutzbar zu sein. Ein großer Vorteil ist die Verbreitung von Smartphones (Kantar, 2017). Im Jahr 2016 lag die Anzahl der Smartphone-Nutzer in Deutschland bei 76% der Einwohner, Tendenz steigend (Bitkom, 2016 a). Der Nutzerkreis hält sich bei 14- bis 64-jährigen Menschen mit leicht abfallender Tendenz zwischen 95% und 88% in den jeweiligen Altersgruppen. Lediglich in der Altersgruppe von 65 und mehr verwenden nur 27% der Deutschen Smartphones (Bitkom, 2016 b).

Ein weiterer, besonderer Vorteil ist, dass die meisten aktuell erhältlichen Geräte bereits über Assisted GPS Sensoren verfügen, die in der Lage sind, ein oder häufig sogar mehrere Satellitennavigationssysteme gleichzeitig, wie NAVSTAR GPS, GLONASS, Galileo oder Beidou, zu unterstützen. Darüber hinaus werden aufgrund der vorhandenen Schnittstellen ins Mobilfunknetz und weiterer Funkstandards wie NFC, Bluetooth und WLAN außerdem viele ortsbasierte Dienste zur Lokalisierung unterstützt, wie zum Beispiel über eine IP-Adresse oder die Ortung über bekannte WLAN-Netze in der Umgebung, wie Skyhook sie anbietet. (Skyhook, o.J.)

Als Softwareentwickler hat man für jede dieser Plattformen keinen wirklichen Einfluss darauf, welche der hier vorgestellten Technologien auf dem jeweiligen Gerät zum Einsatz kommen, um die aktuelle Position zu ermitteln. Dies ist bedingt durch die Hardware-Spezifikation der Geräte. Für jede dieser Plattformen gibt es unterschiedliche Geräte, bei denen GPS- bzw. Mobilfunk-Module verbaut sein können, oder nicht. Um die Positionsbestimmung für den Softwareentwickler zu vereinfachen, kümmert sich das jeweilige Betriebssystem (OSI-Schicht 5 und 6)

um die Standortermittlung des Gerätes. Der ermittelte Standort kann plattformspezifisch durch eine Programmierschnittstelle angerufen werden.

Plattform	Framework
iOS	Core Location (Apple, o.J. b)
Android	Android.Location.Location (Android, o.J.)
Universal Windows Platform	Windows.Devices.Geolocation (Microsoft, 2017)

Tabelle 3.5: Übersicht über die Frameworks zur Standortermittlung

3.3.1.3 Echtzeitübertragung von Positionsinformationen

Im Kapitel 2.6 wurden verschiedene technische Möglichkeiten gegenübergestellt, die eine Übertragung in Echtzeit ermöglichen. Da jedes dieser vorgestellten Verfahren seine Vor- und Nachteile sowohl in Bezug auf Kompatibilität zwischen Server, Client und dazwischenliegender Netzwerkinfrastruktur als auch Performanz (gemessen an der Anzahl der erstellten Verbindungen zwischen Server und Client) mit sich bringen ist es sinnvoll eine Möglichkeit zu finden, die es erlaubt, für jede Kombination aus Server und Client das bestmögliche, beiderseitig unterstützte Verfahren zu ermitteln und anschließend automatisch zu verwenden.

Die Entwickler des SignalR Frameworks haben sich genau dieses Ziel gesetzt und eine Komponente geschaffen, die genau dies ermöglicht. SignalR ist ein Open Source Projekt, unter Apache 2.0 Lizenz, von David Fowler und Damian Edwards (Fowler, et al., 2017). Beide sind Mitglieder des ASP.NET Teams bei Microsoft. SignalR wird aktuell selbst bei vielen Echtzeitanwendungen eingesetzt, wie zum Beispiel den Kollaborationsfunktionen der diversen Microsoft Office Produkte, OneDrive oder SharePoint (Aguilar, 2014 S. 17). Darüber hinaus gibt es viele Open Source Echtzeit-Anwendungen wie den Chat JabbR (Fowler, et al., o. J.) und das Browserspiel ShootR (Mullen, o. J.).

SignalR kapselt aus Sicht des Entwicklers die technischen Details zum Verbindungsaufbau und stellt eine ihm virtuelle, permanente und bidirektionale Verbindung zur Verfügung. Um das zu erreichen, wird versucht, das effizienteste Echtzeitverfahren einzusetzen. Es werden so lange alternative Verfahren ausgehandelt, bis eins gefunden wird was sowohl vom Server als auch vom Client unterstützt wird (auch bekannt als Negotiation). Darüber hinaus wird dem Entwickler unabhängig vom zugrundeliegenden Verfahren ein einheitliches Programmiermodell zur Verfügung gestellt und somit eine Abstraktion erreicht, damit er sich nicht auf die technischen Besonderheiten konzentrieren muss.

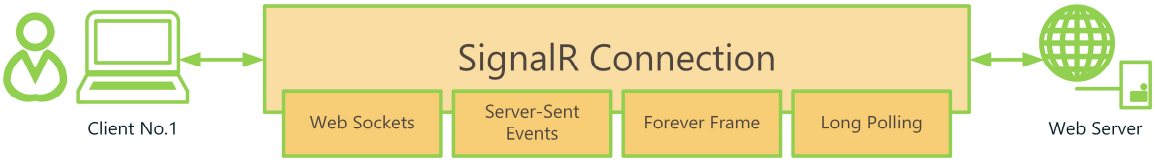

Abbildung 3.14: SignalR - Virtuelle Verbindung

Somit macht es für den Entwickler keinen Unterschied, ob eine Verbindung zwischen Client und Server mittels Web Sockets, Server-Sent Events oder Long Polling aufgebaut wird. (Aguilar, 2014 S. 18-19)

Darüber hinaus bietet die Komponente viele nützliche Möglichkeiten um auf Ereignisse zu reagieren, so ist es zum Beispiel möglich auf neue Clients, Trennungen, Verbindungsabbrüche oder verschiedene Verbindungsgeschwindigkeiten (große Latenzen) zu reagieren. Darüber hinaus kann eine Nachricht an alle verbundenen Clients (Broadcast) oder auch nur an eine Gruppe von ihnen gesendet werden. (Aguilar, 2014 S. 18)

SignalR besteht aus einer Server- und einer Client-Komponente. Die Server-Komponente basiert auf dem ASP .NET Framework, welche dank der OWIN Spezifikation (Open Web Interface für .NET) (OWIN Working Group, 2012), die als Schicht zwischen Webserver und Webanwendung entwickelt wurde, auf beliebigen Hard- und Software-Plattformen ausgeführt werden kann. Darunter zählen neben dem klassischen .NET Framework für Windows auch Mono und das .NET Core Framework (hier speziell unter ASP .NET Core), welche eine Lauffähigkeit auf Linux und OSX ermöglichen.

Die SingalR Client-Komponente wird für diverse Plattformen in verschiedenen Programmiersprachen bereitgestellt. Sie existiert derzeit für verschiedenste Browser als JQuery Plugin (Fowler, 2014) in JavaScript, für verschiedene Versionen und Derivate des .NET Frameworks in C# (Fowler, et al., 2017), aber auch in Java (Zaidenvoren, o. J.) und vielen anderen Programmiersprachen. Auf Grund des Open Source Charakters ist es sogar möglich, durch eigene Implementierungen, weitere Client-Plattformen zu unterstützen.

3.3.2 Rechtliche Möglichkeiten

Da die im Rahmen der Anwendung erfassten mobilen Positionsdaten von beweglichen Objekten und Personen Einsatzkräften von Behörden und Organisationen mit Sicherheitsaufgaben zugeordnet sind, unterliegen sie nicht nur dem Bundesdatenschutzgesetz (BDSG) (Bundesministerium der Justiz und für Verbraucherschutz, 2015), sondern auch den jeweiligen Gesetzen der Länder, in der die Anwendung zum Einsatz kommt. Obwohl die Erfassung der Positionen zur Gefahrenverhütung und zur Gefahrenabwehr geschieht, müssen trotzdem die geltenden Bestimmungen eingehalten werden. Für das Land Berlin existieren verschiedene Rechtsgrundlagen für den Allgemeinen Bereich, den Rettungsdienst und den Katastrophenschutz. Dazu zählen unter anderem

- das Allgemeine Gesetz zum Schutz der Öffentlichen Sicherheit und Ordnung in Berlin (ASOG) (Land Berlin, 2006),
- das Gesetz über die Feuerwehren im Land Berlin (FwG) (Land Berlin, 2003) oder auch
- das Gesetz über die Gefahrenabwehr bei Katastrophen (KatSG) (Land Berlin, 1999).

Für den Einsatz in Berlin bedeutet das, dass die Erfassung und Verarbeitung der Positionen oder Objekten von Einsatzkräften von Behörden und Organisationen mit Sicherheitsaufgaben gewissen Einschränkungen, Rechten und Pflichten unterliegen könnte, die bei der Verwendung des Systems je nach Einsatzort genauer beachtet werden müssen. In erster Linie werden Personen nicht über ihren echten Namen, sondern über einen frei definierbaren eindeutigen Schlüssel, zum

Beispiel die Personalnummer mit vorangestellter Bundeslandkennung, geortet. Dieser Schlüssel kann sich einsatzübergreifend ändern. Für die Einsatzleitung ist in erster Linie nicht wichtig zu erfahren welche private Person sich hinter der Einsatzkraft verbirgt, sondern vielmehr welcher taktischen Einheit sie angehört und wie lange sie bereits im Einsatz ist.

3.3.3 Gesellschaftliche Akzeptanz

Um die gesellschaftliche Akzeptanz für das Konzept zu ermitteln, muss man diese Akzeptanz in verschiedene Bereiche unterteilen. Zum einen muss ermittelt werden, inwiefern Personen bereit sind die Ortungsfunktion der eigenen mobilen Geräte zu aktivieren, damit eine Ortung der Einsatzkräfte mithilfe von Assisted-GPS überhaupt möglich ist. Darüber hinaus muss ermittelt werden, wie gut die vielen auf dem Kopf tragbaren Virtual Reality und Augmented Reality Geräte akzeptiert werden, die in den vergangenen Jahren auf den Markt gekommen sind um Rückschlüsse auf das bisher einzige, neu erhältliche Mixed Reality Gerät zu ziehen. Zu guter Letzt muss die Akzeptanz von Open Data innerhalb der Staatsführung beleuchtet werden um herauszufinden wie sehr Bürgerinnen und Bürger der Open Data Initiative vertrauen und vor allem, wie brauchbar die aus der Initiative erstellten Daten eigentlich sind.

Ein weiterer zu beleuchtender Aspekt bezüglich der Akzeptanz und Nutzung genannter neuer Technologien ist der Grad der Technik-Affinität der betreffenden Anwender. Die Anwender müssen in verschiedene Personengruppen mit unterschiedlichen demographischen sowie psychologischen Eigenschaften und mit unterschiedlich stark ausgeprägter Technik-Affinität unterteilt werden. Dies Unterschiedlichkeit der Anwender muss bei der Einführung und beim Einsatz der Gerätschaften und Applikationen berücksichtigt werden. Die Technik-Affinität ist derzeit mit einem höheren Alter der Anwender tendenziell geringer ausgeprägt als die der jüngeren. Dies wird sich jedoch im Laufe der kommenden Jahre wahrscheinlich ändern, da die derzeit junge, vermeintlich Technik-affinere Anwendergruppe, älter wird und somit der Grad der Technik-Affinität in den nächsten Jahren deutlich höher sein wird im Vergleich zum jetzigen Zeitpunkt. Es ist davon auszugehen, das die AR-Technologie, ebenso wie derzeit das Internet, zukünftig

und selbstverständlich zum Alltag gehören wird.

Dies soll einerseits erreicht werden durch immer leistungsfähigere und transportable Geräte, die untereinander vernetzt sind und Daten ständig und überall zum Abruf bereit halten, andererseits auch durch den verstärkten Einsatz der Technik in allen Bereichen des täglichen Lebens. (Schwiertz, 2017)

3.3.3.1 Ortung, Positionsbestimmung und Ortsbasierte Dienste

Laut Umfragen besaßen im Jahr 2014 bereits 26,7 Millionen Menschen in Deutschland ein GPS-fähiges Mobiltelefon. Aus einem Vergleich zu den Vorjahren lässt sich eine steigende Tendenz erkennen (VuMA (Arbeitsgemeinschaft Verbrauchs- und Medienanalyse), 2014). Die steigende Tendenz lässt sich damit begründen, dass heutzutage Android (75,6%), iOS (21,6%) und Windows (2,5%) (Kantar, 2017) zu den weitverbreitetsten Betriebssystemen für Mobiltelefone im Smartphone-Bereich zählen und die meisten dieser Geräte von Werk aus mit entsprechender Hardware für GPS und Ortsbasierten Diensten ausgestattet sind. Zusätzlich ist erkennbar, dass immer mehr Menschen vom konventionellen Handy zum Smartphone wechseln und so davon auszugehen ist, dass bis zum Jahr 2019 voraussichtlich 83,1% aller Personen ein Smartphone besitzen und das regelmäßig benutzen. (eMarketer, 2015) Da so viele Menschen ohnehin ein Smartphone besitzen und es häufig bei sich tragen ist die Ortung über diese Geräte denkbar. Einsatzkräfte müssten somit nicht zusätzlich weitere Transponder zur Ortung mit sich tragen.

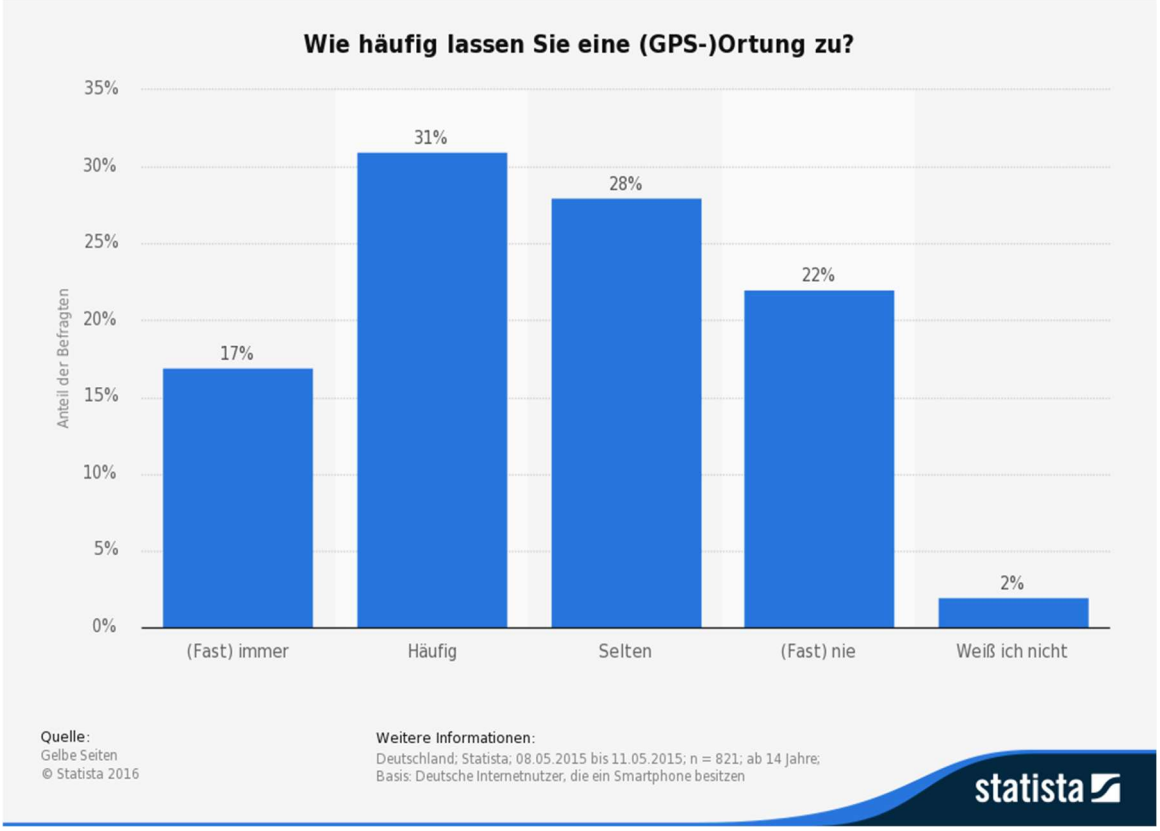

Abbildung 3.15: Umfrageergebnisse zur Häufigkeit der GPS-Ortung, (Horizont, 2015)

Weiterhin sind von den Smartphone-Besitzern mit einem GPS-fähigem Gerät annähernd die Hälfte bereit dazu, sich von Anwendungen „häufig“ bzw. „(fast) immer“ orten zu lassen. (Horizont, 2015) Das lässt darauf schließen, dass das im Rahmen des Konzepts zu entwickelnde Ortungsmodul als Teil einer Smartphone-Anwendung seine Akzeptanz bei der Einsatzkraft finden könnte. Darüber hinaus ist hervorzuheben, dass es sich hierbei um eine Ortung im Einsatzfall handelt, daher ist davon auszugehen, dass die Anwendungsbereitschaft in diesem Fall noch wesentlich höher ist, als die Bereitschaft einer Ortung im alltäglichen Leben, denn hierbei geht es um die Koordinierung von Sicherheitsaufgaben und unter Umständen auch um die Rettung von Menschenleben.

3.3.3.2 Virtual Reality und Headmounted Displays

Obwohl die Anzahl der derzeit erhältlichen Virtual Reality Geräte und Headmounted Displays sehr überschaubar ist, existieren doch sehr viele Prognosen über den Ab- und Umsatz in den kommenden Jahren. Demnach wird sich der jährliche Umsatz der Virtual Reality Branche im Consumer-Bereich von gerade mal 160

Millionen Euro bis zum Jahr 2020 voraussichtlich auf mehr als eine Milliarde Euro allein in Deutschland steigern (Deloitte, 2016). Im Geschäftsbereich ist nochmals mit einem ähnlichen Umsatz und einem ähnlichen Anstieg zu rechnen (Deloitte, et al., 2016). Umfrageergebnisse zeigen, dass gerade jüngere Anwender offen für diese neuen Technologien sind. Demnach können sich mehr als die Hälfte der befragten Internetnutzer im Alter von 16 bis 34 Jahren vorstellen zukünftig Virtual-Reality-Brillen für Computerspiele einzusetzen. Die Bereitschaft dazu nimmt jedoch bei älteren Menschen ab. Im Alter ab 55 und mehr ist immerhin noch einer von fünf dazu bereit ein solches Gerät für Spiele zu verwenden. (BIU, 2016). Hierbei muss jedoch kritisch hinterfragt werden wie viele Personen in dieser Altersklasse Computerspiele spielen.

Aus diesen Prognosen und Umfrageergebnissen lässt sich eine generelle Bereitschaft in der Gesellschaft für die Verwendung von Virtual Reality Brillen, die vom Anwender auf dem Kopf getragen werden, ableiten. Da es aktuell nur sehr wenige Augmented Reality Geräte gibt und die HoloLens derzeit das einzige frei erhältliche Gerät auf dem Markt ist, dass dem Anwender eine wirkliche Mixed-Reality-Erfahrung verspricht, existieren derzeit nur sehr wenige Prognosen und Umfragen zu dem Thema. Da Mixed Reality jedoch ein ähnliches Nutzererlebnis liefert, vermute ich, dass die ermittelten Prognosen und Umfrageergebnisse auch in diesem Fall Anwendung finden.

3.3.3.3 Open Governement

Obwohl sich viele Gemeinden innerhalb und außerhalb Deutschlands Open Government und in diesem Zusammenhang Open Data, als Begriff für frei zugängliche Verwaltungsdaten der öffentlichen Verwaltung, bereits schon seit längerem betreiben, scheinen vor allem in Deutschland viele Menschen nicht ausreichend über die aktuellen Möglichkeiten im Zusammenhang der Open Data Initiative informiert zu sein. Demnach wissen nur 24 Prozent der Deutschen Internetbenutzer über 18 Jahre überhaupt, dass Informationen in Form von Open Data durch Behörden bereitgestellt werden. Noch weniger machen davon gebraucht. Da allerdings über die Hälfte der Befragten äußert, in Zukunft aktiv solche Dienste nutzen zu wollen ist erkennbar, wieviel Potential darin steckt. (D21,

et al., 2016) Vielmehr lässt sich sogar vermuten, dass vielen der Befragten bis zum Zeitpunkt der Befragung überhaupt nicht bekannt ist, welche Dienste und Informationen überhaupt von Behörden und öffentlichen Einrichtungen für die Bevölkerung zur Verfügung gestellt werden.

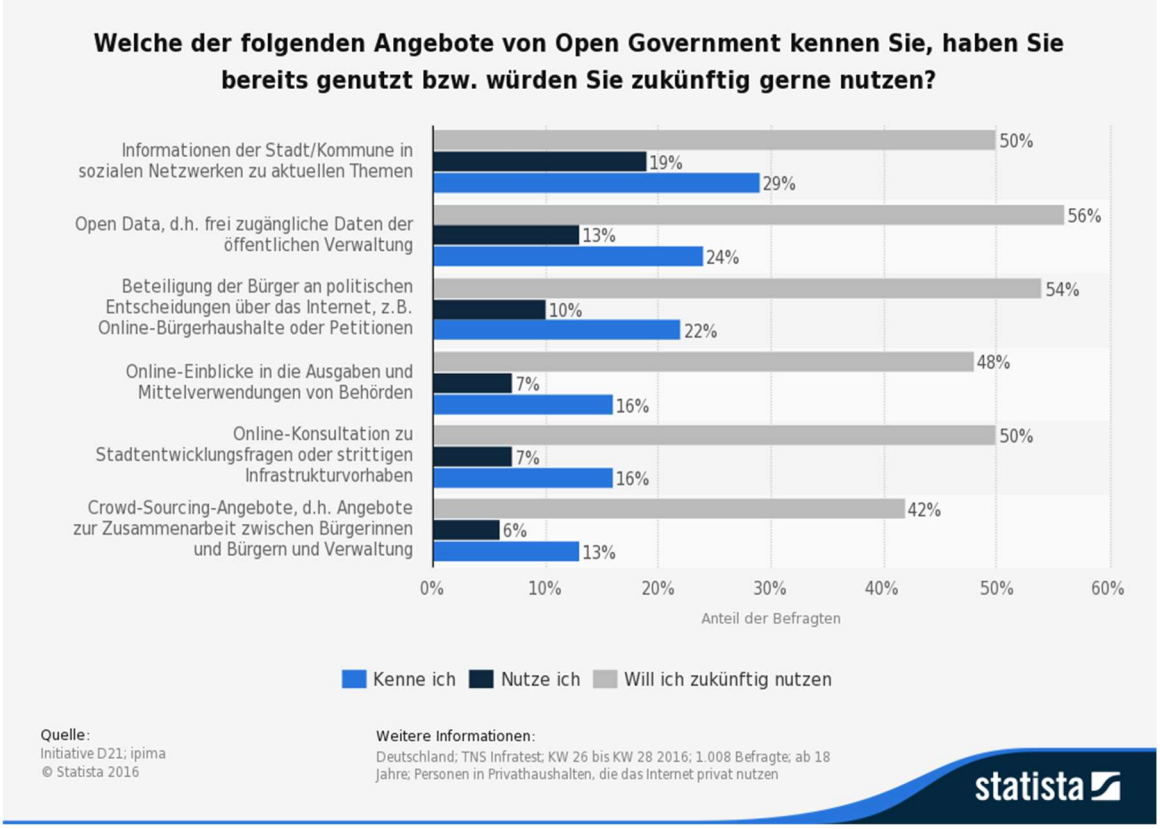

Abbildung 3.16: Umfrage zur Nutzung von Open Government

Diese Vermutung wird bestärkt, wenn man sich die Entwicklung des E-Government Development Index (EGDI) in Deutschland der vergangenen Jahre betrachtet. Der Index wird nach dem Angebot von E-Services, Telekommunikations-Infrastruktur sowie den Fähigkeiten der Menschen zur Nutzung von E-Services bemessen und demnach hält sich der Wert für Deutschland seit 2005 bis mit einigen Ausnahmen zwischen 2008 und 2010 immer um die 0,8, so auch im Jahr 2016 mit einem Wert von 0,82. (United Nations, 2016 b). Im Vergleich zu anderen Nationen belegt Deutschland nur den 15. Platz, wobei die Weltrangliste von Nationen wie dem Vereinigten Königreich (0,92), Australien (0,91) und Südkorea (0,89) angeführt wird. (United Nations, 2016 a). Neben der fehlenden Information äußerten viele der befragten Deutschen verschiedene Ängste im Bereich von

Datenschutz und Datensicherheit, die sie davon abhalten, Onlinedienste von Behörden zu nutzen. Dazu zählen vor allem die Angst vor Datendiebstahl (50%), Mangelnde Informationen darüber, was mit den Daten passiert (50%), Mangelnde Sicherheit bei der Datenübertragung (48%) und die Befürchtung im Hinblick auf den „Gläsernen Bürger“, was aus dem Zusammenführen verschiedener Daten zu einer zentralen Datenbank mit sich bringen könnte. (48%). (Initiative, et al., 2016).

Für das zu entwickelnde Konzept lässt sich entnehmen, dass bereits jetzt schon sehr viele Daten öffentlich zugänglich sind und anzunehmen ist, dass sich die Anzahl der öffentlich zur Verfügung stehenden Datensätze in Zukunft global deutlich erhöhen wird, allein schon da das Interesse der Bürger vorhanden ist. Gerade dreidimensionale Stadtmodelle werden konkret in diesem Projekt aus dem Open Data Bereich verwendet. Die Daten liegen allerdings in sehr unterschiedlichen Qualitäten vor. Es existiert jedoch kein einheitliches Datenformat und keine einheitliche Schnittstelle zum Abrufen der Daten. (vergl.3.3.1.1)

3.4 Anforderungen

In den nachfolgenden Absätzen werden Anforderungen aus den Rahmenbedingungen definiert und anschließend User Stories gebildet. User Stories beschreiben die Funktionen einer Anwendung aus der Sicht der Anwender, wobei die Anwender an sich in verschiedene Personas zu unterteilen sind.

Im Anschluss werden sowohl die funktionalen als auch die nichtfunktionalen Anforderungen definiert, die sich aus den Rahmenbedingungen und den User Stories ergeben. Die funktionalen Anforderungen beschreiben die Funktionen, die das Projekt zu erfüllen hat, während nichtfunktionale Anforderungen die Qualität festlegen, in der die geforderte Funktionalität erbracht werden soll.

3.4.1 Anforderungen aus den Rahmenbedingungen

Damit das Konzept korrekt angewendet werden kann, sind zunächst alle Personas zu definieren. Zum einen gibt es die Personen der Einsatzleitung (Speziell des Sachgebietes Lage S2), die jeweils eine HoloLens auf dem Kopf tragen werden, wo die Mixed Reality Anwendung ausgeführt wird (im Folgenden „HoloCommander“ genannt). Dadurch bekommen Sie die die Möglichkeit sich die Umgebung des Einsatzortes in Form vom dreidimensionalen Modell direkt in der Einsatzzentrale anzuzeigen um somit einen besseren Überblick über den Ort zu erhalten. Darüber hinaus werden Einsatzkräfte und bewegliche Objekte, die sich am Einsatzort befinden, für die Einsatzleitung sichtbar, innerhalb des Modells vom Einsatzort dargestellt, damit Sie sich jederzeit ein Bild des aktuellen Geschehens machen kann. Demnach stellen Einsatzkräfte inklusive der beweglichen Objekte die zweite Persona „Einsatzkraft“. Um das Anzeigen der Standorte im Modell zu ermöglichen, ist es notwendig, dass die Einsatzkräfte und bewegliche Objekte regelmäßig Informationen über ihren aktuellen reellen Standort zur Verfügung stellen. Diese Aufgabe soll eine Smartphone Anwendung automatisch im Hintergrund erfüllen (im Folgenden „HereIAm“ genannt) indem sie immer eine Nachricht mit den aktuellen Positionsinformationen versendet sobald festgestellt wird, dass sich die Position verändert hat. Damit keine Direkte Verbindung zwischen Smartphone App und HoloLens erforderlich ist, wird eine Serveranwendung (nachfolgend „PositionHub“ genannt) entwickelt, die sowohl die Positionsdaten der HereIAm-App empfängt als auch verarbeitet und in Echtzeit an die HoloCommander App weiterleitet.

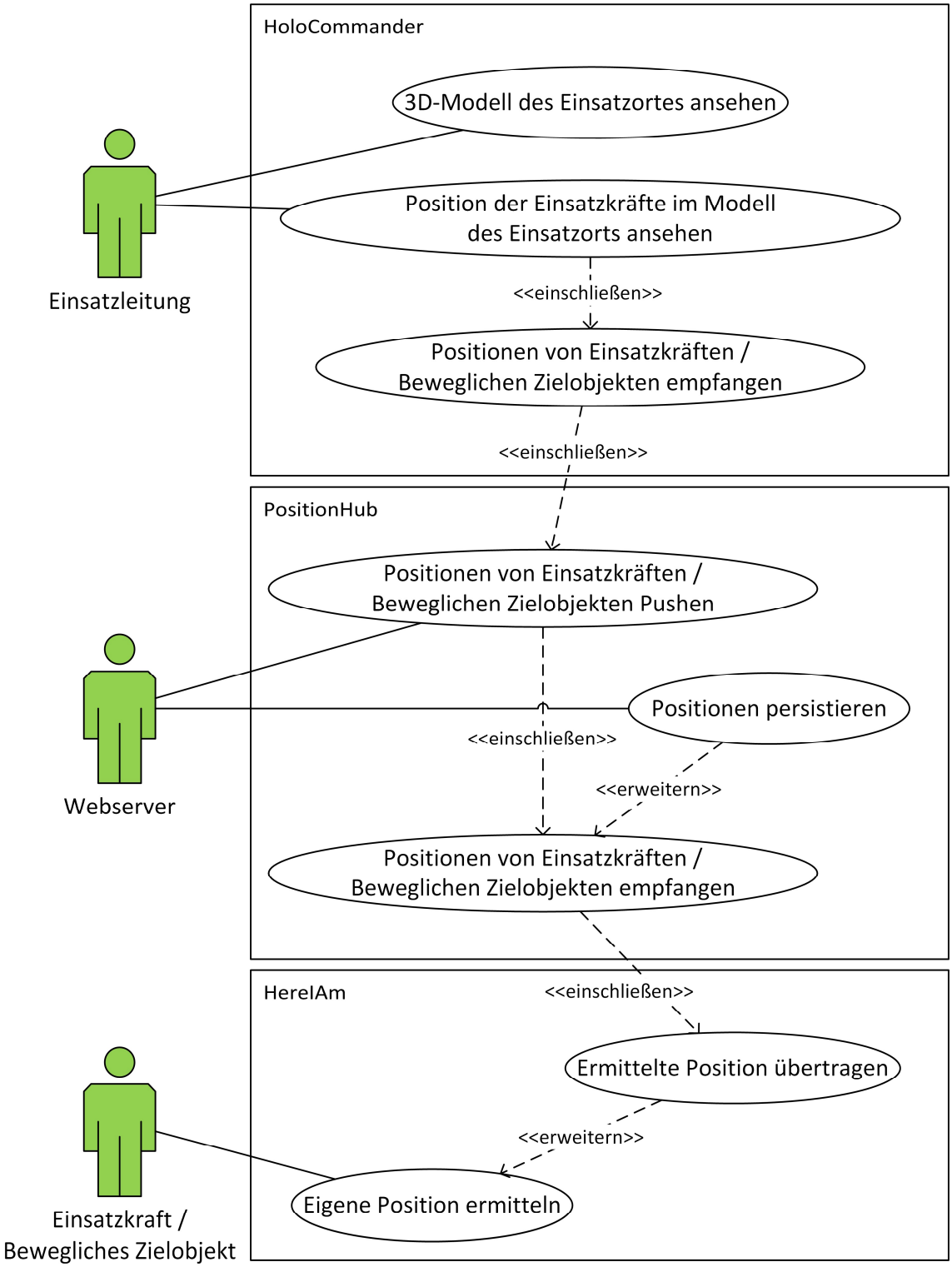

Abbildung 3.17: Übersicht der Anwendungsfälle für Einsatz (Ohne Persistenz)

Aus den Anwendungsfällen lassen sich folgende User Stories formulieren, die es im Rahmen des Konzeptes umzusetzen gilt:

1. Als Mitglied der Einsatzleitung möchte ich das 3D-Modell des Einsatzortes mithilfe einer Mixed-Reality-Brille ansehen können, um die örtlichen Gegebenheiten am Einsatzort besser zu überblicken.
2. Als Mitglied der Einsatzleitung möchte ich stets die aktuelle Position der Einsatzkräfte und der beweglichen Zielobjekte am Einsatzort innerhalb des 3D-Modells angezeigt bekommen um mir so einen besseren Eindruck vom tatsächlichen Geschehen vor Ort zu schaffen.
3. Als Mitglied der Einsatzleitung möchte ich alle Bewegungsdaten, die innerhalb eines Einsatzes angefallen sind, verwenden können, um mir die Protokollierung des Einsatzverlaufes zu erleichtern.
4. Als Mitglied der Einsatzleitung möchte ich die Möglichkeit bekommen einen vergangenen Einsatz erneut durchzuspielen um eine Auswertung der Entscheidungen und damit verbundenen Geschehnisse zu ermöglichen.
5. Als Einsatzkraft am Einsatzort möchte ich die Möglichkeit haben der Einsatzleitung ständig meine aktuelle Position mitzuteilen um ihr einen besseren Überblick über die Gesamtsituation zu verschaffen.

3.4.2 Funktionale Anforderungen

Aus den Rahmenbedingungen und den definierten User Stories lassen sich folgende funktionale Anforderungen an das Konzept definieren, die es zu erfüllen gibt:

3.4.2.1 Datenerhebung durch HerelAm

Die Smartphone App soll in der Lage sein, durch hardwareseitig vorhandene Assisted GPS Sensoren und ortsbasierte Dienste den eigenen Standort zu bestimmen und überträgt die Daten zum PositionHub, um sie in Echtzeit darzustellen und für die spätere Auswertung zu persistieren. Gemäß Abbildung 3.17 wird diese Teilfunktion der Rolle Einsatzkraft zugewiesen.

3.4.2.2 Übertragung der ermittelten Position zwischen HerelAm und PositionHub

Die Smartphone App soll den ermittelten eigenen Standort immer dann an den PositionHub übertragen, wenn sich der eigene Standort geändert hat. Darüber

hinaus muss der PositionHub benachrichtig werden, wenn die Einsatzkraft oder das Objekt die Teilnahme am Einsatz beendet, da diese ansonsten ständig am letzten übertragenen Standort dargestellt werden würde.

3.4.2.3 Push der empfangenen Position von PositionHub an HoloCommander

Der PositionHub soll die empfangene Position direkt an alle HoloCommander Pushen, die sich für Positionsupdates des betreffenden Einsatzes registriert haben.

3.4.2.4 Anzeige von Stadtmodellen im HoloCommander

Die HoloCommander Anwendung muss in der Lage sein Open Data Stadtmodelle darzustellen. Gemäß Abbildung 3.17 wird diese Teilfunktion der Rolle Einsatzleitung zugewiesen.

3.4.2.5 Anzeige von Empfangenen Positionsdaten im HoloCommander

Der HoloCommander muss die Position der Einsatzkräfte oder beweglichen Objekte im dreidimensionalen Modell der Umgebung darstellen können. Dafür ist es notwendig, die Umgebungsdaten des Modells auf echte Geopositionen zu mappen um eine korrekte Anzeige zu ermöglichen. Gemäß Abbildung 3.17 wird diese Teilfunktion der Rolle Einsatzleitung zugewiesen.

3.4.2.6 Persistieren der empfangenen Positionsdaten im PositionHub

Im PositionHub sollen die empfangen Daten zusätzlich zur direkten Weiterleitung an den HoloCommander persistiert werden.

3.4.2.7 Echtzeit-Wiedergabe der Positionsdaten im PositionHub

Die im PositionHub persistierten Positionsdaten sollen wiedergegeben werden können. Das bedeutet, dass der PositionHub das echte Empfangen von Positionsdaten simulieren kann, um im HoloCommander einen vergangenen Einsatz zur besseren Auswertung nachspielen zu können.

3.4.3 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben die Qualität, in der die funktionalen Anforderungen zu erfüllen sind.

3.4.3.1 Übertragungssicherheit

Da es sich bei den Personenpositionsdaten um sensible Daten handelt, sollen sie möglichst sicher übertragen werden können. Das bedeutet, dass zu keiner Zeit Daten unverschlüsselt übertragen werden sollen, sondern ausschließlich via HTTPS.

3.4.3.2 Effizienz, Flexibilität und Wartbarkeit

Die Anwendungen sollen so implementiert werden, dass plattformübergreifend möglichst viel Code wiederverwendet werden kann. Darüber hinaus soll es mit möglichst geringen Aufwand möglich sein, Änderungen vorzunehmen. Die Implementierung soll so erfolgen, dass möglichst viele mobile Clients eingesetzt werden können.

3.4.3.3 Kompatibilität

Alle zu entwickelnden Softwarebestandteile sollen mit Hinblick auf größter Kompatibilität entwickelt werden. Die mobile Clientanwendung HereIAm soll sowohl auf iOS als auch Android lauffähig sein. Die Server-Anwendung PositionHub soll sowohl auf Windows, Linux und OSX ausgeführt werden können. Die HoloCommander Anwendung soll mit dem Unity Framework entwickelt werden um ebenfalls höchste Kompatibilität mit verschiedenen Plattformen zum Ausführen der Anwendung zu ermöglichen.

3.5 Konzept

In den folgenden Abschnitten wird das theoretische Konzept für die Implementierung des Prototyps entwickelt. Dabei wird im Einzelnen erläutert, welche Schritte notwendig sind, um die Aufgabenstellung vollständig zu erfüllen.

3.5.1 Ideales Konzept

Das ideale Konzept stellt den Fall dar, der die gegebenen Anforderungen nach besten Möglichkeiten erfüllt. In den folgenden Schritten werden die Bestandteile eines Idealen Konzeptes beschrieben, dessen praktische Realisierbarkeit in der prototypischen Implementierung zu beweisen ist.

3.5.1.1 Komponentenüberblick

Da es sich um ein verteiltes System von mobilen Geräten handelt, wird zuerst ein technischer Überblick geschaffen:

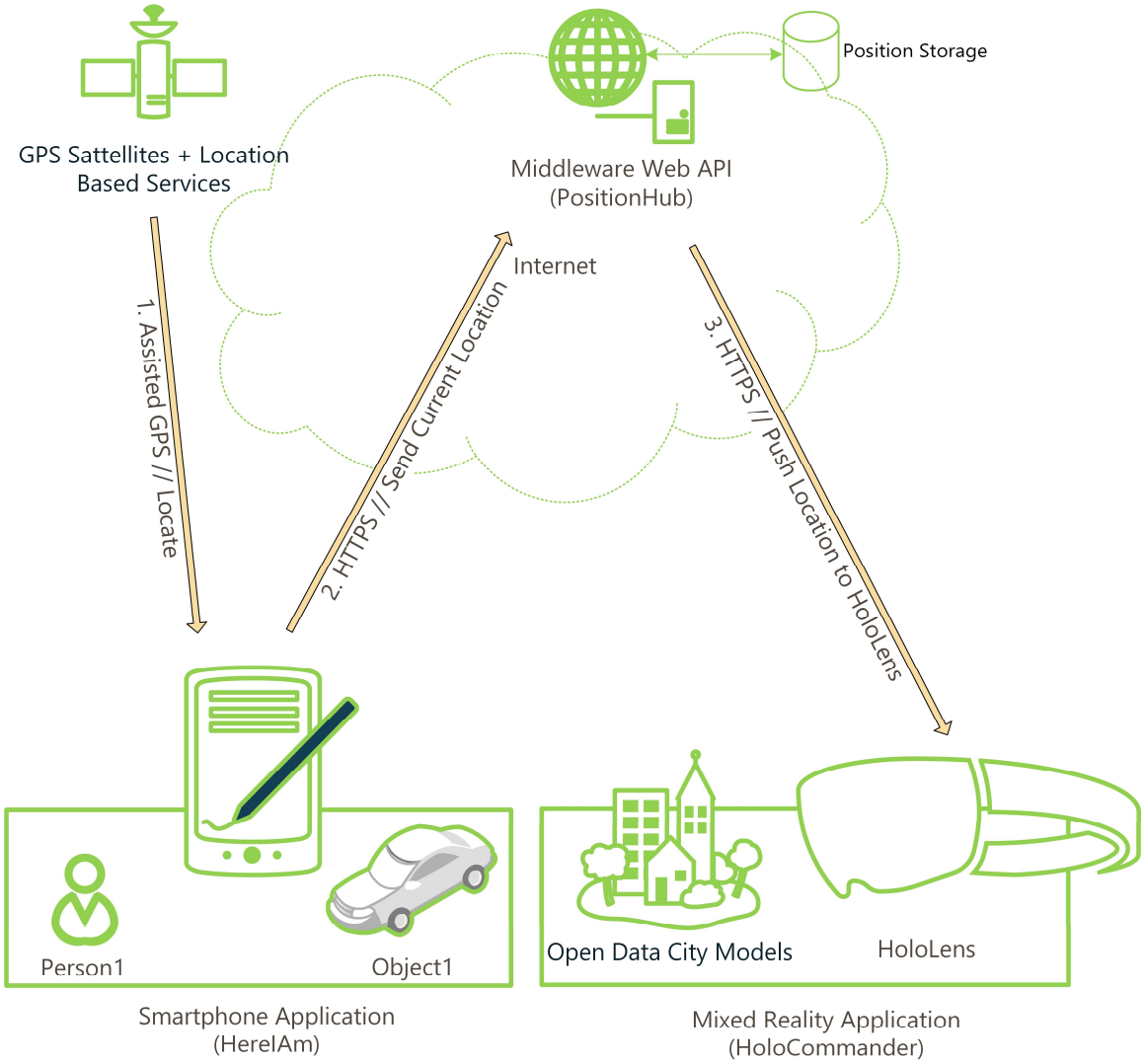

Abbildung 3.18: Überblick über alle Komponenten

Um eine Darstellung der aktuellen Geschehnisse auf mehreren Mixed Reality Geräten gleichzeitig zu ermöglichen und zusätzlich die eingehenden Daten zugunsten einer besseren Protokollierung oder späteren Auswertung zu speichern, wird eine Web-Anwendung namens PositionHub als zentrale Schnittstelle für die eingehenden Daten agieren. Diese Anwendung wird als Server die eingehenden Positionsdaten der HereIAm-Clients entgegennehmen, abspeichern und diese in Echtzeit via Push an den auf der HoloLens ausgeführten HoloCommander weiterleiten. Dieses Modell ergibt einen weiteren Vorteil gegenüber einer direkten Verbindung zwischen HoloCommander und HereIAm App: Der HoloCommander muss nicht im Internet erreichbar sein und agiert somit nicht als Server, sondern als Client. Das vereinfacht eine Netzwerkkonfiguration im privaten Netzwerk, indem der HoloCommander ausgeführt wird deutlich, da dieser keine eingehenden Verbindungen und somit keine Portfreigaben erfordert, sondern aufgrund des Server-Client-Prinzips lediglich selbstständig eine ausgehende Verbindung zum zentralen Server aufbaut. Auf der Seite der HereIAm-App ergibt sich dadurch der Vorteil, dass die Gefahr der IP-Adressänderung nicht besteht, selbst wenn sich die IP-Adresse auf der Seite der Einsatzzentrale aufgrund einer Verlegung an einen anderen Standort ändern würde.

3.5.1.2 Smartphone Anwendung: HereIAm

Die HereIAm Smartphone Anwendung wird als Cross-Plattform-App erstellt um möglichst großen Teil der Geschäftslogik nur einmalig implementieren zu müssen. Um das zu erreichen wird nach dem MVVM-Entwurfsmuster vorgegangen und die Anwendung in die drei Schichten Model, View und ViewModel unterteilt, wobei jede Schicht ihre eigenen Aufgaben zu erfüllen hat:

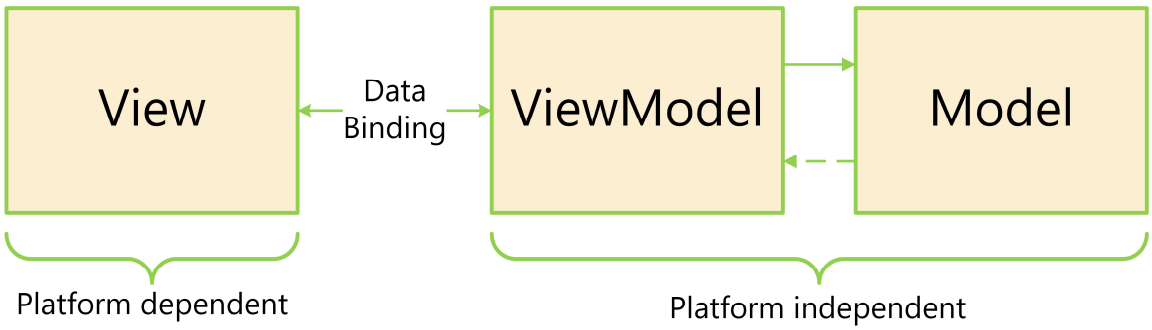

Abbildung 3.19: Schematische Darstellung des MVVM Entwurfsmusters

1. **Model**: Beinhaltet die Datenzugriffschicht und die Geschäftslogik und ist somit vollständig durch Unit Tests prüfbar. Diese Sicht ist bei Cross-Plattform-Anwendungen im plattformunabhängigen Teil (App Core) zu implementieren.
2. **View**: Beinhaltet alle grafischen Steuerelemente die dem Benutzer für die Interaktion mit der Anwendung angezeigt werden. Darüber hinaus beinhaltet diese Schicht die plattformspezifische Präsentationslogik um Animationen oder ähnliches darzustellen. Dieser Teil ist Abhängig von der Plattform, auf der die Anwendung ausgeführt werden soll und muss daher für jede Plattform implementiert werden.
3. **ViewModel**: Enthält die UI Logik (Model zur View) und stellt die Verbindung zwischen Model und View dar. Für jede plattformspezifische View existiert ein plattformunabhängiges ViewModel, welches jedoch keine Kenntnis über die Views hat. Der Datenaustausch zwischen beiden Schichten wird durch eine Datenbindung realisiert und durch Steuerungsumkehr (Inversion of Control) ausgelöst. Um dies zu realisieren werden plattformspezifische Abhängigkeiten gemäß (Dependency Injection) durchgeführt. Da das ViewModel genau wie das Model plattformunabhängig ist, wird es bei einer Cross-Plattform-Implementierung ebenfalls im App Core definiert.

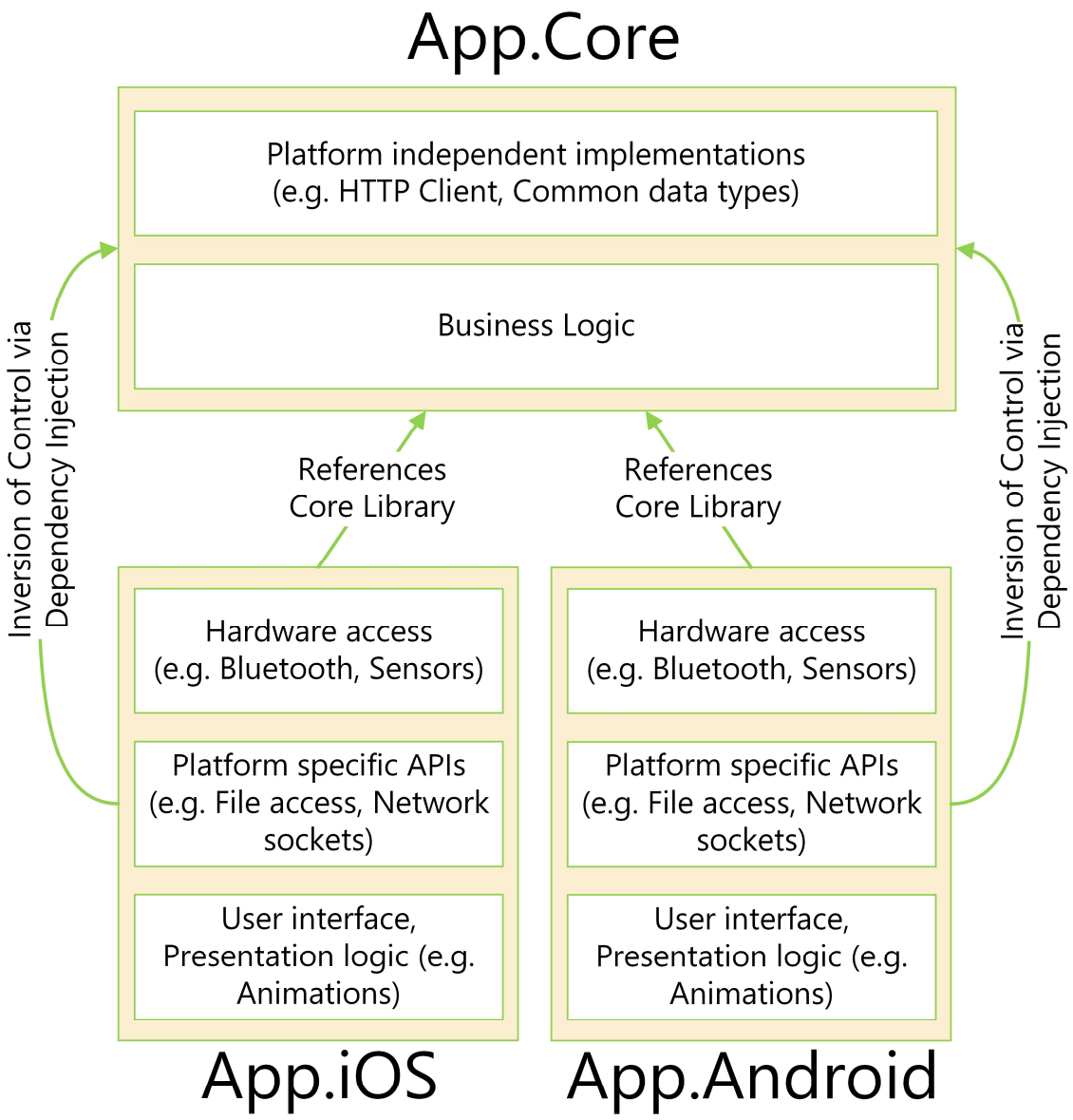

Abbildung 3.20: Aufbau einer Cross-Plattform-App

In der Smartphone App gibt es sowohl für iOS als auch für Android folgende Features zu implementieren:

1. Ermittlung der eigenen Position (plattformspezifisch)
2. Übertragung der Position zur Middleware, sobald sie diese ändert (plattformunabhängig)
3. Übertragung der Position zur Middleware in regelmäßigen Zeitabständen,

falls sich die ermittelte Position länger gleichgeblieben ist. (plattformunabhängig)

3.5.1.3 Middleware Web API: PositionHub

Die PositionHub Middleware wird als WebApi in Form einer ASP.NET Core Anwendung implementiert um auf allen gängigen Serverbetriebssystemen lauffähig zu sein. Zum einen wird ein Endpunkt zur Verfügung gestellt, durch den die HereIAm App ermittelte Positionsdaten zur Middleware senden kann. Zusätzlich dazu wird ein SignalR Endpunkt zur Verfügung gestellt, zu den sich die Instanzen der HoloCommander Anwendung, die auf mehreren HoloLenses ausgeführt werden, verbinden, um via Push Benachrichtigungen über neue Positionsdaten zu erhalten. Darüber hinaus muss noch ein weiterer Endpunkt zur Steuerung der Wiedergabe persistierter Positionsdaten zur Verfügung gestellt werden. Die HoloCommander App ist dadurch in der Lage, die Positionen zur Auswertung eines abgeschlossenen Einsatzes abzurufen oder alle Daten erneut wiederzugeben.

3.5.1.3.1 Endpunkt: /api/location

Schnittstelle zum Übermitteln des eigenen Standortes von HereIAm-App zum PositionHub. Die zu Nachricht soll folgende Parameter beinhalten und sowohl im XML-Format als auch im JSON-Format zum PositionHub übertragen werden können, um neben der HereIAm-Anwendung auch kompatibel zu möglichen anderen Clients zu sein. Die JSON-Formatierung ist jedoch aufgrund des geringeren Overheads gegenüber XML zu bevorzugen. Die gewählte Formatierung soll im HTTP-Header „Content-Type“ der POST-Anfrage definiert werden. Unabhängig welche Formatierung für die Datenübermittlung gewählt wurde, sollen im JSON-Format Attributbezeichnungen im „lowered Camel Case“ (beginnend mit einem kleinen Buchstaben) übertragen werden, um der ECMA-404 JSON Spezifikation gerecht

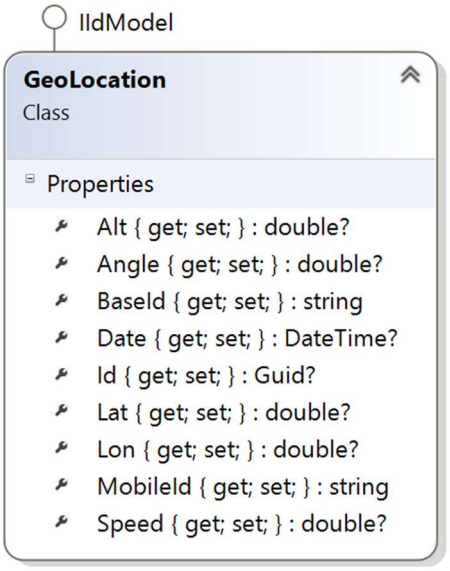

Abbildung 3.21: Aufbau der Klasse GeoLocation

zu werden (ECMA, 2013).

Aufbau der Klasse GeoLocation, die sterilisiert im XML- oder JSON-Format übertragen werden soll:

Eigenschaft	Beschreibung	Erforderlich/ Optional
Alt	Ermittelte Höhe in Meter, relativ zum Meeresspiegel	Optional
Angle	Bewegungsrichtung in Grad, relativ zum geografischen Nordpol	Optional
BaseId	Id des Einsatzes. Empfohlen im UUID-Format gemäß RFC4122 Spezifikation	Erforderlich
Date	Datum und Uhrzeit der Position, übertragen als String im ISO-8601 Format	Erforderlich
Id	Eindeutige des Datensatzes, im UUID Format gemäß RFC4122 Spezifikation. Diese Wird vom Server generieret.	Optional
Lat	Latitude, Geografische Breite	Erforderlich
Lon	Longitude, Geografische Länge	Erforderlich
MobileId	Id des Mobilgerätes zur Identifizierung der Person der des beweglichen Objektes. Empfohlen im UUID-Format gemäß RFC4122 Spezifikation	Erforderlich
Speed	Aktuelle Geschwindigkeit in Meter pro Sekunde	Optional

Tabelle 3.6: Spezifikation der GeoLocation-Klasse

Eigenschaften die als Erforderlich definiert sind müssen in jeder HTTP POST Anfrage vorhanden sein. Die als optional definierten Eigenschaften können entweder nicht oder mit dem Wert einer Null-Referenz übertragen werden (nil="true" als XML-Attribut bzw. „null" als Wert bei JSON).

Ein beispielhafte JSON-formatierte Nachricht würde dann wie folgt aussehen:

```
{
  "alt": 37.90179443359375,
  "angle": 0,
  "baseId": "Einsatz1",
  "date": "2016-12-21T15:25:17.13",
  "lat": 52.518460829802208,
  "lon": 13.415174771164118,
  "mobileId": "96020401-3e33-48bb-ae76-b62fd916c6c7",
  "speed": 9.2700004577636719
}
```

Listing 3.1: Beispiel JSON api/location-Endpunkt

Ein beispielhaftes XML-formatierte Nachricht würde dann wie folgt aussehen:

```
<GeoLocation>
    <Alt>37.90179443359375</Alt>
    <Angle>0</Angle>
    <BaseId>Einsatz1</BaseId>
    <Date>2016-12-21T15:25:17.13</Date>
    <Lat>52.518460829802208</Lat>
    <Lon>13.415174771164118</Lon>
    <MobileId>96020401-3e33-48bb-ae76-b62fd916c6c7</MobileId>
    <Speed>9.2700004577636719</Speed>
</GeoLocation>
```

Listing 3.2: Beispiel XML api/location-Endpunkt

3.5.1.3.2 Endpunkt: /signalr

Dieser Endpunkt stellt die SignalR-Server-Schnittstelle bereit und ermöglicht die SignalR-Kommunikation mit dem HoloCommander. Über diese Schnittstelle werden eintreffende Positionsinformationen der Location-API in Echtzeit via Push zur HoloCommander-Anwendung weitergeleitet. Die HoloCommander-Anwendung muss eine EinsatzId angeben, um den Einsatz festzulegen, zu dem Positionsinformationen in Echtzeit empfangen werden sollen.

3.5.1.4 Mixed Reality Anwendung: HoloCommander

Die HoloCommander-Anwendung stellt die Mixed-Reality-Umgebung für die Einsatzleitung bereit. Die Anwender tragen die HoloLens auf dem Kopf und erhalten mithilfe von dreidimensionalen Umgebungsmodellen des Einsatzortes einen besseren Überblick über das Geschehen vor Ort.

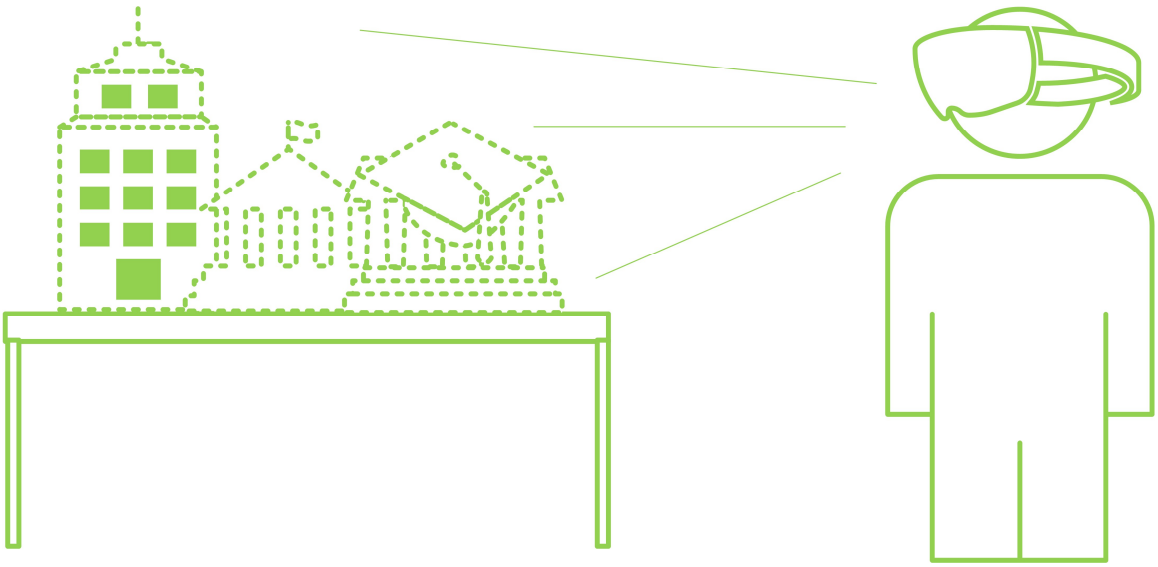

Abbildung 3.22: Veranschaulichung HoloCommander-App

Das Hologramm der Einsatzumgebung kann vom Anwender an einen beliebigen Ort, zum Beispiel auf einem freien Tisch der dem Fußboden projiziert werden. Anschließend ist er bereit mit dem Hologramm zu interagieren. Die Interaktion soll eine räumliche Bewegung um das dreidimensionale Hologramm beinhalten um den Einsatzort aus allen Blickwinkeln zu betrachten.

3.5.2 Konzeptüberführung

Bei der Umsetzung wird an das synchrone Zusammenspiel mehrerer HoloCommander-Anwendungen gedacht, welches bei simultaner Ausführung auf mehreren HoloLenses dazu führen würde, dass mehrere Anwender gleichzeitig mit ein und demselben Hologramm interagieren könnten. Die im PositionHub erforderlichen Schnittstellen zur simultanen Ausführung auf mehreren Geräten wird nicht implementiert, die Grundvoraussetzungen für einen Mehrbenutzereinsatz sind jedoch durch die Verwendung des SignalR-Frameworks bereits erfüllt. Es besteht die Möglichkeit die Clients in Gruppen zu organisieren und nur die Anwender über eine neue Position zu benachrichtigen, die zur Einsatzleitung eines bestimmten Einsatzes gehören. Eine parallele Durchführung eines anderen Einsatzes wäre dadurch ebenfalls möglich.

4 Implementierung

In den folgenden Abschnitten wird die Erstellung eines Prototyps gemäß den konzeptionellen Vorgaben beschrieben. Das ganze System wurde ausschließlich in der Programmiersprache C# implementiert. Das betrifft die native mobile Smartphone Anwendung „Here I Am“, die Webschnittstelle namens PositionHub und auch die Universal Windows Platform Anwendung HoloCommander, die auf der HoloLens ausgeführt wird. Zur eigentlichen Darstellung der Hologramme durch die Hololens wird die Unity Game Engine eingesetzt werden, welche ebenfalls C# zur Interaktionssteuerung verwendet.

4.1 Technischer Aufbau

Das Gesamtsystem, welches implementiert werden soll, teilt sich in vier große Bereich auf:

1. Smartphone Anwendung: Here I Am
2. Web Schnittstelle: Position Hub
3. HoloLens Anwendung: HoloCommander
4. Open Data 3D Stadtmodelle von Berlin

Die ersten drei dieser Bereich stellen jeweils eine Anwendung dar, die auf einer anderen Hardware ausgeführt werden muss. Der vierte Bereich stellt den Import und die anschließende Darstellung der vom Land Berlin bereitgestellten dreidimensionalen Gebäudemodellen dar.

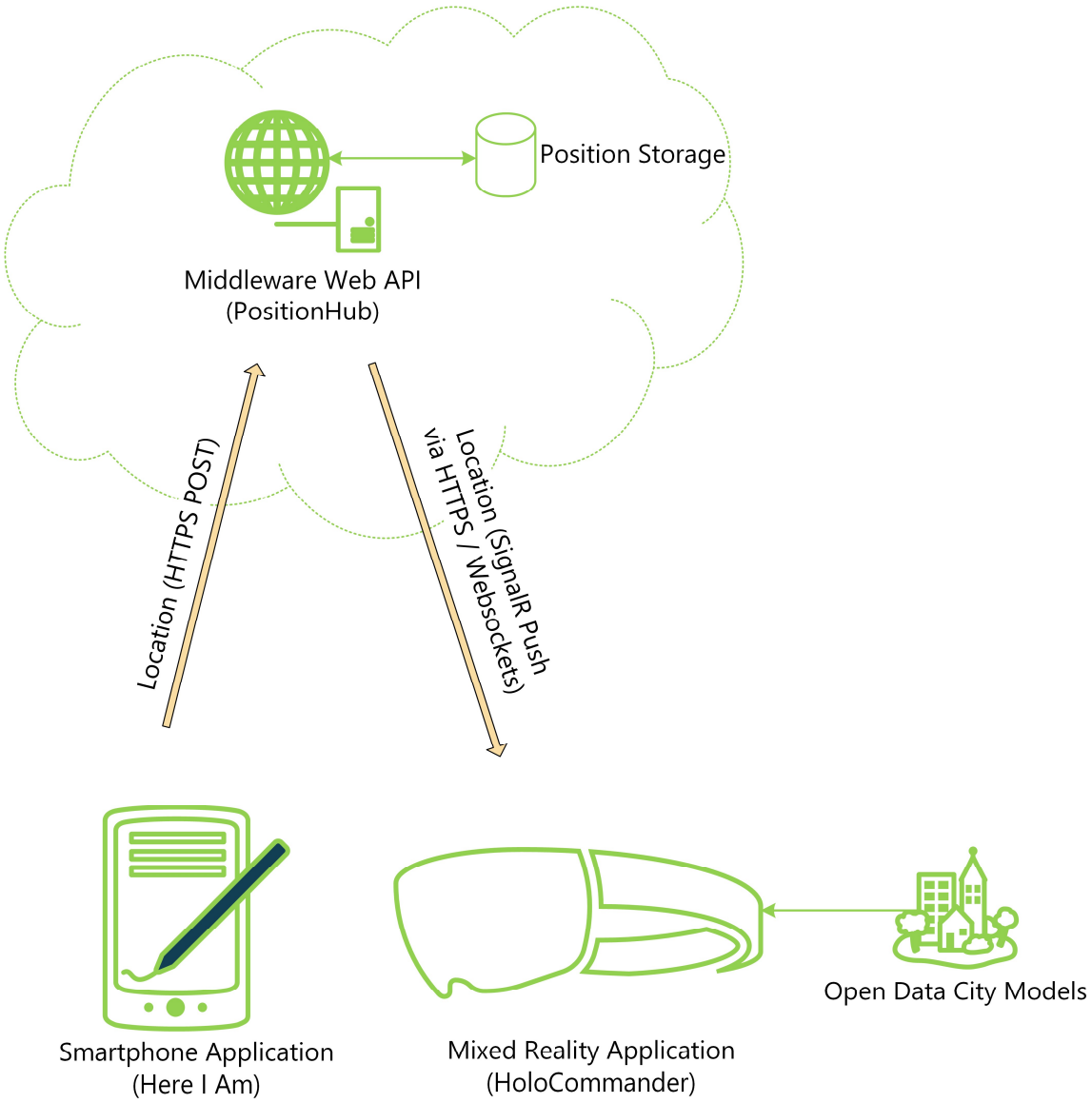

Abbildung 4.23: Technischer Aufbau Gesamtsystem

Die Smartphone Anwendung hat einzig und allein die Aufgabe, die eigene Position zu bestimmen und diese an die Webschnittstelle weiterzuleiten. Die Webschnittstelle wiederum persistiert die eingehenden Positionsinformationen und leitet diese außerdem per Push mittels SignalR an alle HoloLenses weiter, auf denen die HoloCommander App ausgeführt wird. Zusammen mit den Positionsinformationen und den dreidimensionalen Stadtmodellen wird dem Träger der HoloLens ein Echtzeit-Eindruck der sich real abspielenden Bewegungen am Einsatzort vermittelt.

4.2 Entwicklungswerkzeuge

Für die Implementierung des Gesamtsystems sind viele Unterschiedliche Werkzeuge erforderlich.

Zur Erstellung der mobilen Smartphone Applikation für iOS und Android mit Hilfe von Xamarin ist ein Apple Computer mit OSX Betriebssystem erforderlich, da das Programm, obwohl es in der Programmiersprache C# implementiert wird, durch einen Cross-Compiler in nativen Objective-C Code umgewandelt wird, bevor die App mithilfe von Apples Xcode durch den nativen Standard-Compiler für iOS Anwendungen kompiliert wird. Die Android-Version kann jedoch sowohl unter Windows als auch unter OSX erstellt werden. Hierbei findet ebenfalls eine Cross-Kompilierung in nativen Java-Code statt, welcher anschließend mithilfe des Standard Android SDKs in eine native Android App kompiliert wird. Da für die Erstellung des iOS-Kompilats ohnehin ein Computer mit der XCode Entwicklungsumgebung vorhanden sein muss, wird die gesamte Smartphone App (Here I Am) sowohl für iOS als auch für Android mithilfe der Xamarin Studio Entwicklungsumgebung in C# implementiert.

Die Erstellung der Webservice Middleware (PositionHub) erfolgt unter Windows im Visual Studio. Obwohl der PositionHub aufgrund der Tatsache, dass er auf dem .NET Core Framework basiert, unter Linux, OSX und auch Windows lauffähig ist, wird er unter Windows im Visual Studio implementiert. Die dort enthaltenen umfangreichen Werkzeuge unterstützen den Entwickler bei der Implementierung. Als einfachen Code-Editor zum Erstellen und Debuggen von .NET Core Anwendungen unter Linux, OSX und Windows bietet Microsoft das Programm Visual Studio Code an, welches zwar den Familiennamen „Visual Studio“ in sich trägt, im besten Fall aber einen durch Plug-Ins erweiterbaren leichtgewichten Editor mit geringem Funktionsumfang, im Vergleich zum „echten“ Visual Studio, darstellt.

Die HoloLens App (HoloCommander) wird mit Hilfe der Game Engine Unity und

der gleichnamigen Unity-Entwicklungsumgebung erstellt. In Unity hat der Entwickler sowohl die Möglichkeit, seine zwei- oder dreidimensionale Welt grafisch zu erstellen, als auch jedes grafische Objekt mit Skripten zu bestücken, welche die Interaktionssteuerung zur Laufzeit der Anwendung ermöglichen. Die Skripte werden in C# implementiert und durch den Compiler des Mono Frameworks kompiliert. Nutzt ein Entwickler Unity unter Linux oder OSX, wird Mono Develop, welches mit Unity ausgeliefert wird, zur Implementierung der C# Skripte verwendet. Unter Windows, so wie in diesem Fall, wird die Erstellung der Skripte, aufgrund der im Vergleich zum Mono Develop besseren Entwicklerwerkzeuge, im Visual Studio erfolgen.

4.3 Beschreibung der Komponenten

In den folgenden Absätzen wird jede technische Komponente für sich beschrieben. Dabei wird für jede Komponente ein Aspekt, der für die Implementierung der jeweiligen Komponenten besonders schwierig, neu oder außergewöhnlich ist, besonders hervorgehoben.

4.3.1 Open Data Stadtmodelle von Berlin

Das Land Berlin stellt die gesamten 3D-Modelle der Stadt zum Download auf einem FTP-Server bereit. Man kann die ganze Stadt oder einzelne Bezirke anhand der Verzeichnisstruktur und diese im Gesamten herunterladen. Wem die Datenmenge zu groß ist, der bekommt die Möglichkeit, einzelne Gebäude oder Gebiete auf der Website Berlin 3D - Downloadportal (Business Location Center, o.J.) zum Download zu markieren. Die Markierung erfolgt durch die Auswahl der in 3D projizierten Gebäude auf einer Open Street Map. Dabei stehen die Formate CityGML (mit oder ohne Textur), 2D Shape, 3D Shape, KMZ, DXF, DWG und 3DS (mit oder ohne Textur) zur Verfügung. Nachdem man sich für ein Format entschieden und seine E-Mail Adresse angegeben hat, bekommt man nach einer Bearbeitungszeit von bis zu mehreren Stunden eine E-Mail, welche einen Downloadlink zu einer Zip-Datei auf einem FTP-Server enthält, in der die Modelle im entsprechenden Format der zuvor gewählten Gebäude vorhanden ist.

Da das Ziel war, eine möglichst detailgetreue Umgebung darzustellen, galt es sich auf die Datenformate zu konzentrieren, die inklusive Textur exportiert werden können. Demnach können von den ursprünglich sieben Formaten nur noch CityGML und 3DS in Betracht gezogen werden. Um die Qualität der beiden Exporte zu vergleichen wurde das Konzerthaus Berlin, welches sich auf dem Gendarmenmarkt befindet, exportiert.

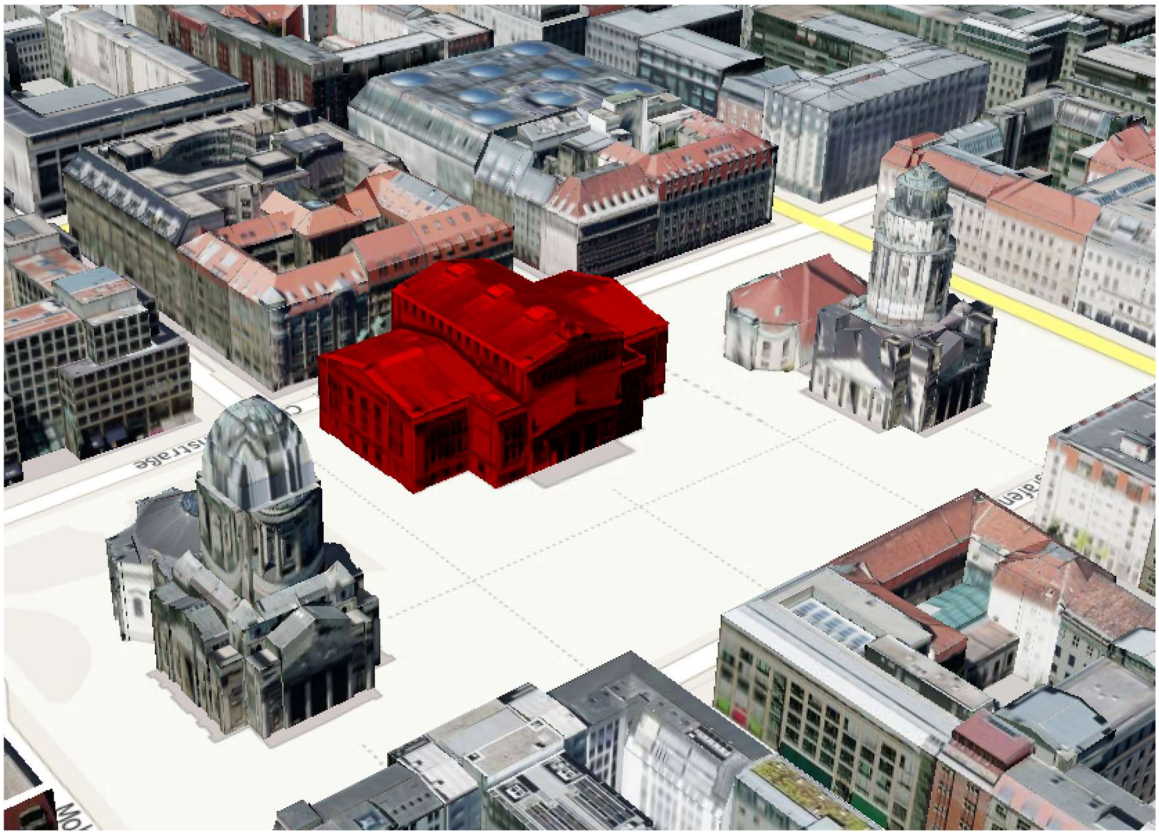

Abbildung 4.24: Konzerthaus Berlin in 3D im Downloadportal

Ganz gleich für welches der beiden Formate man sich entscheidet, man bekommt immer, je nach gewähltem Format, eine Datei mit den identischen Vektordaten entweder im CityGML- oder 3DS-Format und dazu unabhängig vom Format die gleichen Texturen. Für das Konzerthaus Berlin bedeutet das konkret, dass das exportierte Modell 588 Vektoren enthält, welche 424 Flächen bilden. Die Texturen für dieses Gebäude werden als 81 JPEG-Dateien in unterschiedlichen Abmaßen geliefert, welche mit insgesamt 686KB als Textur für ein einzelnes Gebäude recht groß sind.

Unter genauer Betrachtung des Modells im 3DS-Format mithilfe des Open Source Programms MeshLab (CityGML kann vom Programm nicht geöffnet werden), welches zur Verarbeitung von 3D Meshes gedacht ist, fällt sofort auf, dass sich durch die Anwendung einfacher Optimierungsalgorithmen eine signifikante Reduzierung der Anzahl von Vektoren und Flächen durchführen lässt. So können allein durch die Entfernung doppelter Vektoren (Überführung von mehreren Vektoren mit den gleichen Koordinaten in einen einzigen) 382 Vektoren eingespart und die Anzahl auf 206 reduziert werden. Das entspricht einer Vektorenanzahl von ca. 35% gegenüber dem Original, bei gleichbleibender Flächenanzahl.

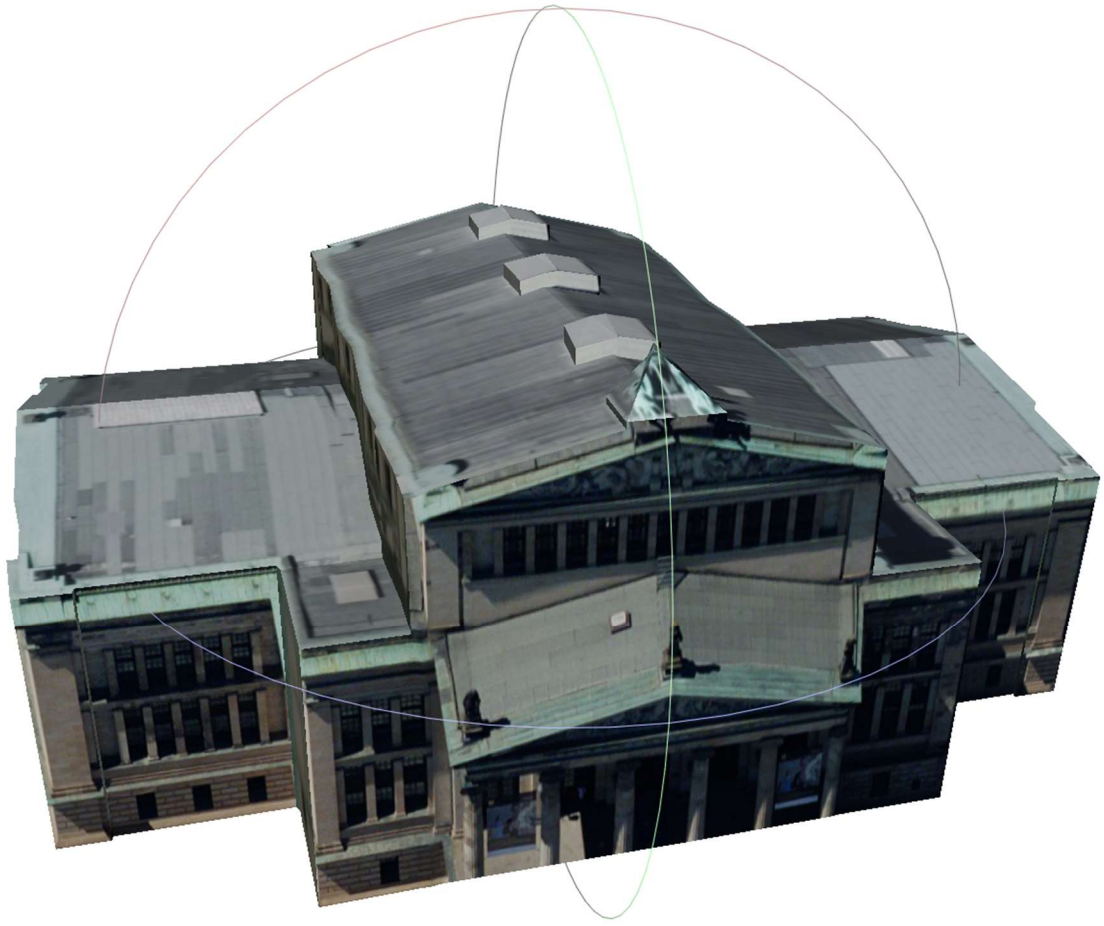

Abbildung 4.25: Konzerthaus Berlin in 3D in Mashlab

Als zweites fällt auf, dass die Unterteilung der Textur für das Gebäude in 81 Bildern mit unterschiedlichen Abmaßen ungewöhnlich erscheint, werden doch typischer Weise Texturen für ein 3D-Objekt als einzelne Bilddatei für das gesamte Modell generiert.

Um sich nun final zwischen CityGML und 3DS als Formate für die Darstellung der Modelle mit Hilfe der HoloLens zu entscheiden wird geprüft, welche der beiden Formate durch Unity geöffnet und verarbeitet werden können. Da Unity nur mit Modelle verarbeiten kann, die im OBJ- oder im FBX-Format vorliegen, wird eruiert mit welchem Aufwand sowohl CityGML als auch 3DS in OBJ oder FBX umgewandelt werden können. Da sowohl 3DS als auch FBX zu Autodesk als Hersteller von Software zur Erstellung von 2D und 3D Modellen gehören, bietet Autodesk mit dem Programm Autodesk FBX Converter ein kostenfreies Werkzeug an, mit dem sich 3DS-Dateien schnell in FBX-Dateien umwandeln lassen. Diese können anschließend problemlos in Unity importiert werden.

4.3.2 HoloCommander – Unity vs. UWP

Da die HoloCommander App auf der HoloLens ausgeführt werden soll, wird in Unity ein Plattform-Export der Anwendung für die Universal Windows Plattform (UWP) durchgeführt. Das ist notwendig, weil ausschließlich UWP-Apps, welche wiederum auf dem .NET Framework basieren, auf der HoloLens ausgeführt werden können. Unity generiert darauf hin bei jedem Export für UWP ein neues Standard Visual Studio .NET Framework UWP-Projekt, welches den C# Code zum Initialisieren der Unity Game Engine, direkt beim Start der UWP-App, beinhaltet. Sobald die Game Engine direkt nach dem Start der UWP App mithilfe einer plattformabhängigen Brücke initialisiert ist, wird innerhalb der Game Engine die in Unity erstellte Anwendung gestartet. Die Unity-Anwendung läuft jetzt im plattformunabhängigen Unity-kontext und wird gewissermaßen nur von der UWP-Anwendung gehostet.

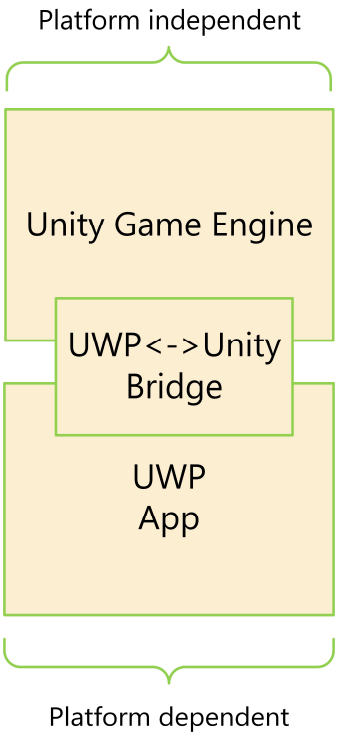

Abbildung 4.26: UWP Unity Host

4.3.2.1 Die Stadt als Hologramm

Als Beispielort zur Darstellung berliner Open Data Gebäude dient weiterhin der Berliner Gendarmenmarkt in Mitte. Zusätzlich zum bereits analysierten Konzerthaus Berlin wurden die Modelle vom Französischen Dom und dem Deutschen Dom heruntergeladen, vom 3DS-Format ins FBX-Format umgewandelt und in Unity importiert, um die Darstellung des Platzes zu vervollständigen. Um darüber hinaus einen besseren Eindruck der Umgebung zu bekommen, wurde der Ausschnitt der Open Street Map Karte des Platzes und der umliegenden Straßen als Ebene hinzugefügt, auf der die Gebäude platziert wurden.

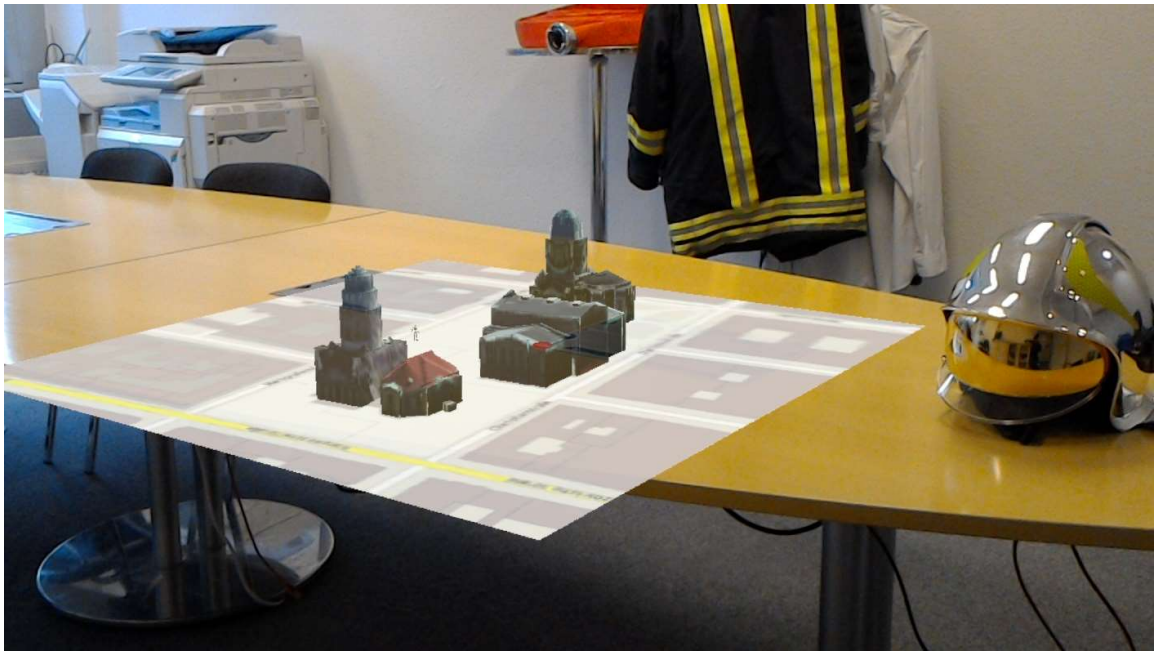

Abbildung 4.27: Bildschirmaufnahme HoloCommander mit Modellen des Gendarmenmarkts

Um die Position einer Person oder eines Objektes darzustellen, wird ein beliebiges Personen-Modell, hier in Form eines Roboters, aus dem kostenlosen Unity Asset Store importiert und ebenfalls auf dem Modell des Gendarmenmarktes platziert. Anschließend wird dem Roboter ein Unity Script zugewiesen, welches bei eintreffenden Koordinaten von echten Personen oder Objekten dafür sorgt, dass zuerst die globalen Geokoordinaten von Längen- und Breitengrade in passende Koordinaten im Unity-Modell bezogen auf den echten Gendarmenmarkt umgerechnet werden und im Anschluss den Roboter zu den ermittelten Koordinaten bewegt. Da davon auszugehen ist, dass jede Person bzw. jedes Objekt bei

der kleinsten Veränderung des eigenen Standortes die eigenen Koordinaten verschickt, ergibt sich für jede eintreffende Koordinate ebenfalls eine neue Koordinate im Modell.

Bereits bei der Analyse der Modelle der Gebäude ist aufgefallen, dass die Modelle in Bezug auf Vektoranzahl und Texturen nicht optimal erstellt sind (siehe Kapitel 4.3.1). Das schlägt sich nun auf die Performanz der Anwendung nieder, denn betrachtet man die Gebäude zur Laufzeit des Programmes mit der HoloLens genauer, stellt man fest, dass die Anwendung bei direktem Blick auf eines der Gebäude merklich anfängt zu ruckeln. Das Ruckeln wird dadurch ausgelöst, dass die HoloLens die angestrebten 60 Frames pro Sekunde nicht mehr darstellen kann, sondern sich die Framerate merklich soweit reduziert, bis sie vom menschlichen Auge als Ruckeln wahrgenommen werden kann. Als Grund kann vermutet werden, dass die verwendeten Modelle der Gebäude vom Gendarmenmarkt mit den vielen Texturdateien in unterschiedlichen Abmessungen für die Anwendung mit der HoloLens nicht optimal erstellt wurden.

Abbildung 4.28: Bildschirmaufnahme HoloCommander mit alternativen Modellen

Um diese Vermutung zu untermauen wurde ein Paket bestehend aus sechs kostenlosen 3D Gebäuden aus dem Unity Asset Store geladen. Vier dieser Gebäude

wurden anstelle der Modelle der echten Berliner Gebäude auf den Gendarmenmarkt platziert. Bei der Ausführung der Anwendung mit der HoloLens lässt sich nun kein Ruckeln feststellen. Bei näherer Betrachtung eines dieser Gebäude nach gleichen Gesichtspunkten wie bei der Analyse des Konzerthaus Berlin kommen folgende Parameter zum Vorschein: Das Gebäude besteht aus 2204 Vektoren, die 3736 Flächen bilden. Das entspricht mehr als die dreieinhalbfache Anzahl an Vektoren und mehr als das Achtfache an Flächen, verglichen mit dem Konzerthaus Berlin. Der Optimierungsversuch durch entfernen von doppelten Vektoren wird damit mit dem Ergebnis Null beendet, keine doppelten Vektoren vorhanden. Die Texturen für dieses Gebäude sind zusammengefasst zu einer einzigen PNG-Datei, welche 10,9MB groß ist, jedoch mit der quadratischen Größe von 2048x2048 Pixel eine optimale Form zum rendern von 3D-modellen besitzt. Daraus lässt sich schließen, dass die Modelle der Berliner Gebäude wahrscheinlich ruckelfrei dargestellt werden würden, würden die Texturen von vielen Dateien unterschiedlicher Größe zu einer einzelnen, quadratischen Datei optimiert werden.

4.3.2.2 Aktualisierung der Position – Die Technik

Neben der Darstellung des 3D Stadtmodells ist die zweite Hauptaufgabe der App, Positionsdaten von Personen und Objekten in Echtzeit zu empfangen und die entsprechende Position im Modell zu verändern. Um einen Push vom PositionHub in Echtzeit zu empfangen muss die HoloCommander Anwendung einen SignalR Client implementieren. Die Open Source Implementierung der SignalR Projektgruppe basiert auf dem .NET 4.0 oder höher. Die in Unity erstellten C# Skripte werden gegen Mono 2.0 (kompatibel zu .NET 2.0) kompiliert. Demzufolge ist der Standard SignalR Client nicht mit Unity kompatibel. Da das Problem mittlerweile länger besteht gibt es Open Source SignalR-Client-Implementierungen von dritten, die auf ältere .NET Framework-Versionen abzielen. Verwendet man einen solchen .NET 2.0 SignalR Client in Unity, ist es möglich innerhalb der Unity IDE eine Verbindung zum SignalR Server, gehostet vom PositionHub, aufzubauen. Der Export der Unity-Anwendung für die UWP-Plattform schlägt jedoch fehl, da UWP-Projekte auf dem .NET Framework 4.5 basieren und nur noch die asynchrone „Non-Blocking“-Programmierschnittstelle unterstützen, die durch

asynchrone Ausführung von langsamen Programmteilen (so auch HTTP-Anfragen) das einfrieren der Grafischen Benutzeroberfläche verhindern sollen. Diese API ist jedoch im .NET 2.0 bzw. Mono 2.0 Framework noch nicht verfügbar. Daraus ergibt sich ein Framework-Versionskonflikt, der sich jedoch folgendermaßen umgehen lässt:

Es ist möglich innerhalb eines C# Skriptes in Unity öffentliche Klassen mit Öffentlichen Methoden und Eigenschaften zu definieren, auf die man von der UWP-Anwendung aus zugreifen kann. Durch die Einhaltung von Softwarearchitektur-Entwurfsmuster wie „Dependency Injection“ und „Inversion of Control“ ist es möglich, im Unity Projekt plattformunabhängige Schnittstellen zu definieren und diese innerhalb der plattformabhängigen Anwendung zu implementieren.

```
public class SignalRUnityController : MonoBehaviour
{
    //removed code not important here!
    [...]

    //define custom delegate type
    public delegate void StartSignalRConnectionEventHandler(string url);

    //define static variable of custom delegate type
    public static StartSignalRConnectionEventHandler StartSignalRConnection;

    //define static variable of generic .NET Framework delegate type
    public static EventHandler<GeoLocationEventArgs> LocationReceived;

    //Gets called by Unity during app initialization
    void Start()
    {
        //wiring up delegate with local handler method, called by UWP app
        LocationReceived = LocationReceivedHandler;

        if (UseSignalR)
            //call delegate to wired up by UWP app to connect to SignalR hub
            UnityEngine.WSA.Application.InvokeOnUIThread(() =>
                   StartSignalRConnection(SignalRUrl), false);
    }
    //handler, called by UWP app
    void LocationReceivedHandler(object sender, GeoLocationEventArgs e)
    {
        UnityEngine.Debug.Log("Location Reveived: " + e.Location.Id);
        _lastGeoLocation = e.Location;
    }
}
```

Listing 4.3: Unity Skript zur SignalR Steuerung

Überführt man dieses Prinzip auf das Problem mit der SignalR Client-Komponente, so lässt sich das Problem der inkompatiblen Frameworks wie folgt umgehen: Wenn man den für die UWP-Plattform verfügbare Standard SignalR Client einbindet, kann sich mit dem SignalR Server verbinden. Allerdings besteht so noch keine Möglichkeit mit dem Unity-Kontext zu interagieren.

Definiert man innerhalb eines C# Skriptes statische öffentliche Eigenschaften innerhalb von öffentlichen Klassen, kann darauf von der UWP App aus zugegriffen werden. Somit ist eine Steuerung von außen (plattformabhängig) nach innen (plattformunabhängig möglich). Um nun die Steuerung von innen nach außen (Inversion of Control) zu ermöglichen, werden im Unity Skript statische Eigenschaften als Delegate definiert. Delegate im Sinne von C# sind Typen, die anders als herkömmliche Referenzvariablen nicht auf Objekte referenzieren, sondern stattdessen auf Methoden mit einer vorgegebenen Signatur (Globale Parameterliste und Rückgabetyp). Weist man nun einer Delegat-Variablen eine beliebige, zur Signatur passenden Methode zu, kann diese Methode innerhalb des Zugriffs auf diese Variable aufgerufen werden, auch wenn die Klasse, die diese Methode enthält, innerhalb des Sichtbarkeitsbereiches nicht zugreifbar ist. Konkret bedeutet das: Wenn man innerhalb der UWP App einer statischen Delegat-Variablen eine Methode zuweist, die sich ebenfalls innerhalb einer Klasse innerhalb der UWP App befindet, so kann diese Delegat-Referenzvariable im Kontext des Skriptes aufgerufen werden. Die Voraussetzung dafür ist jedoch, dass die Zuweisung des Delegats vor der Ausführung des Skriptes geschieht.

```
internal class SignalR2UnityBridge
{
    //removed code not important here!
    [...]

    //Singleton instance of this class
    private static SignalR2UnityBridge _instance;

    //SignalR hub connection
    private HubConnection _hubConnection;

    //constructor wires up methods to delegates to get called
    //within Unity script
    private SignalR2UnityBridge()
    {
        SignalRUnityController.StartSignalRConnection = StartSignalRConnection;
    }

    //initialization method called by UWP app at startup to before initializing
    //Unity to create a singleton instance
    public static SignalR2UnityBridge Build()
    {
        return _instance ?? (_instance = new SignalR2UnityBridge());
    }

    //method called by Unity scipt via delegate
    private void StartSignalRConnection(string url)
    {
        //connect to SignalR server and subscribe to location hub
        _hubConnection = new HubConnection(url);
        var locationProxy = _hubConnection.CreateHubProxy("LocationHub");

        //when new location is received via SignalR connection
        locationPrody.On<GeoLocation>("NotifyNewLocation", location =>
        {
             //call handler via delegate from Unity script
             SignalRUnityController.LocationReceived?.Invoke(this,
                   new GeoLocationEventArgs(location));
        });

        //removed code not important here!
        [...]
    }
}
```

Listing 4.4: UWP Brücke zu Unity zur SignalR Steuerung

Dafür bietet sich der Programmstart der UWP App an, zu einem Zeitpunkt bevor die Unity Game Engine Initialisierung abgeschlossen wird. Um zu verhindern, dass Unity den plattformabhängigen, beim Export der Unity Anwendung für die UWP-Plattform selbsterstellten Teil überschreibt, kann man Dateien konfigurieren, die beim Export nicht überschrieben werden.

4.3.3 PositionHub

Der PositionHub agiert als Mittelsmann zwischen der Here I Am App, die die Positionsdaten der Personen oder beweglichen Objekte erfasst und der HoloCommander App, die auf der HoloLens ausgeführt wird und die Positionen im 3D Stadtmodell anzeigt.

Um das zu ermöglichen stellt der PositionHub zwei Schnittstellen bereit. Die erste Schnittstelle ist eine REST-API, die von der Here I Am App aufgerufen wird, sobald eine neue Position erkannt wurde. Die zweite Schnittstelle stellt einen SignalR Server dar, der Verbindungen von SignalR Clients annimmt und dadurch in die Lage ist, Nachrichten in Form von Push vom Server zum Client zu senden, sobald in der REST-API neue Positionen eintreffen.

Der PositionHub ist als ASP.NET Core Anwendung implementiert und somit als unabhängige Anwendung auf Linux, OSX und Windows-Betriebssystemen lauffähig. Außerdem kann sie, ähnlich wie eine Node.JS Anwendung, in Web- beziehungsweise Application- Servern bereitgestellt werden. ASP.NET Core ist Microsofts ASP (Active Server Pages) Lösung, die dadurch, dass sie auf dem .NET Core Framework basiert, für Cloud-Hosting optimiert ist. Sie ist darauf ausgelegt sehr leichtgewichtig zu sein um somit möglichst wenig Rechenleistung in Anspruch zu nehmen.

Da der PositionHub sowohl von den mobilen Smartphone Clients als auch von der HoloLens erreichbar sein muss, muss sie im Internet bereitgestellt werden. Um den Hosting-Prozess zu vereinfachen, wird die Anwendung auf Microsofts Azure Cloud Plattform als Web App bereitgestellt. Web Apps in Bezug auf Azure stellen den „Platform as a Service“-Ansatz von Cloudlösungen dar und bieten im Gegensatz zum „Infrastructure as a Service“-Ansatz den entscheidenden Vorteil, dass die gesamte Hardware und auch das Betriebssystem abstrahiert wird und der Host somit wie „von der Stange“ funktioniert und keine größeren Konfigurationen notwendig sind. Man wählt einen Domain Namen als Subdomain zu *.azurewebsites.net und bekommt dazu gleich ein passendes SSL/TLS-Zertifikat um

die Anwendung im Internet sicher via HTTPS erreichen zu können. Der PositionHub wird also vorerst unter *https://dowser-hereiam.azurewebsites.net/* bereitgestellt. Die Veröffentlichung der Anwendung kann auf verschiedene Wege erfolgen, in diesem Fall wurde ein FTP-Upload direkt aus dem Visual Studio heraus durchgeführt.

Der PositionHub stellt folgende öffentliche Schnittstellen:

Ressource	**Art**	**Beschreibung**
/api/Location	HTTP POST	Endpunkt für den Versand der ermittelten Positionen der Here I Am Smartphone Anwendung. Die empfangenen Positionsdaten werden persistiert, alle verfügbaren HoloCommander Instanzen werden via Push über die neue Position informiert
/api/LocationDirect	HTTP POST	So wie /api/location/, allerdings werden die Positionen nicht persistiert. Dieser Endpunkt ist zum Test gedacht.
/signalr	SignalR Server	Endpunkt für den SignalR Client des HoloCommanders
LocationHub	SignalR Hub	SignalR Hub auf dem die eingehenden Positionsdaten verteilt werden.

Tabelle 4.7: Öffentliche Schnittstellen des PositionHubs

Um die eingehenden Positionsdaten zu persistieren kommt das Entity Framework Core zum Einsatz. Das Entity Framework ist Bestandteil des .NET Core Frameworks und stellt einen einfachen, aber mächtigen Objekt-Relationalen Mapper bereit der Treiber für viele Objekt- und Relationale Datenbankmanagementsysteme bietet. Im Fall des PositionHubs wird Azure SQL verwendet um

eingehende Positionsdaten zu persistieren. Azure SQL stellt die Cloud-Variante des Microsoft SQL Servers dar. Das bedeutet, dass die Datenbank ebenfalls im Azure Rechenzentrum von Microsoft gehostet wird.

Im folgenden Listing ist der Controller dargestellt, der die eingehenden Positionen der Smartphone Anwendung Here I Am empfängt und an alle HoloCommander Apps weiterleitet.

```
//class inherits from HubControler and can notify every
//SignalR client which subsribes LocationHub channel
public class LocationDirectController : ApiHubController<LocationHub>
{
    //removed unimportant code here!
    [...]

    //method gets called when HTTP POST request is received
    //to /api/LocationDirect endpoint
    [HttpPost]
    public IActionResult Post([FromBody] ApiClient.DataContracts.GeoLocation value)
    {
        //validate reveived JSON or XML POST data which is deserialized into
        //variable 'value' as instance of GeoLocation class.
        if (!ModelState.IsValid)
        {
            //returns HTTP status 400 (Bad Request) if not valid.
            return BadRequest(ModelState);
        }
        if (!value.Validate())
        {
            //returns HTTP status 400 (Bad Request) if nor valid.
            return BadRequest();
        }
        try
        {
            //set unique UUID for incoming message
            value.Id = Guid.NewGuid();

            //notify all SignalR clients about new position via SignalR push.
            //'value' gets serialized as JSON for SignalR push
            Clients.All.NotifyNewLocation(value);
        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex);
            //returns HTTP status 500 (Internal Server Error)
            //inclusive error information (exception details, stack trace, etc.)
            throw;
        }
        //returns HTTP status 200 (OK)
        return Ok();
    }
}
```

Listing 4.5: LocationDirect Controller des PositionHubs

4.3.4 Here I Am

Die Hauptaufgabe der Here I Am Anwendung besteht darin die eigene Position zu ermitteln und zum PositionHub zu übertragen. Da die App als native Anwenudng sowohl für iOS als auch für Andorid erstellt werden soll, wird sie mit

Hilfe des Xamarin.Forms Frameworks als Cross-Plattform Anwendung umgesetzt. Beim Cross-Plattform-Ansatz wird die Anwendung pro Plattform in zwei Teile unterteilt, einen plattformspezifischen und einen plattformunabhängigen. Der Entwickler versucht so viel Code wie möglich im plattformunabhängigen Teil (Core App) der Andendung auszulagern, wie zum Beispiel die Geschäftslogik oder die Datenzugriffe. Teile die aufgrund von plattformspezifischen Abhängigkeiten nicht in den plattformunabhängigen Teil ausgelagert werden können, werden mit Hilfe des Entwurfsmusters „Dependency Injection“ im Core -Teil zur Verfügung gestellt. Da es sich bei der Here I Am App um eine native Anwendung handelt, ist die gesamte Benutzeroberfläche inklusive der Präsentationslogik und außerdem die Positionsbestimmung plattformabhängig. Damit die Oberfläche nicht zweimal, also für iOS und für Android plattformspezifisch implementiert werden muss, schafft das Xamarin.Forms Framework die Möglichkeit die Oberfläche einmalig in der Core App in Form von XAML Markups (basierend auf XML) zu definieren. Während der Laufzeit der App wird das XAML Markup interpretiert und die native Benutzeroberfläche für die jeweilige Plattform generiert. Zum Beispiel wird ein in XAML definierter Button unter iOS als „UIKit.UIButton“ und unter Android als „android.widget.Button“ generiert. Im folgenden Listing sieht man das XAML Markup für die Hauptseite der App, bestehend aus einem Grid, in dem mehrere Labels, Entries (Textboxen), zwei Switches (Toggle Buttons) und Buttons als Schaltflächen definiert sind.

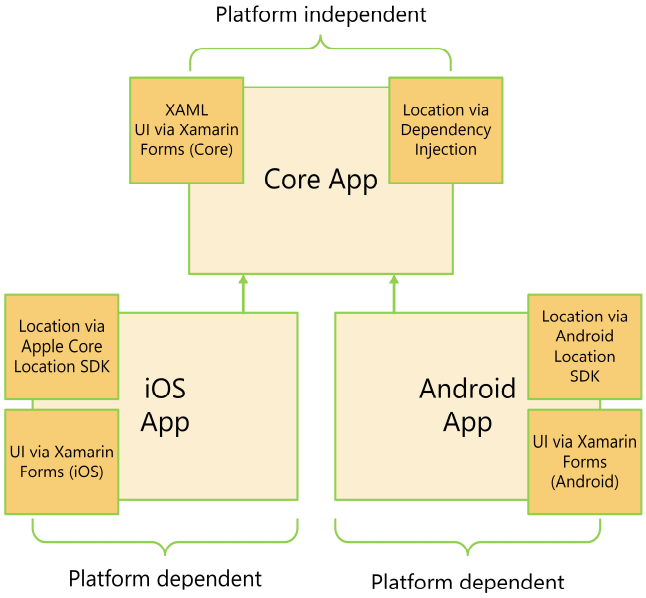

Abbildung 4.29: Softwarearchitektur der Here I Am App

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:local="clr-namespace:Dowser.HereIAm.Mobile"
       x:Class="Dowser.HereIAm.Mobile.MobilePage">
  <Grid  Margin="10,40,10,10">
    <!-- Removed row and column definition of grid here-->

    <!-- Platform independent definition of UI elements-->
    <Label Grid.Column="0" Grid.Row="0" Text="Latitude:" />
    <Label x:Name="LatLabel" Grid.Column="1" Grid.Row="0" Text="xx" />
    <Label Grid.Column="0" Grid.Row="1" Text="Longitude:"/>
    <Label x:Name="LongLabel" Grid.Column="1" Grid.Row="1" Text="xx" />
    <Label Grid.Column="0" Grid.Row="2" Text="Altitude:"  />
    <Label x:Name="AltLabel" Grid.Column="1" Grid.Row="2" Text="xx" />
    <Label Grid.Column="0" Grid.Row="3" Text="Mobile Id:" />
    <Entry x:Name="MobileIdEntry" Grid.Column="1" Grid.Row="3"
       Text="{}{0D5D806D-0855-4287-B6A0-964C33D2E371}" />
    <Label Grid.Column="0" Grid.Row="4" Text="Update position automatically:" />
    <Switch x:Name="UpdatePositionSwitch" Grid.Column="1" Grid.Row="4" IsToggled="true" />
       <Label Grid.Column="0" Grid.Row="5" Text="Show Notifications" />
    <Switch x:Name="ShowNotificationsSwitch" Grid.Column="1" Grid.Row="5"
       IsToggled="true" />
    <Button Grid.Column="0" Grid.Row="6" Text="Manual update"
       Command="{Binding UpdatePositionCommand}"/>
  </Grid>
</ContentPage>
```

Listing 4.6: Plattformunabhängiger XAML Code für iOS und Android UI

Um die Funktionsweise der Anwendung zu testen wurden zwölf freiwillige Testpersonen mit der App für ihr eigenes iOS oder Android Mobiltelefon ausgestattet um sie unter realen Bedingungen zu testen. Sie sollten sich in einem Zeitraum von vier Stunden auf einer Veranstaltung, die im Dezember in Berlin auf dem Gendarmenmarkt stattgefunden hat frei bewegen und ihre Positionsinformationen mithilfe der Here I Am App, die dabei im Hintergrund lief, zum PositionHub übertragen. Dabei wurden insgesamt 10.425 Datensätze mit Positionsinformationen vom PositionHub empfangen.

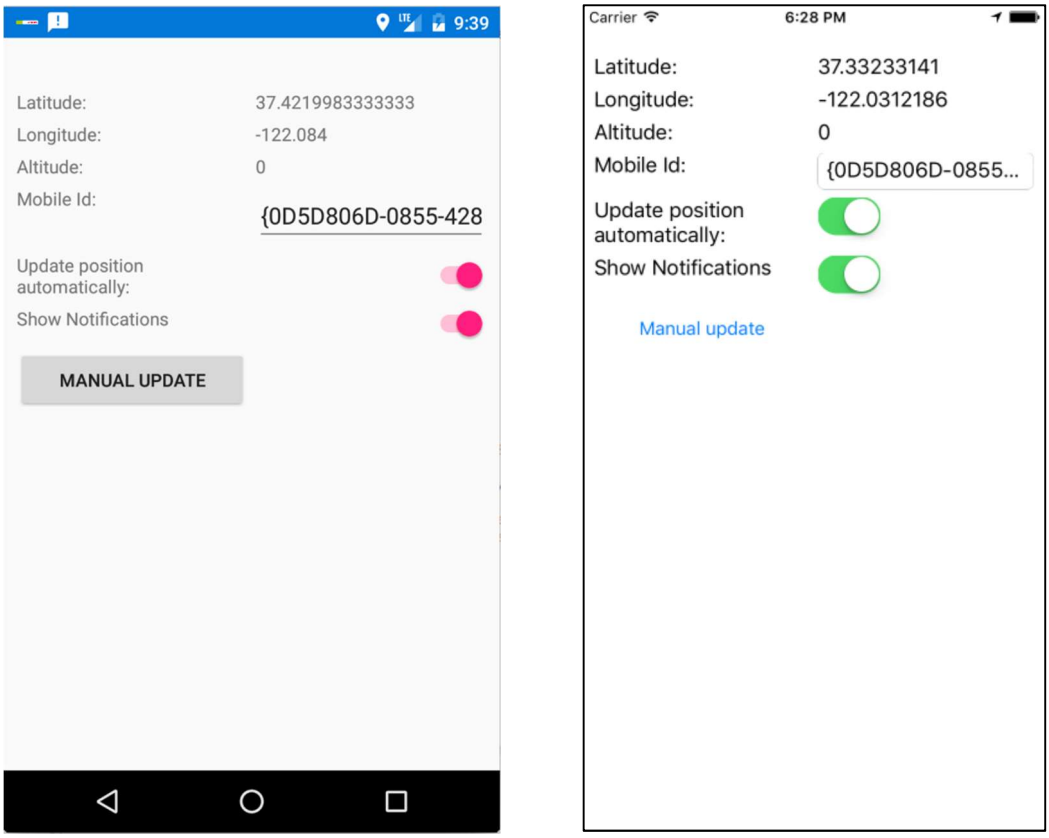

Abbildung 4.30: Bildschirmaufnahmen der HerelAm App für Android (links) und iOS (rechts)

4.4 Bewertung der Ergebnisse

Im Anschluss an die Implementierung soll nun eine Bewertung der Ergebnisse vorgenommen werden. Dabei wird nochmals jede der vier Komponenten unabhängig betrachtet.

4.4.1 Open Data Stadtmodelle

Ber Land Berlin stellt seine Stadtmodelle nach expliziter Auswahl der Gebäude zum Download zur Verfügung. Dadurch, dass nach der Auswahl eine Zeit von bis zu mehreren Stunden vergehen kann, bis man den Link zum Download des Modells per E-Mail zugesendet bekommt, lässt sich der Prozess der dynamischen Einbindung dieser Modell nur sehr schwer realisieren. Um eine umfassende Darstellung aller Gebäude und Plätze in Berlin in kurzer Zeit zur Verfügung zu haben, müsste man alle Daten abrufen und sie gesondert aufbereiten. Hinzu kommt, dass die angebotenen Formate der 3D Modelle vielfältig sind, aber für die praktische Anwenudng nicht ideal sind, um sie ohne vorherige Konvertierung

direkt zu benutzen. CityGML als quelloffenes standardisiertes Format zum Austausch von 3D Stadtmodellen zur bietet zwar sehr hohes Potential, kann in der Praxis jedoch aufgrund fehlender Unterstützung nicht immer einfach verarbeitet werden. Um die Modelle des Landes Berlin zu importieren ist daher eine vorherige Umwandlung in die gängigen Formate OBJ oder FBX erforderlich.

Als großer Nachteil stellt sich die Qualität der angebotenen 3D-Modelle dar. Die Vektoren der Modelle könnten mit einfachen mathematischen verfahren optimiert werden und dadurch viel Speicherplatz zu sparen und performanter zu rendern. Darüber hinaus ist die Auslieferung der Modelle mit vielen Textur-Dateien in jeweils unterschiedlichen Abmaßen ungeeigtet zur Darstellung in eigenständigen Geräten wie die HoloLens, die die Grafik selbst rendern und nicht von hochleistungsfähigen Desktop-Computern unterstützt werden, denn sie bieten in etwa eine ähnliche Leistungsfähigkeit wie gängiges Mobiltelefone im oberen Preissegmet. Besser wäre es die Texturen so aufzubereiten, dass sie als eine quadratische Bilddatei in einer für 3D Programme gängigen Größe, wie zum Beispiel 2038x2048 oder 1024x1024 Pixel bereit gestellt werden würde. Zur Textur selbst ist noch hinzuzufügen, dass sie aus zusammengesetzten Luftaufnahmen der Gebäude besteht. Überlagerungen wie Baukräne, Schatten anderer Gebäude oder Bäume sind direkt auf der Textur zu erkennen und mindern somit die Qualität des Modells.

Die vom Land als Beispiel bereit gestellt Smartphone App „Smart Map“ bietet zwar die Darstellung der Gebäude Berlins in 3D an, aber auch hier wurden die Modelle nochmals grundlegend überarbeitet. Als Benutzer bekommt man zwar die Gebäude in 3D angezeigt, hat jedoch nur die Möglichkeit sie aus vier vorgegebenen Blickwinkeln zu betrachten (Nord, Süd, Ost, West). Die Darstellung der Gebäude aus diesen vier Perspektiven scheint serverseitig als Bild vorgerendert, somit bekommt der Nutzer der App keinen wirklichen räumlichen Eindruck der Umgebung.

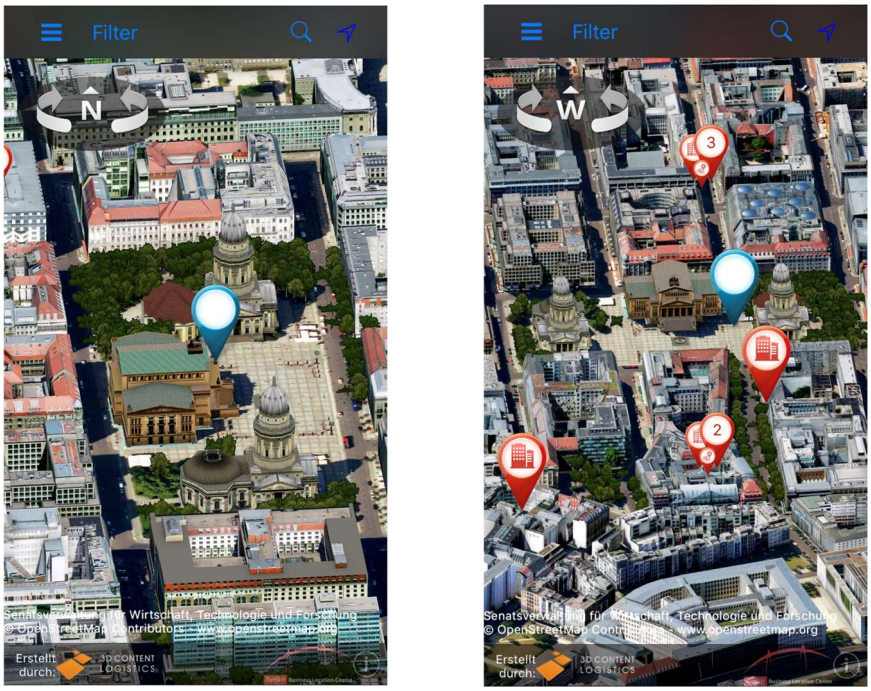

Abbildung 4.31: Bildschirmaufnahme der App smartMap aus zwei Perspektiven

4.4.2 HoloCommander

Die HoloCommander App zur Darstellung der Hologramme der Berliner Gebäude ist mit der HoloLens entwickelt worden. Alle geplanten Funktionen konnten umgesetzt werden, wenn es jedoch aufgrund des Zusammenspiels zwischen Unity und der UWP-Plattform zu beginn einige Startschwierigkeiten gegeben hat. Die HoloLens selbst ist ein eigenständiges Gerät und bietet im Vergleich zu Desktop Computern, die zur Darstellung von Computerspielen oder 3D Design spezialisiert sind, nur geringe Performance. Das muss vorallem bei der Erstellung der Modelle berücksichtigt werden, die durch die HoloLens dargestellt werden sollen. Auf besonders rechenintensive Bestandteile, wie Licht und Schatten, sollte entweder verzichtet oder möglichst zur Design-Zeit der Anwendung von einem leistungsstarken Computer durch sogenanntes „light baking“ und nicht während der Programmlaufzeit durch die HoloLens durchgeführt werden.

4.4.3 Here I Am

Die Here I Am App als native mobile Clientanwenung konnte mit durch den Einsatz des Xamarin Frameworks sehr einfach als Cross-Plattform App für die Plattformen iOS und Android umgesetzt werden. Da das übermitteln der eigenen Popsition via HTTP-Protokoll geschieht, ist im realen Umfeld ein Einsätz eines beliebigen Geräts denkbar, solange es in der Lage ist, via HTTP über eine mobile oder stationäre Internetverbindung zu kommunizieren.

4.4.4 PositionHub

Der PositonHub als Mittelsmann wurde in Form einer plattformunabhängigen -.NET Core Anwendung entwickelt und in der Microsoft Azure Cloud gehostet. Der Vorteil dabei ist die Skalierbarkeit, denn ohne großen Aufwand kann der Betreiber der Anwendung dem Host mehr Rechenleistung und mehr Speicher zuweisen (vertikale Skalierung) oder dafür sorgen, dass die Anwendung in mehreren Instanzen parallel ausgeführt wird (horizontale Skalierung), ohne manuell einen Load Balancer konfigurieren zu müssen. Darüber hinaus bietet das Cloud Hosting den OnDemand-Vorteil in Zusammenhang eines „Pay as you go“ Vertrages was bedeutet, dass die benötigten Kapazitäten in Form von Rechenleistung flexibel angepasst werden können und am Ende nur so viel Speicher und Rechenzeit bezahlt werden muss, wie tatsächlich genutzt wurde. Im Rahmen dieser Implementierung muss also im Vorfeld keine Server-Hardware angeschafft werden um den PositionHub im Internet bereitzustellen.

Die Echtzeitübertragung der Positionsdaten als Push mit Hilfe von SignalR funktikoniert genau so wie geplant. Da es Implementierungen vom SignalR Client in vierschiedenen Programmiersprachen und für verschiedene Plattformen gibt, ist eine Darstellung der Positionsdaten in einer beliebigen anderen Form denkbar, so könnte zum Beispiel eine Website entwickelt werden, auf der die konkreten Positionen der Einheiten in echtzeit auf einer zweidimensionalen Karte dargestellt werden.

5 Fazit und Ausblick

Ziel dieser Arbeit war es, einen Prototyp zu entwickeln, der in der Lage ist, in Echtzeit Positionsdaten von Objekten und Personen in einem Mixed Reality Umfeld darzustellen, welches in das bestehende Projekt DOWSER als zusätzliches Hilfsmittel eingebunden werden kann. Konkret sollten die Echtzeitpositionsdaten in Form von Hologrammen von dreidimensionalen Modellen wie Plätzen, Straßen oder Vierteln auf der HoloLens visualisiert werden. Speziell sollten hier die Daten der Open Data Stadtmodelle am Beispiel Berlins verwendet werden. Die Anwender des Programms, in der Rolle einer Einsatzleitung, sollten mit Hilfe von Mixed-Reality in der Lage sein, in der Einsatzzentrale einen Überblick über die sich am Einsatzort bewegenden Einsatzkräfte und Objekte, wie zum Beispiel Fahrzeuge, zu bekommen. Um das zu ermöglichen, wurde im Rahmen dieser Arbeit erfolgreich ein Kommunikationskonzept mit Schnittstellen entworfen und implementiert, deren Daten auf der HoloLens visualisiert werden. Erkannt wurden die Rollen Einsatzleitung und Einsatzkraft, die zugleich auch bewegliche Objekte wie Fahrzeuge, Straßensperren, oder ähnliches sein können. Das Ziel einer übersichtlichen Darstellung von 3D-Informationen für die Einsatzleitung bereitzustellen, um somit die Auswirkungen über zu treffende Entscheidungen besser abschätzen zu können, wurde erreicht. Die erhobenen Daten werden neben der Live-Darstellung für eine spätere Auswertung gespeichert. Somit leistet diese Arbeit einen positiven Beitrag im Rahmen der Digitalisierungsstrategie von Behörden und Organisationen mit Sicherheitsaufgaben. Im Rahmen dieser Arbeit stellte sich heraus, dass die vom Land Berlin bereitgestellten Open Data Stadtmodelle nicht direkt eingebunden werden können, da diese durch Schnittstellenvorgabe nur mit erheblicher Wartezeit und in einer unzureichenden Qualität abgerufen werden können, die sich nicht für den direkten Einsatz mit der HoloLens eignet. Für die Visualisierung wurde zusätzlich die HereIAm App entwickelt, die auch unabhängig vom Projektrahmen anderweitig zur Erfassung von Positionsdaten genutzt werden kann, aber auch für die Weiterentwicklung dieses Mixed-Reality-Ansatzes.

Abschließend wurde mithilfe von Live- und Simulationsdaten geprüft, ob der

Mixed-Reality-Ansatz in Verbindung mit der automatischen Ortung von Einsatzkräften und -fahrzeugen zur Informationsüberlagerung führen könnte. Diese Bedenken konnten nicht bestätigt werden. C# als Programmiersprache konnte ohne Probleme für die Umsetzung für verschiedene Plattformen genutzt werden.

Um in Zukunft eine Anwendung zu gestalten, die im Einsatzfall direkt verfügbar ist, müss die aktuell vom Land Berlin zur Verfügung gestellte Schnittstelle zugunsten von optimierten 3D-Modellen und direkten Ladezugriffen angepasst werden. Im unmittelbaren Anschluss wird im Rahmen einer Masterarbeit eine Gesamtdigitalisierungsstrategie für Behörden und Organisationen mit Sicherheitsaufgaben unter Berücksichtigung der erreichten Teilergebnisse dieser Arbeit entwickelt. Die Erkenntnisse dieser und Folgearbeiten werden im Projekt DOWSER genutzt.

QUELLENVERZEICHNIS

3D Geoinformation group at TU Delft. o. J.. CityGML Homepage. [Online] o. J. [Zitat vom: 22. 02 2017.] https://www.citygml.org/.

Aguilar, Jose M. 2014. *SignalR Programming in Microsoft ASP.NET.* Redmond, Washington, USA : Microsoft Press, 2014. 978-0735683884.

Android. o.J.. Location. *API Reference.* [Online] o.J. [Zitat vom: 30. 01 2017.] https://developer.apple.com/reference/corelocation.

Apple. o. J. a. Erinnerungen auf dem iPhone, iPad oder iPod touch verwenden. [Online] o. J. a. [Zitat vom: 30. 01 2017.] https://support.apple.com/de-de/HT205890.

—. o.J. b. Framework Core Location. *API Reference.* [Online] o.J. b. [Zitat vom: 30. 01 2017.] https://developer.apple.com/reference/corelocation.

—. o. J. c. iPhone Modelle vergleichen. [Online] o. J. c. [Zitat vom: 27. 01 2017.] http://www.apple.com/de/iphone/compare/.

Banks, Andrew und Gupta, Rahul. 2015. MQTT Version 3.1.1 Plus Errata 01. [Online] 10. 12 2015. [Zitat vom: 07. 02 2017.] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html.

Berlin Partner für Wirtschaft und Technologie GmbH. o. J.. #Berlin3D - Hacking Berlin's City Model. *Business Locartion Center.* [Online] o. J. [Zitat vom: 18. 01 2017.] http://www.businesslocationcenter.de/berlin3d-hackathon.

—. 2015 a. Hauptstadt-Modell als Open Data zugänglich. *Business Location Center.* [Online] 13. 03 2015 a. [Zitat vom: 18. 01 2017.] http://www.businesslocationcenter.de/imperia/md/blc/wirtschaftsatlas/downloadportal/content/2015-03-13_pi-blc-stadt-3d.pdf.

—. 2015 b. Projektübersicht #Berlin3D. *Business Location Center.* [Online] 16. 07 2015 b. [Zitat vom: 18. 01 2017.] http://www.businesslocationcenter.de/imperia/md/blc/wirtschaftsatlas/downloadportal/content/berlin3d_projektuebersicht.pdf.

Bitkom. 2016 a. Anteil der Smartphone-Nutzer in Deutschland in den Jahren 2012 bis 2016. *Statista - Das Statistik-Portal.* [Online] 08 2016 a. [Zitat vom: 22. 02 2017.] https://de.statista.com/statistik/daten/studie/585883/umfrage/anteil-der-smartphone-nutzer-in-deutschland/.

—. 2016 b. Anteil der Smartphone-Nutzer in Deutschland nach Altersgruppe im Jahr 2016. *Statista - Das Statistik-Portal.* [Online] 08 2016 b. [Zitat vom: 22. 02 2017.] https://de.statista.com/statistik/daten/studie/459963/umfrage/anteil-der-smartphone-nutzer-in-deutschland-nach-altersgruppe/.

BIU. 2016. Anteil der deutschen Internetnutzer, die Virtual-Reality-Brillen für digitale Spiele nutzen wollen, nach Altersgruppe im Jahr 2016 . *Statista - Das Statistik-Portal.* [Online] 08 2016. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/448749/umfrage/umfrage-zum-interesse-an-virtual-reality-brillen-fuer-digitale-spiele-in-deutschland-nach-alter/.

Buckl, Dr. Christian. 2013. Vorlesungsfolien: Echtzeitsysteme. *Technische Universität Münschen.* [Online] 2013. [Zitat vom: 06. 02 2017.] http://www6.in.tum.de/pub/Main/TeachingWs20123chtzeitsysteme/Echtzeit-Gesamt.pdf.

Bundesministerium der Justiz und für Verbraucherschutz. 2015. Bundesdatenschutzgesetz (BDSG). [Online] 25. 02 2015. [Zitat vom: 02. 03 2017.] http://www.gesetze-im-internet.de/bdsg_1990/BJNR029550990.html.

Business Location Center. o.J.. Berlin 3D - Downloadportal. [Online] o.J. [Zitat vom: 17. 03 2017.] http://www.businesslocationcenter.de/berlin3d-downloadportal/.

Comer, Douglas E. 2011. *TCP/IP - Studienausgabe.* s.l. : MITP Verlags GmbH , 2011. 978-3826691119.

D21, Initiative und ipima. 2016. Welche der folgenden Angebote von Open Government kennen Sie, haben Sie bereits genutzt bzw. würden Sie zukünftig gerne nutzen? . *Statista - Das Statistik-Portal.* [Online] 09 2016. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/166999/umfrage/bekannheit-und-nutzung-von-egovernment-angeboten/.

Deloitte. 2016. Prognose zum Umsatz mit Virtual Reality in Deutschland von 2016 bis 2020 (in Millionen Euro) . *Statista - Das Statistik-Portal.* [Online] 04 2016. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/604199/umfrage/prognose-zum-umsatz-mit-virtual-reality-in-deutschland/.

Deloitte, FIT, Fraunhofer und Bitkom. 2016. Prognose zum B2B-Umsatz mit Virtual-, Augmented- und Mixed-Reality in Deutschland von 2016 bis 2020 (in Millionen Euro). *Statista - Das Statistik-Portal.* [Online] 05 2016. [Zitat vom: 16. 02 2017.]

Dr. Nils Barnickel, Dr. Matthias Flügge, Daniel Hanke, Dr. Edzard Höfig, Jens Klessmann, Florian Marienfeld, Prof. Dr. Ina Schieferdecker, Jan Ziesing. 2012. *Berliner Open Data-Strategie - Organisatorische, rechtliche und technische Aspekte ofener Daten in Berlin. Konzept, Pilotsystem und Handlungsempfehlungen.* Berlin : Fraunhofer Verlag, 2012.

Duden. o.J.. Lage. *Stelle, wo etwas (in Bezug auf seine Umgebung) liegt.* [Online] o.J. [Zitat vom: 16. 02 2017.] http://www.duden.de/rechtschreibung/Lage#Bedeutung1a.

ECMA. 2013. Standard ECMA-404 - The JSON Data Interchange Format. [Online] 10 2013. [Zitat vom: 01. 03 2017.] https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

eMarketer. 2015. Anteil der Smartphone-Nutzer an allen Mobiltelefonnutzern in Deutschland von 2011 bis 2014 und Prognose bis 2019. *Statista - Das Statistik-Portal.* [Online] 08 2015. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/237079/umfrage/anteil-der-smartphone-nutzer-an-allen-mobilfunknutzern-in-deutschland/.

Europäische Kommission. 2016. Betriebsstart für Galileo! *Europäische Kommission - Pressemitteilung.* [Online] 14. 12 2016. [Zitat vom: 27. 01 2017.] http://europa.eu/rapid/press-release_IP-16-4366_de.htm.

Fette, I. und Melnikov, A. 2011. The WebSocket Protocol. [Online] 11 2011. [Zitat vom: 07. 02 2017.] https://tools.ietf.org/html/rfc6455.

Feuerwehr-Dienstvorschrift 100. 1999. *Führung und Leitung im Einsatz - Führungssystem.* Stuttgart : W. Kohlhammer Deutscher Gemeindeverlag, 1999. 978-3-555-01318-3.

Fielding, R., et al. 1999. Hypertext Transfer Protocol -- HTTP/1.1. [Online] 06 1999. [Zitat vom: 07. 02 2017.] http://www.ietf.org/rfc/rfc2616.txt.

Fowler, David. 2014. SignalR JS Client. *Github.* [Online] 27. 02 2014. [Zitat vom: 26. 02 2017.] https://github.com/SignalR/SignalR/wiki/SignalR-JS-Client.

Fowler, David und Edwards, Damian. 2017. SignalR. [Online] 10. 02 2017. [Zitat vom: 26. 02 2017.] http://signalr.net/.

Fowler, David und Girgin, Murat. o. J.. JabbR - Collaborative chat done right. [Online] o. J. [Zitat vom: 26. 02 2017.] http://about.jabbr.net/.

Friedman, Nat. 2013. Announcing Xamarin 2.0. *Xamarin Blog.* [Online] 02 2013. [Zitat vom: 18. 01 2017.] ttps://blog.xamarin.com/announcing-xamarin-2-0/.

Gentile, Camillo, et al. 2013. *Geolocation Techniques: Principles and Applications* . New York : Springer-Verlag, 2013. 978-1-4614-1835-1.

h3ko. 2017. DOWSER. [Online] 15. 02 2017. [Zitat vom: 16. 02 2017.] http://www.dowser-app.com.

HCI International. o. J.. 9th International Conference on Virtual, Augmented and Mixed Reality. *HCI International 2017.* [Online] o. J. [Zitat vom: 19. 01 2017.] http://2017.hci.international/vamr.

Holtkamp, Heiko. 2002. *Einführung in TCP/IP.* Technische Fakultät, Universität Bielefeld. Bielefeld : s.n., 2002.

Horizont. 2015. Wie häufig lassen Sie eine (GPS-)Ortung zu? *Statista - Das Statistik-Portal.* [Online] 08 2015. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/455346/umfrage/nutzungshaeufigkeit-der-ortungsfunktion-gps-durch-smartphone-besitzer/.

Horowitz, Ken. 2004. Sega VR: Great Idea or Wishful Thinking? *sega-16.* [Online] 28. 12 2004. [Zitat vom: 24. 01 2017.] http://www.sega-16.com/2004/12/sega-vr-great-idea-or-wishful-thinking/.

Icaza, Miguel de. 2011. Announcing Xamarin. *Personal blog of Miguel de Icaza.* [Online] 16. 05 2011. [Zitat vom: 18. 01 2017.] http://tirania.org/blog/archive/2011/May-16.html.

Initiative, D21 und ipima. 2016. Welche Bedenken haben Sie im Bereich Datenschutz/Datensicherheit, die Sie von einer (intensiveren) Nutzung von Online-Behördendiensten abhalten? *Statista - Das Statistik-Portal.* [Online] 09 2016. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/167114/umfrage/bedenken-beim-datenschutz-bei-egovernment-angeboten/.

International Committee on Global Navigation Satellite Systems. 2016. *The Way Forward - 10 Years of Achievement 2005-2015.* New York : UNITED NATIONS PUBLICATION Sales No. E.16.I.5, 2016. 978-92-1-101333-7.

International Organization for Standardization. 2004. *ISO -8601:2004 Data elements and interchange formats - Information interchange - Representation of dates and times.* Schweiz : s.n., 2004.

ITWissen. o.J.. Signallaufzeit. *itwissen.info.* [Online] o.J. [Zitat vom: 01. 02 2017.] http://www.itwissen.info/definition/lexikon/Signalausbreitungsgeschwindigkeit-signal-propagation-speed.html.

Jonsson, Prof. Jan. 2015. Vorlesungsfolien: Parallel & Distributed Real-Time Systems. *Chalmers Universe of Technology.* [Online] 2015. [Zitat vom: 06. 02 2017.] http://www.cse.chalmers.se/edu/year/2015/course/EDA421_Parallel_and_Distributed_Real_Time_Systems/Documents/Slides/Slides_1.pdf.

Kantar. 2017. Marktanteile der mobilen Betriebssysteme am Absatz von Smartphones in Deutschland von Oktober bis Dezember in den Jahren 2015 und 2016. *Statista - Das Statistik-Portal.* [Online] 02 2017. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/198435/umfrage/marktanteile-der-smartphone-betriebssysteme-am-absatz-in-deutschland/.

Koch, Robert, Golling, Mario und Rodosek, Gabi Dreo. 2013. *Advanced Geolocation of IP Addresses.* Deutschland : International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:7, No:8, 2013.

Land Berlin. 2006. Allgemeines Sicherheits- und Ordnungsgesetz - ASOG Bln. [Online] 11. 10 2006. [Zitat vom: 17. 03 2017.] http://www.gesetze.berlin.de/jportal/portal/t/117a/page/bsbeprod.psml?pid=Dokumentanzeige&showdoccase=1&js_peid=Trefferliste&documentnumber=1&numberofresults=1&fromdoctodoc=yes&doc.id=jlr-ASOGBE2006rahmen&doc.part=X&doc.price=0.0#focuspoint.

—. 2003. Feuerwehrgesetz. [Online] 23. 09 2003. [Zitat vom: 17. 03 2017.] http://www.gesetze.berlin.de/jportal/portal/t/11g0/page/bsbeprod.psml?pid=Dokumentanzeige&showdoccase=1&js_peid=Trefferliste&documentnumber=1&numberofresults=1&fromdoctodoc=yes&doc.id=jlr-FeuerwGBErahmen&doc.part=X&doc.price=0.0#focuspoint.

—. 1999. Katastrophenschutzgesetz. [Online] 11. 02 1999. [Zitat vom: 17. 03 2017.] http://www.gesetze.berlin.de/jportal/?quelle=jlink&query=KatSchG+BE&psml=bsbeprod.psml&max=true.

Land Berlin, BerlinOnline Stadtportal GmbH & Co. KG. 2015 b. LoD2 Gebäudedaten Berlin.

Berlin.de. [Online] 22. 05 2015 b. [Zitat vom: 18. 01 2017.] http://daten.berlin.de/datensaetze/lod2-geb%C3%A4udedaten-berlin.

Land Berlin; BerlinOnline Stadtportal GmbH & Co. KG. 2015 a. Berlin 3D - Stadtmodell als Open Data. *Berlin.de.* [Online] 22. 05 2015 a. [Zitat vom: 18. 01 2017.] https://daten.berlin.de/interaktion/artikel/berlin-3d-stadtmodell-als-open-data.

Landeshauptstadt Dresden. 2017. Virtuelles 3D-Stadtmodell. *Dresden.* [Online] 04. 01 2017. [Zitat vom: 18. 01 2017.] http://www.dresden.de/de/leben/stadtportrait/statistik/geoinformationen/3-d-modell.php?shortcut=3D.

Leach, P., Mealling, M. und Salz, R. 2005. A Universally Unique IDentifier (UUID) URN Namespace. [Online] 07 2005. [Zitat vom: 01. 03 2017.] https://tools.ietf.org/html/rfc4122.

Microsoft. 2013 d. .NET Framework. [Online] Microsoft Corporation, 2013 d. [Zitat vom: 2013. Mai 23.] http://msdn.microsoft.com/de-de/vstudio/aa496123.

—. 2017. Get the user's location. *API Reference.* [Online] 25. 01 2017. [Zitat vom: 30. 01 2017.] https://msdn.microsoft.com/en-us/windows/uwp/maps-and-location/get-location.

—. 2016 a. Hardware Details. *Microsoft Dev Center.* [Online] 2016 a. [Zitat vom: 24. 01 2017.] https://developer.microsoft.com/de-DE/windows/holographic/hardware_details.

—. 2016 c. Microsoft HoloLens. [Online] 2016 c. [Zitat vom: 24. 01 2017.] https://www.microsoft.com/microsoft-hololens/de-de.

—. 2016 b. Why HoloLens. [Online] 2016 b. [Zitat vom: 24. 01 2017.] https://www.microsoft.com/microsoft-hololens/de-de/why-hololens.

Milgram, Paul und Kishino, Fumio. 1994. *A Taxonomy of Mixed Reality Visual Displays.* 1994.

Mono. o. J. a. Mono - Compatibility. [Online] o. J. a. [Zitat vom: 18. 01 2017.] http://www.mono-project.com/docs/about-mono/compatibility/.

—. o. J. b. Mono - Download. [Online] o. J. b. [Zitat vom: 18. 01 2017.] http://www.mono-project.com/download/.

Mullen, N. Taylor. o. J.. ShootR. [Online] o. J. [Zitat vom: 26. 02 2017.] http://shootr.signalr.net/.

OASIS. 2014. MQTT Version 3.1.1 becomes an OASIS Standard. [Online] 30. 10 2014. [Zitat vom: 07. 02 2017.] https://www.oasis-open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard.

Oculus. 2012. Oculus Rift: Step Into the Game. *Kickstarter.* [Online] 2012. [Zitat vom: 24. 01 2017.] https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game.

OWIN Working Group. 2012. OWIN - Open Web Interface for .NET. [Online] 10. 10 2012. [Zitat vom: 26. 02 2017.] http://owin.org/.

Postel, Jon. 1981. RFC 793: Transmission Control Protocol. [Online] 09 1981. [Zitat vom: 06. 02 2017.] https://tools.ietf.org/html/rfc793.

—. 1980. RFC: User Datagram Protocol. [Online] 28. 07 1980. [Zitat vom: 06. 02 2017.] https://www.ietf.org/rfc/rfc768.txt.

Prof. Dr. Jörn von Lucke, Christian P. Geiger, M.A. 2010. *Open Government Data Frei verfügbare Daten des öffentlichen Sektors Gutachten für die Deutsche Telekom AG zur T-City Friedrichshafen Version vom 03.12.2010.* Lehrstuhl für Verwaltungs- und Wirtschaftsinformatik, Zeppelin Universität. Friedrichshafen : s.n., 2010.

Rotterdam Open Data Store. o. J.. Rotterdam 3d. *Rotterdam Open Data Store.* [Online] o. J. [Zitat vom: 18. 01 2017.] http://rotterdamopendata.nl/dataset/rotterdam-3d.

Samsung. o. J.. Galaxy S7 Spezifikation. [Online] o. J. [Zitat vom: 27. 01 2017.] http://www.samsung.com/de/smartphones/galaxy-s7/more/#!/spec.

Schnabel, Patrick. o.J.. Ortung und Positionsbestimmung mit Mobilfunk. *Elektronik Kompendium.* [Online] o.J. [Zitat vom: 01. 02 2017.] http://www.elektronik-kompendium.de/sites/kom/1201061.htm.

Schwiertz, E. 2017. *Erstellung eines zielgruppenorientierten Interfacedesigns am Beispiel „DOWSER“.* Berlin, 20. 03 2017.

Skyhook. o.J.. It all started with Wi-Fi. [Online] o.J. [Zitat vom: 30. 01 2017.] http://www.skyhookwireless.com/about-skyhook.

Stanford-Clark, Dr. Andy J. o.J.. Frequently Asked Questions. *MQTT.org.* [Online] o.J. [Zitat vom: 07. 02 2017.] http://mqtt.org/faq.

Stankovic, John A. 1988. *Misconceptions About Real-Time Computing.* University of Massachusetts. Los Alamitos, CA, USA : IEEE Computer Society Press, 1988.

Sutherland, Ivan E. 1968. A Head-Mounted Three-Dimensional Display. *American Federation of Information Processing Societies: Proceedings of the AFIPS '68 Fall Joint Computer Conference, December 9-11, 1968, San Francisco, California, USA - Part I.* s.l. : AFIPS / ACM / Thomson Book Company, Washington D.C., 1968, S. 757-764.

Transparenzportal Hamburg. 2014. 3D-Stadtmodell Hamburg. *Transparenzportal Hamburg.* [Online] 01. 09 2014. [Zitat vom: 18. 01 2017.] http://suche.transparenz.hamburg.de/dataset/3d-stadtmodell-hamburg.

United Nations. 2016 a. E–Government Development Index (EGDI) weltweit nach Ländern im Jahr 2016 (Top 20). *Statista - Das Statistik-Portal.* [Online] 07 2016 a. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/165817/umfrage/top-20-im-e%25E2%2580%2593government-development-index-weltweit/.

—. 2016 b. Entwicklung des E-Government Development Index (EGDI) in Deutschland in ausgewählten Jahren von 2005 bis 2016. *Statista - Das Statistik-Portal.* [Online] 07 2016 b. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/466763/umfrage/entwicklung-des-e-government-development-index-in-deutschland/.

VuMA (Arbeitsgemeinschaft Verbrauchs- und Medienanalyse). 2014. Anzahl der Personen in Deutschland, die ein Smartphone oder Handy mit GPS-Funktion besitzen, von 2010 bis 2014(in Millionen) . *Statista - Das Statistik-Portal.* [Online] 11 2014. [Zitat vom: 16. 02 2017.] https://de.statista.com/statistik/daten/studie/251941/umfrage/besitz-eines-smartphones-handys-mit-gps-funktion-in-deutschland/.

Wikibooks. 2010. Digitale Schaltungstechnik/ Signallaufzeit/ Funk. *Wikibooks.* [Online] 27. 06 2010. [Zitat vom: 01. 02 2017.] https://de.wikibooks.org/wiki/Digitale_Schaltungstechnik/_Signallaufzeit/_Funk.

—. 2015. Digitale Schaltungstechnik/ Signallaufzeit/ Leitungen. *wikibooks.org.* [Online] 22. 05 2015. [Zitat vom: 01. 02 2017.] https://de.wikibooks.org/wiki/Digitale_Schaltungstechnik/_Signallaufzeit/_Leitungen.

Wikipedia. 2017 a. .NET Core. *Wikipedia.* [Online] 01. 03 2017 a. [Zitat vom: 02. 03 2017.] https://de.wikipedia.org/wiki/.NET_Core.

—. 2017 d. .NET Framework. *Wikipedia.* [Online] 04. 02 2017 d. [Zitat vom: 02. 03 2017.] https://de.wikipedia.org/wiki/.NET_Framework.

—. 2016 a. Behörden und Organisationen mit Sicherheitsaufgaben. *Wikipedia.* [Online] 16. 11 2016 a. [Zitat vom: 16. 02 2017.] https://de.wikipedia.org/wiki/Beh%C3%B6rden_und_Organisationen_mit_Sicherheitsaufgaben.

—. 2016 c. Common Intermediate Language. *Wikipedia.* [Online] 28. 01 2016 c. [Zitat vom: 02. 03 2017.] https://de.wikipedia.org/wiki/Common_Intermediate_Language.

—. 2016 d. Just-in-time-Kompilierung. *Wikipedia.* [Online] 20. 09 2016 d. [Zitat vom: 02. 03 2017.] https://de.wikipedia.org/wiki/Just-in-time-Kompilierung.

—. 2017 b. Unity (Spiel-Engine). *Wikipedia.* [Online] 17. 02 2017 b. [Zitat vom: 02. 03 2017.] https://de.wikipedia.org/wiki/Unity_(Spiel-Engine).

—. 2016 b. Verkürzungsfaktor. *Wikipedia.* [Online] 17. 12 2016 b. [Zitat vom: 01. 02 2017.] https://de.wikipedia.org/wiki/Verk%C3%BCrzungsfaktor.

—. 2017 c. Xamarin. *Wikipedia.* [Online] 29. 01 2017 c. [Zitat vom: 02. 02 2017.] https://de.wikipedia.org/wiki/Xamarin.

Zaidenvoren, Pablo. o. J.. SignalR Java Client. *Github.* [Online] o. J. [Zitat vom: 26. 02 2017.] https://github.com/SignalR/java-client.

Zhang, Lucy. 2011. Building Facebook Messenger. *Facebook Notes.* [Online] 12. 07 2011. [Zitat vom: 07. 02 2017.] https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920/.

ABBILDUNGSVERZEICHNIS

Abbildung 1.1: DOWSER-Logo (h3ko, 2017) 1
Abbildung 2.2: Prototyp Headmounted Display (Sutherland, 1968 S. 758) 12
Abbildung 2.3: Funktionsweise Headmounted Display (Sutherland, 1968 S. 760) 13
Abbildung 2.4: Vereinfachte Darstellung des Virtualitätskontinuums nach Milgram und Kishino .. 14
Abbildung 2.5: Ergebnis einer Positionsermittlung durch IP2Geo.com 22
Abbildung 2.6: Server-Client-Interaktionsmodell 26
Abbildung 2.7: Polling-Verfahren 27
Abbildung 2.8: Long-Polling-Verfahren 28
Abbildung 2.9: Server-Sent Events-Verfahren 29
Abbildung 2.10: Forever-Frame-Verfahren 31
Abbildung 2.11: Kommunikation nach dem Websocket Protokoll 32
Abbildung 2.12: MQTT-Protokoll mit Subscibe-Publish-Verfahren 33
Abbildung 3.13: Zusammensetzung einer Einsatzleitung 34
Abbildung 3.14: SignalR - Virtuelle Verbindung 39
Abbildung 3.15: Umfrageergebnisse zur Häufigkeit der GPS-Ortung, (Horizont, 2015) 43
Abbildung 3.16: Umfrage zur Nutzung von Open Government 45
Abbildung 3.17: Übersicht der Anwendungsfälle für Einsatz (Ohne Persistenz) 48
Abbildung 3.18: Überblick über alle Komponenten 52
Abbildung 3.19: Schematische Darstellung des MVVM Entwurfsmusters 54
Abbildung 3.20: Aufbau einer Cross-Plattform-App 55
Abbildung 3.21: Aufbau der Klasse GeoLocation 56
Abbildung 3.22: Veranschaulichung HoloCommander-App 59
Abbildung 4.23: Technischer Aufbau Gesamtsystem 61
Abbildung 4.24: Konzerthaus Berlin in 3D im Downloadportal 64
Abbildung 4.25: Konzerthaus Berlin in 3D in Mashlab 65
Abbildung 4.26: UWP Unity Host 66
Abbildung 4.27: Bildschirmaufnahme HoloCommander mit Modellen des Gendarmenmarkts 67
Abbildung 4.28: Bildschirmaufnahme HoloCommander mit alternativen Modellen 68
Abbildung 4.29: Softwarearchitektur der Here I Am App 77
Abbildung 4.30: Bildschirmaufnahmen der HereIAm App für Android (links) und iOS (rechts) 79
Abbildung 4.31: Bildschirmaufnahme der App smartMap aus zwei Perspektiven 81

Tabellenverzeichnis

Tabelle 3.1: Gerätespezifikationen Microsoft HoloLens (Microsoft, 2016 a) 17
Tabelle 3.2: Übersicht über die bekanntesten Satellitennavigationssysteme 18
Tabelle 3.3: Verfahren zur Positionsbestimmung 21
Tabelle 3.4: Verkürzungsfaktoren verschiedener Signalleiter 23
Tabelle 3.5: Übersicht über die Frameworks zur Standortermittlung 38
Tabelle 3.6: Spezifikation der GeoLocation-Klasse 57
Tabelle 4.7: Öffentliche Schnittstellen des PositionHubs 74

Listings

Listing 3.1: Beispiel JSON api/location-Endpunkt 58
Listing 3.2: Beispiel XML api/location-Endpunkt 58
Listing 4.3: Unity Skript zur SignalR Steuerung 70
Listing 4.4: UWP Brücke zu Unity zur SignalR Steuerung 72
Listing 4.5: LocationDirect Controller des PositionHubs 76
Listing 4.6: Plattformunabhängiger XAML Code für iOS und Android UI 78