HTL Perg für EDV und Organisation

Machlandstraße 48

A – 4320 Perg

Diplomarbeit

HeartBeat

(Notfallsoftware für Personen in Not)

Projektteam:

Michael Mayr

David Oberögger

Betreuungslehrer:

Dipl.-Ing Michael Stumpfl

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg,_____Unterschrift_____

Perg,_____Unterschrift_____

Danksagung

Wir bedanken uns bei all jenen, die uns über den gesamten Projektverlauf hinweg in jeglicher Hinsicht unterstützt haben.

Besonderer Dank gilt unserem Betreuungslehrer Dipl.-Ing. Michael Stumpfl, der uns mit großem Engagement und Einsatz geholfen hat. Neben seiner Unterstützung in technischen Fragen konnte er uns auch durch seine Erfahrung im Schreiben von wissenschaftlichen Arbeiten sehr gut sein Wissen vermitteln.

Darüber hinaus bedanken wir uns bei allen Lehrkräften, die uns bei Fragen immer hilfsbereit unterstützt haben und so an dem Erfolg dieser Diplomarbeit teilhaben.

Herzlichen Dank!

Kurzfassung

HeartBeat (HB) ist eine Diplomarbeit, welche im Jahr 2015 im Rahmen der Matura an der HTL Perg implementiert wurde.

HeartBeat ist eine auf Android basierende Applikation, die helfen soll in Notfallsituationen eine schnelle und optimierte Hilfeleistung zur Verfügung zu stellen. Diese Applikation fungiert als Bindeglied zwischen Helfer und Verunglückten und dem Verunfallten werden überlebenswichtige Informationen zur Verfügung gestellt, um dieser Person bestmöglich zu helfen. Nutzer dieser Applikation, die sich im Umkreis eines Unfalles aufhalten, werden via Push-Nachrichten benachrichtigt und können somit auch zur Hilfe beitragen. Mit HeartBeat kann einfach und schnell zivile Hilfe verständigt werden, die oftmals schneller den Verunfallten betreuen kann als diverse Rettungsdienste oder Notfallärzte.

Schlagwörter: HeartBeat, Rettung, Android, Notfall, Gesundheit, Diplomarbeit

Abstract

HearBeat (HB) is a diploma thesis which was developed at the HTL Perg in 2015.

HeartBeat is an android based software which provides a fast and optimized assistance after an accident. This application act as connector between aides and casualty and offers lifesaving information to help the person optimally. User of this application who are near the accident get informed of it via Push Notifications and are now able to contribute to the rescue. With HeartBeat it is possible to get easy and fast access to civil aid which are often faster at the accident scene than emergency services or doctors.

Keywords: HeartBeat, rescue, android, emergency, health, diploma thesis

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ADT ..... *Android Development Tools*
APIs ..... *application programming interface*
CRUD ..... *create, read, update, delete*
CSV ..... Comma-spearated values
DHCP ..... *Dynamic Host Configuration Protocol*
DNS ..... *Domain Name System*
GCM ..... Google Cloud Messaging
GUI ..... *graphical user interface*
HB ..... *HeartBeat*
HTML5 ..... *Hypertext Markup Language*
HTTP ..... *Hypertext Transfer Protocol*
IBM ..... *International Business Machines*
IIS ..... *Internet Information Service*
JSON ..... *Javascript Object Notation*
LINQ ..... *Language Integrated Query*
MD5 ..... Message-Digest Algorithm 5
MDS ..... *Master Data Service*
MS SQL Server ..... *Microsoft Structured Query Language Server*
NTLM ..... *NT LAN Manager*
ODBC ..... *Open Database Connectivity*
ORM ..... *object-relational mapping*
POJOs ..... *Plain Old Java Object*
SDK ..... *Software Development Kit*
SQL ..... *Structured Query Language*
SSL ..... *Secure Sockets Layer*
Uri ..... Uniform Resource Identifier
URL ..... *Uniform Resource Locator*
WCF ..... *Windows Communication Fundation*
XAML ..... *Extensible Application Markup Language*

1 Einleitung

1.1 Ziel der Arbeit

Ziel dieser Diplomarbeit ist, eine Applikation zu implementieren, welche es ermöglicht, schnell und einfach professionelle und zivile Hilfe anzufordern. Dabei soll es möglich sein, dritte Personen in den Kreislauf der Rettung des Verunfallten einzubinden und diesen physisch bzw. psychisch zu unterstützen. Hauptaugenmerk haben wir auf die Optimierung des Rettungsablaufes gelegt, um wichtige Zeit einzusparen, welche im Notfall von äußerster Priorität ist. Zusätzlich ist es möglich, Kontaktdaten, Öffnungszeiten und Fahrplanrouten zu Hilfseinrichtungen wie Krankenhäuser, Rettungsdienste, Ärzte und Feuerwehren einzusehen.

1.2 Motivation

Persönliche Erfahrungen haben gezeigt, dass in der heutigen Zeit die Zivilcourage ein großes Problem darstellt und kaum noch jemand eine helfende Hand für Personen in Not hat. Die Software HeartBeat soll diesen Trend wieder in die richtige Richtung weisen, indem die Aufmerksamkeit auf Unfälle erhöht wird.

2 Grundlagen

Das Kapitel Grundlagen beschäftigt sich mit der Beschreibung der verwendeten Technologien, Entwicklungssysteme und der Systemarchitektur.

2.1 Technologien

2.1.1 ADO.NET

ADO.NET kümmert sich um den Zugriff auf verschiedene Datenquellen und wird verwendet, um sich zu diesen zu verbinden und Daten zu lesen, schreiben, ändern und löschen.

2.1.1.1 ADO.NET Architektur

Diese Architektur besteht aus 2 Komponenten. Den .NET Framework Data Provider und das DataSet.

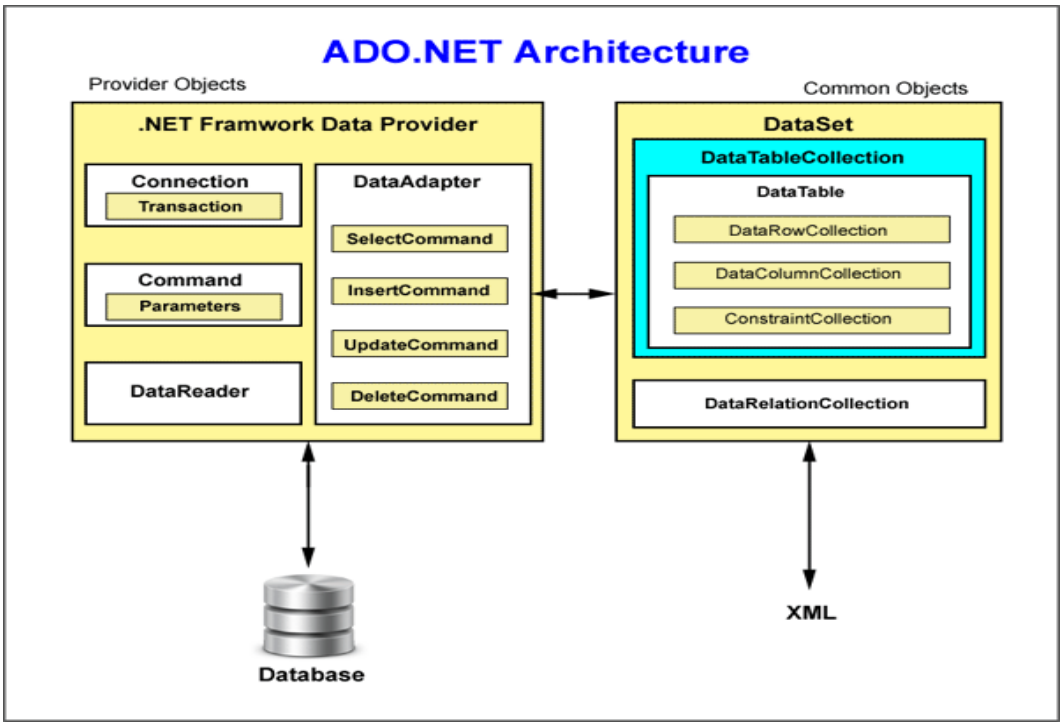

Abbildung 1: ADO.NET Architektur (aus [27])

2.1.1.2 .NET Framework Data Provider

Diese Komponente ist für die Datenbearbeitung und den schnellen Zugriff auf Daten verantwortlich. Dieser Data Provider verfügt über 4 Objekte.

- Connection
  - Das Connection-Objekt ist für die Verbindung mit einer Datenquelle zuständig.
- Command
  - Mit diesem Objekt können auf Datenbankbefehle zugegriffen werden und Daten verändert, gelesen und gespeichert werden.
- DataReader
  - Der DataReader stellt Daten mit höchstmöglicher Geschwindigkeit aus der Datenquelle zur Verfügung.
- DataAdapter
  - Der DataAdapter führt über Command-Objekte SQL-Befehle aus und lädt Daten in die Komponente DataSet.

2.1.1.3 DataSet

Die Komponente DataSet enthält Datenmengen von der Datenquelle und fungiert als Speicherabbild der eigentlichen Datenbank. Das DataTable-Objekt soll eine Datebanktabelle darstellen und beinhaltet wie eine Datenbanktabelle Datensätze und Spalten (vgl. [1]).

2.1.2 Microsoft SQL Server

Der MS SQL Server ist eine relationale Datenbank, die im Jahr 1989 von Microsoft entwickelt wurde. Der Microsoft SQL Server beinhaltet eine Vielzahl von Datenverwaltungs- und Analysetechnologien.

2.1.2.1 Datenbankmodul

Das Datenbankmodul ist für das Speichern, Sichern und Verarbeiten der Daten verantwortlich. Mit diesem Modul können schnelle Transaktionen durchgeführt werden und es unterstützt den Nutzer, die Daten durchgehend zur Verfügung zu stellen.

2.1.2.2 Analysis Services

Dies ist ein Tool, um Daten, welche auf verschiedene Datenquellen verbreitet sind, zu analysieren und Informationen aus diesen zu gewinnen. Ebenfalls trägt es dazu bei, bei Daten aus verschiedenen Tabellen Zusammenhänge zu erkennen.

2.1.2.3 Integration Services

Mit dem Integration Service können Daten aus unterschiedlichen Datenquellen, z.B. JSON-Datendateien, CSV und relationale Datenbanken extrahiert, transformiert und in

bestimmte Ziele geladen werden. Dies heißt, dass sich der Nutzer nicht mehr um das Zusammentragen der Daten aus verschiedenen Quellen kümmern muss.

2.1.2.4 Master Data Services (MDS)

Dies ist ein Produkt von Microsoft, um Datenquellen zu zentralisieren und synchron zu halten. Es wird ebenfalls eine Transaktionsaufzeichnung zur Verfügung gestellt, um alle Änderungen, die auf dieser zentralen Datenquelle vollzogen wurden, aufzuzeichnen. Mit den MDS können auch verschiedene Versionen dieser Daten angelegt werden und gegebenenfalls auf diese zurückgegriffen werden. Da nicht alle Nutzer, die Zugriff auf die Datenbank haben, jede Berechtigung haben dürfen, stellt MDS eine rollenbasierte Sicherheitsfunktion zur Verfügung, um Bereiche und Nutzer einzuschränken.

2.1.2.5 Replikation

Die Replikationsfunktion bietet Technologien zum Kopieren, Auslagern und Verteilen der Daten an. Mit der Replikation können Daten über verschiedene Verbindungen verteilt und synchronisiert werden.

2.1.2.6 Reporting Services

Diese Services beinhalten eine Vielzahl von Funktionen, damit Unternehmen Berichte erstellen und die Sicherheit verwalten können (vgl. [2]).

2.1.3 ADO.NET Entity Framework

Mit dem Entity Framework können Anwendungen erstellt werden, die Objektmodelle anstelle von relationalen Datenmodellen verwenden. Mit dem ADO.Net Entity Framework wird ein großer Teil des Codes erspart, der für datenorientierte Applikationen benötigt werden würde (vgl. [3]).

Mit dem ADO.Net Entity Framework werden Abfragen nicht direkt an die Datenbank, sondern an ein Datenmodell abgegeben, welches das Entity Framework in Form von Objekten bereitstellt. Dazu muss die relationale Datenbank in ein Objektmodell überführt werden, wobei dieser Vorgang als Object-Relational Mapping (ORM) bezeichnet wird. Das Ergebnis einer Datenbankabfrage wird in Form von Objekten an das aufrufende Programm übergeben (siehe 2.1.3.1).

2.1.3.1 Object-Relational Mapping (ORM)

ORM ermöglicht objektorientierten Programmen Objekte in relationalen Datenbanksystemen zu speichern und zu verwalten (siehe Abbildung 2).

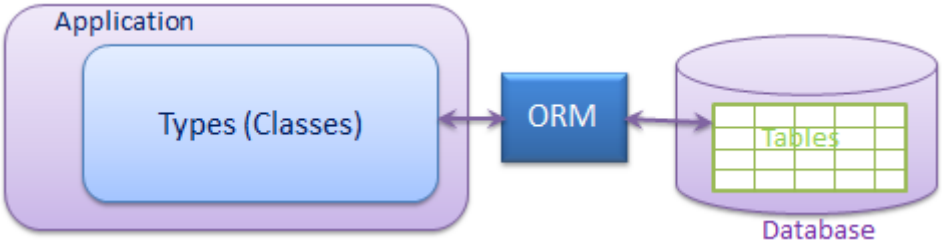

Abbildung 2: ORM Architecture (aus [3])

2.1.3.2 Entity Framework Überblick

Das Entity Framework bietet verschiedene Möglichkeiten, dieses Framework in die Anwendung einzubinden.

Code First

Hier wird zuerst der Code geschrieben, welcher durch das Entity Framework die Datenbank und deren Tabellen beschreibt. Konkret heißt dies, dass zuerst die entsprechenden Entitätsklassen geschrieben werden und anhand dieser durch Code First die Datenbank mitsamt des Datenbankmodelles erstellt wird (siehe Abbildung 3).

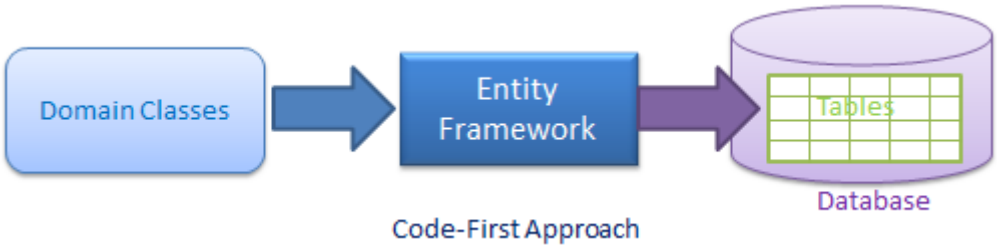

Abbildung 3: Code First (aus [3])

Model First

Im Model First werden die Entitäten, Beziehungen und Hierarchien in einem visuellen Designer definiert. Anschließend wird aus dieser Eingabe ein Datenbankmodell generiert (siehe Abbildung 4).

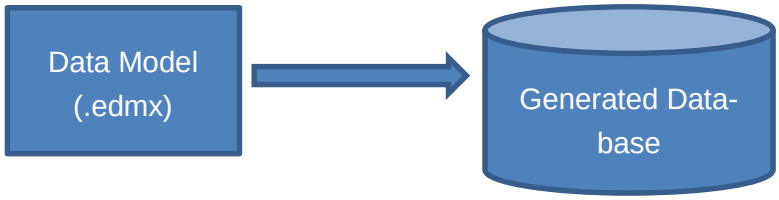

Abbildung 4: Model First

Database First

Bei Database First ist eine bestehende Datenbank vorhanden und aus dieser werden mit dem Entity Framework Entitätsklassen erzeugt (siehe Abbildung 5).

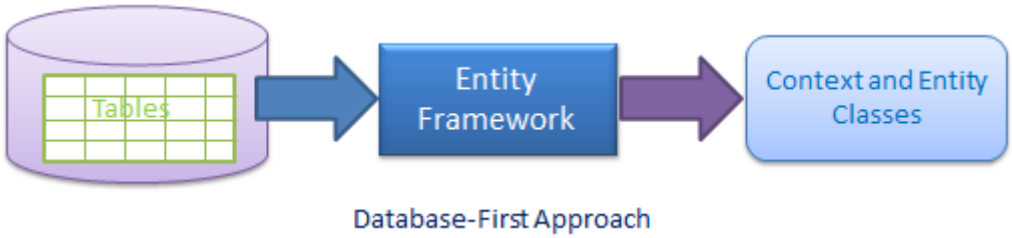

Abbildung 5: Database First (aus [3])

Zusammenfassung

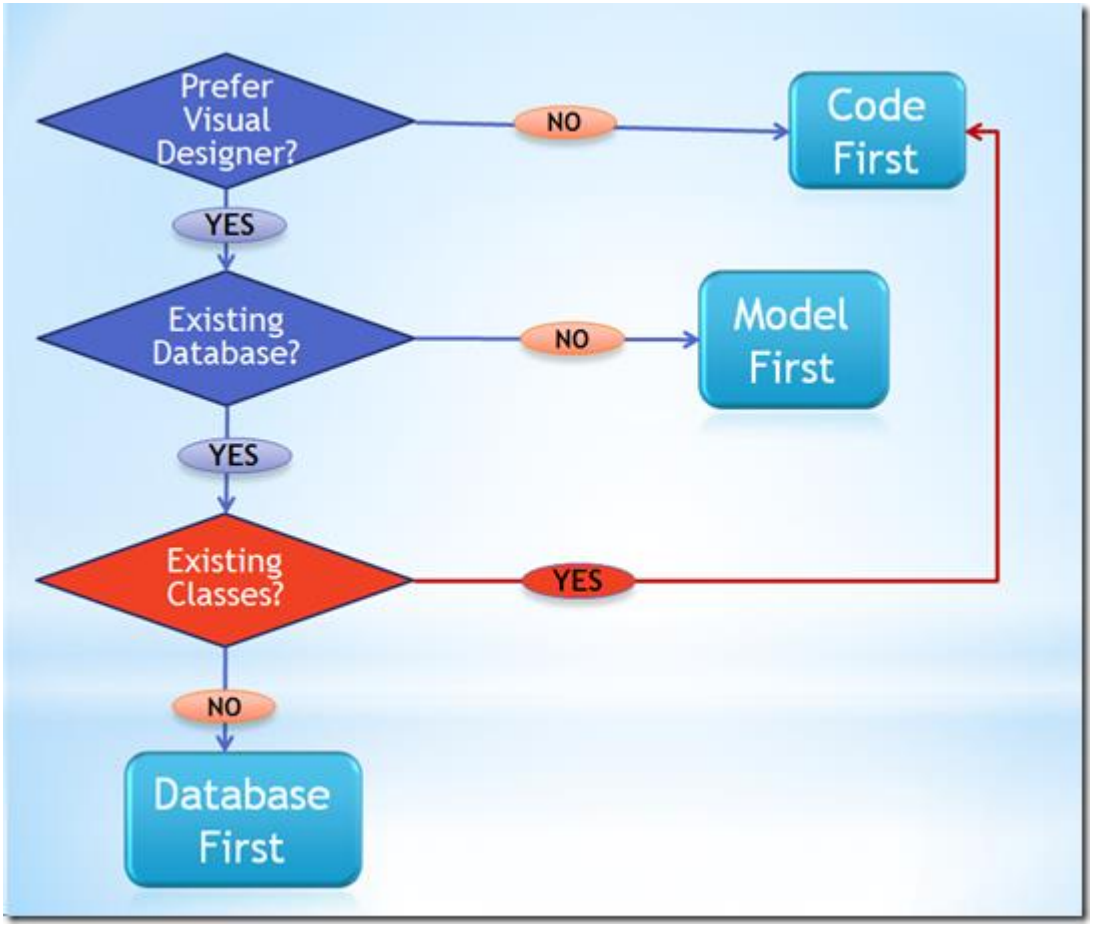

Abbildung 6: Decision tree (aus [28])

2.1.4 Windows Communication Fundation (WCF)

Das Windows Communication Fundation ist ein Framework, welches es ermöglicht, Daten über verschiedene Plattformen zu senden. Dieses Framework besteht aus 5 Komponenten (siehe Abbildung 7).

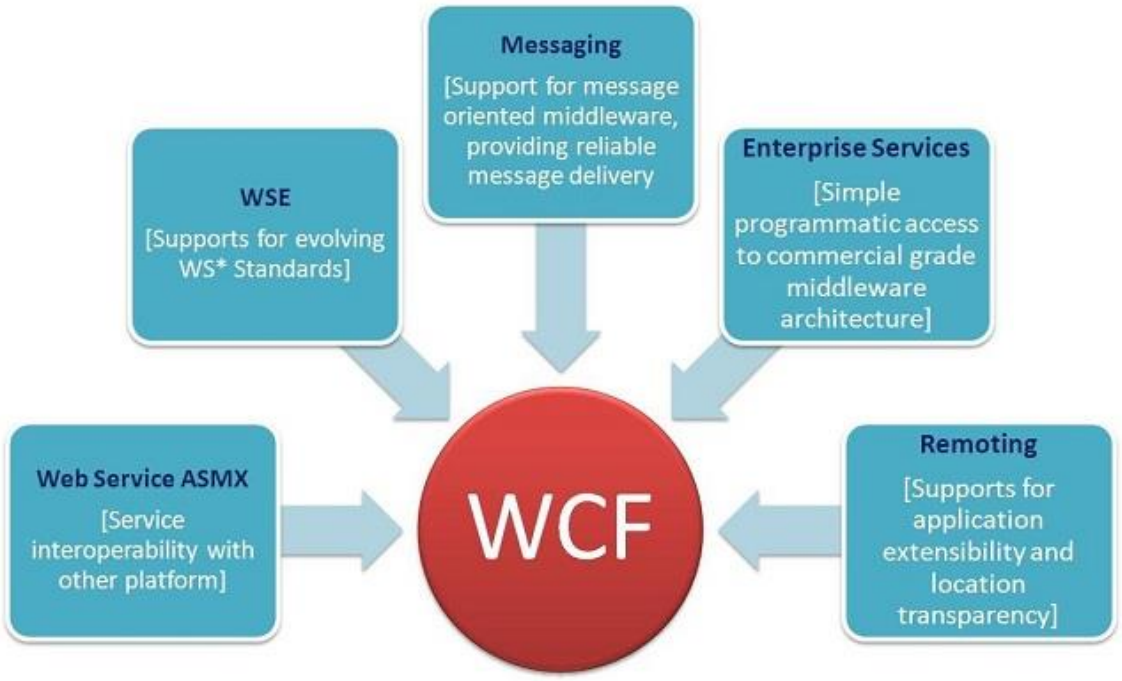

Abbildung 7: WCF Background, (aus [4])

2.1.4.1 Web Service

Ein WCF Service fungiert als Anlaufstelle für Clientanwendungen. Der Client kann Methoden, die das WCF Service zur Verfügung stellt, aufrufen und über das Service CRUD Operationen auf die Datenbank absetzen (vgl. [5]).

2.1.5 Language Integrated Query (LINQ)

LINQ ist eine Abfragesprache, welche in C# und Visual Basic integriert ist. Sie ermöglicht mit SQL ähnlichen Zeichenketten Daten abzufragen und zu aktualisieren. LINQ Abfragen können im ADO.NET Framework auf DataSets, SQL Server und auf Entitäten durchgeführt werden (vgl. [6]).

2.1.5.1 Syntax

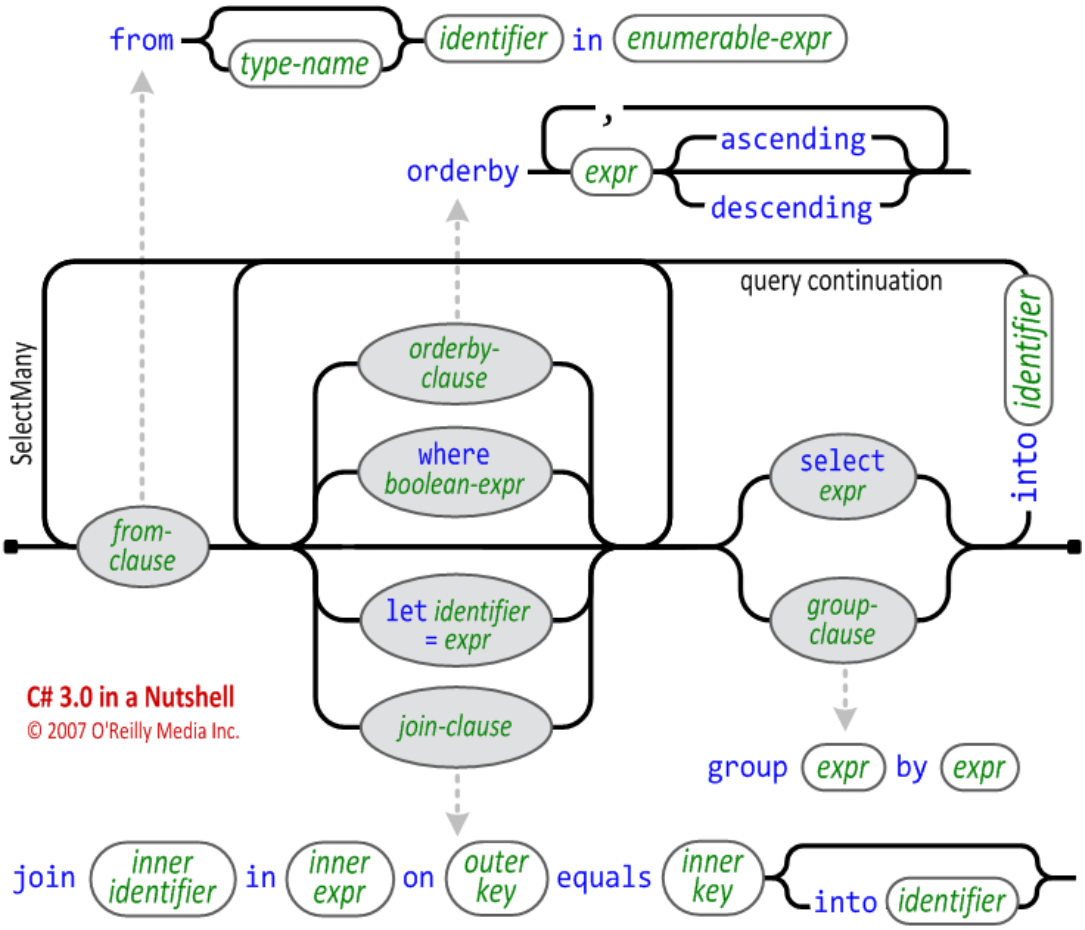

Abbildung 8: LINQ Syntax, (aus [7])

2.1.5.2 Select

```
List<Person> people = GetPeopleList();

 var peopleNames =
     from p in people
     select p.PersonName;
```

Beispiel 1: LINQ Select

Die Select-Klausel erlaubt es, Daten aus verschiedenen Datenquellen auszuwählen.

2.1.5.3 Where

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

     var lowNums =
         from n in numbers
         where n < 5
         select n;
```

Beispiel 2: LINQ Where

Die Where-Klausel erlaubt es, die Datenmenge einzuschränken. In diesem Beispiel heißt es konkret, dass alle Nummern selektiert werden, welche kleiner als 5 sind.

2.1.5.4 Funktionen

LINQ unterstützt eine Vielzahl von Funktionen, welche Quellcode ersparen sollen. Beispiele für solche Funktionen sind:

- Count
- Sum
- Min
- Max

2.1.6 Extensible Application Markup Language (XAML)

XAML ist eine deklarative Markupsprache. Bei Anwendung auf das .NET Framework-Programmiermodell vereinfacht XAML das Erstellen einer Benutzeroberfläche für eine .NET Framework-Anwendung. Es können sichtbare Benutzeroberflächen-Elemente im deklarativen XAML-Markup erstellen werden und anschließend die Benutzeroberflächen-Definition mithilfe von Code-Behind-Dateien, die über partielle Klassendefinitionen an das Markup geknüpft sind, von der Laufzeitlogik trennen. XAML stellt die Instanziierung von Objekten in einem spezifischen Satz von in Assemblys definierten Unterstützungstypen direkt dar (vgl. [8]).

2.1.7 Internet Information Service (IIS)

Auf dem IIS können verschiedene Webanwendungen und Dienste gehostet werden, um diese im Internet zur Verfügung zu stellen.

2.1.7.1 Architektur

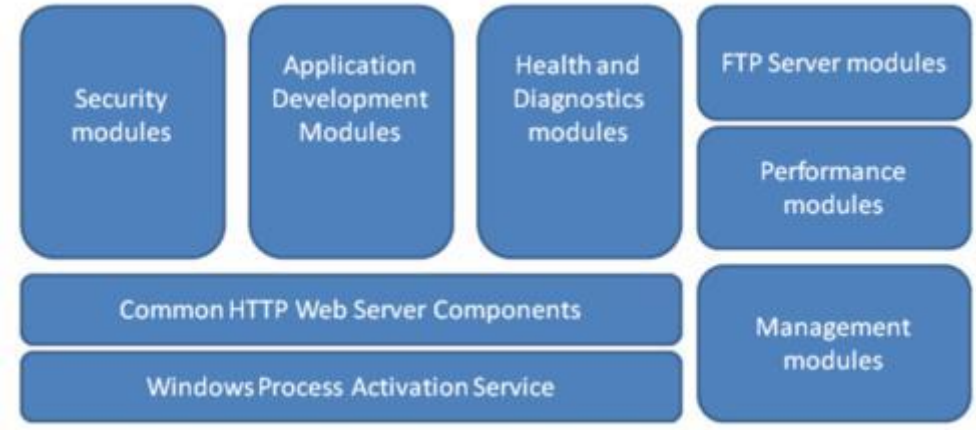

Abbildung 9: IIS Architektur, (aus [9])

2.1.7.2 Verwaltung

Um diese Dienste und Webanwendungen zu verwalten, stellt Microsoft den IIS Manager zur Verfügung. Der IIS Manager wird verwendet, um IIS und ASP.NET Einstellungen zu konfigurieren. Ebenfalls werden Tools zur Überwachung der gehosteten Anwendungen zur Verfügung gestellt (siehe Abbildung 10).

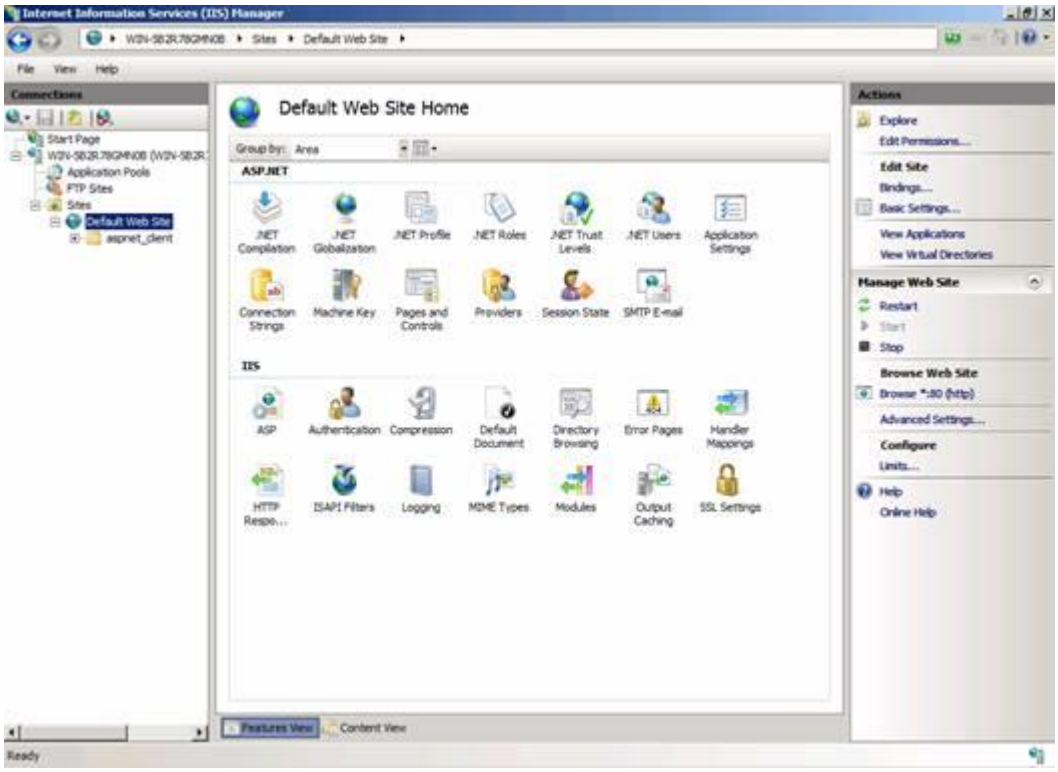

Abbildung 10: IIS Manager Home, (aus [9])

2.1.7.3 Sicherheit

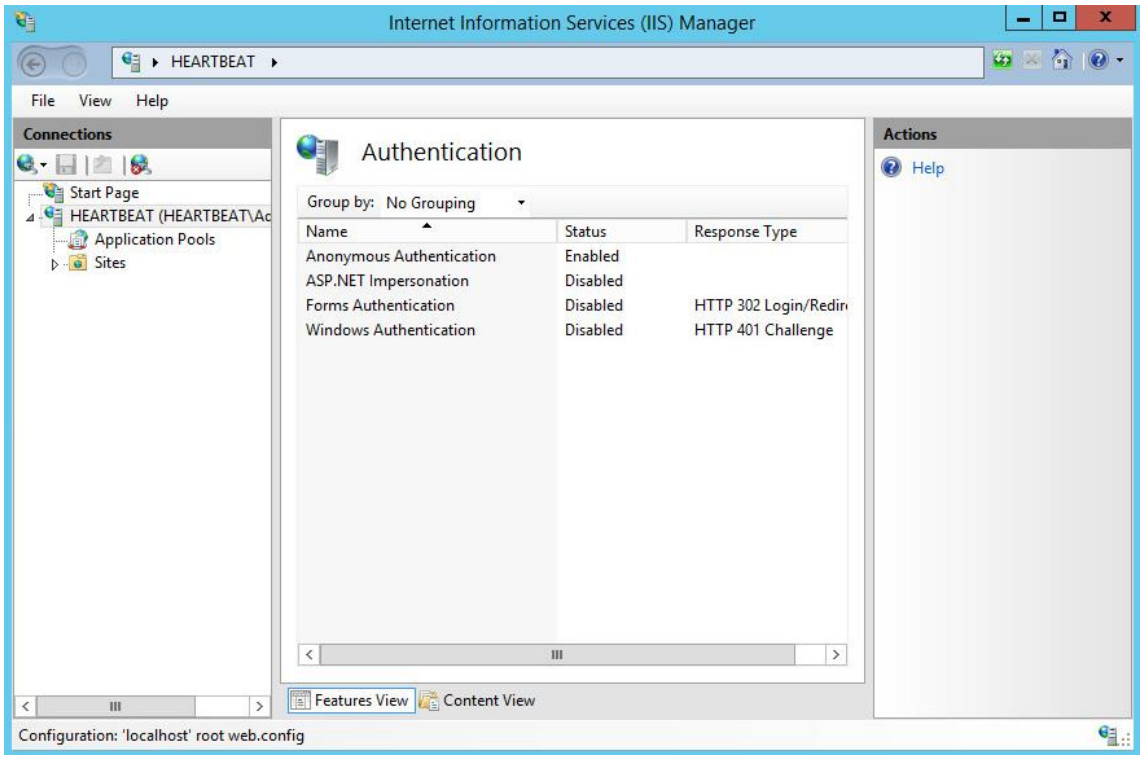

Abbildung 11: IIS Authentifizierungsmethoden

Anonymous Authentication

Mit der anonymen Authentifizierung werden sämtliche Inhalte öffentlich zugänglich gemacht, ohne dass Benutzernamen oder Passwort abgefragt werden. Diese Authentifikation haben wir deshalb verwendet, da diese einfach zu implementieren ist und zum Großteil keine sensiblen Daten versendet werden.

ASP.NET Impersonation Authentication

Diese Authentifizierungsart ermöglicht, dass sich der zu Authentifizierende mit verschiedenen Accounts anmelden kann. Diese verschiedenen Accounts haben oft andere Berechtigungen und deshalb muss über ASP.NET Impersonation ein Identitätswechsel durchgeführt werden.

Basic Authentication

Basic Authentication ist weit verbreitet. Diese Art funktioniert mit Benutzernamen und Passwort, welche aber unverschlüsselt im Netzwerk herumgeschickt werden. Basic Authentication wird oftmals in Verbindung mit SSL verwendet, um die Sicherheit der Accounts zu gewährleisten.

Client Certificate Mapping Authentication

Diese Methode ermöglicht dem User, sich mit Zertifikaten bei der Authentifizierungsstelle zu authentifizieren. Dadurch werden keine Basic, Digest und ASP.NET Impersonation Authentifizierungen benötigt.

Digest Authentication

Die Digest Authentifizierung basiert auf dem gleichen Prinzip wie die Basic Authentication, mit dem Unterschied, dass die Daten mit einem MD5 Hashwert verschlüsselt über das Netzwerk gesendet werden.

Forms Authentication

Benutzerdaten werden wieder unverschlüsselt im Netzwerk versendet. Es sind Benutzername und Passwort notwendig, aber da diese unverschlüsselt sind, wird diese Methode oftmals in Verbindung mit Verschlüsselungsprotokollen wie SSL verwendet.

Windows Authentication

Bei der Windows Authentication sind Benutzernamen und Passwort verschlüsselt und sicher über das Netzwerk versandt. Es gibt 2 verschiedene Authentifizierungsprotokolle (Kerberos und NTLM). Die Windows Authentifizierung ist sehr sicher, da die Verschlüsselung und die Überprüfung der Benutzerdaten vom Windowsrechner übernommen wird (vgl. [10]).

2.1.8 Android

Android ist ein Betriebssystem für mobile Geräte und liegt derzeit in der Version 5.1 vor. Android ist eine Open-Source Software, welche unter Apache 2.0 lizensiert ist (vgl. [11]).

2.1.8.1 Basisdaten

Tabelle 1: Android Daten (aus [12])

Entwickler	**Open Handset Alliance**
Sprache	Mehrsprachig (> 68 Sprachen)
Aktuelle Version	5.1 Lollipop
Abstammung	Linux
Kernel	Monolithisch (Linux)
Lizenz	Apache 2.0, GPLv2

2.1.8.2 Architektur

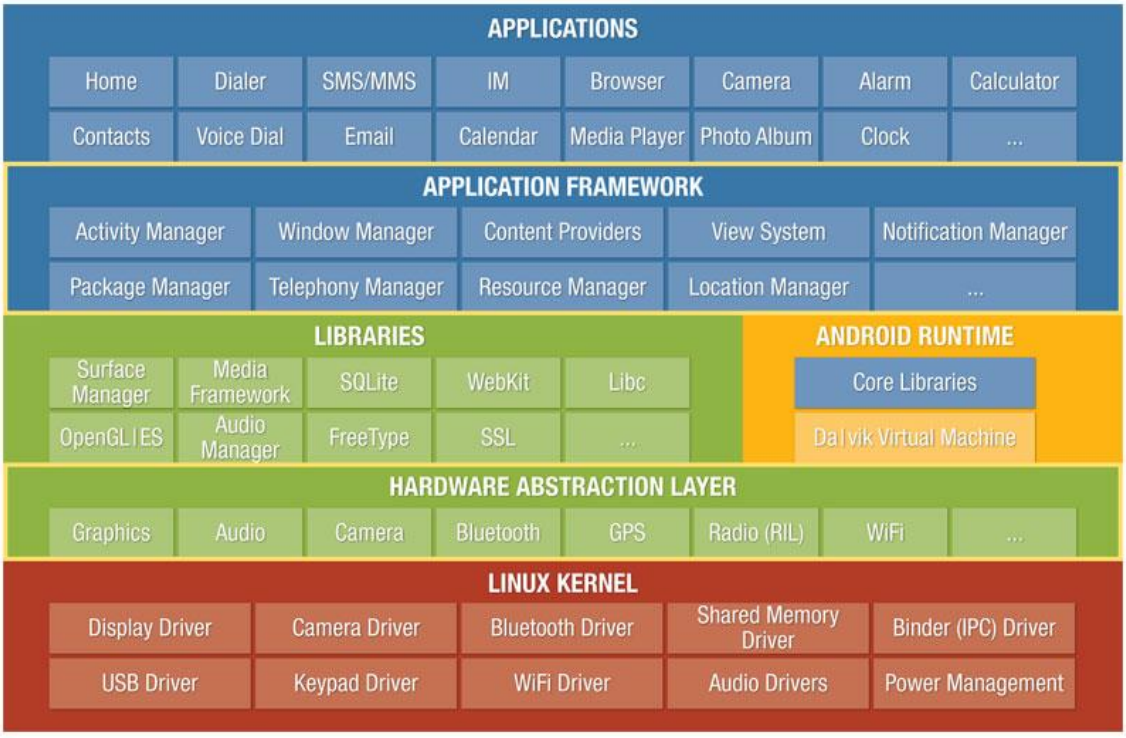

Abbildung 12: Android Architektur, (aus [13])

Linux Kernel

Die unterste Ebene des Android Betriebssystems bildet der Linux Kernel. Der Kernel stellt Multitasking, Prozessoren- und Energiemanagement zur Verfügung. Ebenfalls werden Gerätetreiber für Display, Wi-Fi und Audio bereitgestellt. Der originale Linux Kernel wurde 1991 von Linus Torvalds entwickelt und stellt die Grundlage für viele Linux Distributionen dar.

Android Runtime – Dalvik Virtual Machine

In Android läuft jede Applikation in einer eigenen Instanz der Dalvik Virtual Machine. Dies schränkt die Applikation ein und verhindert, dass Anwendungen mit anderen Anwendungen kommunizieren oder direkt auf die Hardware zugreifen.

Applikation Framework

Dieses Framework besteht aus mehreren Diensten, die das Android Applikationen verwalten (vgl. [13]).

wichtige Dienste:

- Activity Manager
  - Dieser kontrolliert den Lebenszyklus einer Applikation.
- Content Provider
  - Dieser erlaubt der Applikation mit anderen zu kommunizieren.
- Resource Manager
  - Dieser ermöglicht den Zugriff auf Ressourcen wie Farbeinstellungen, Strings und Views.
- Notification Manager
  - Applikationen können mit dem Notification Manager Warnungen und Notifikationen anzeigen
- View System
  - Das View System ermöglicht, Benutzerinterfaces zu erstellen.
- Package Manager
  - Applikationen können mit diesem Dienst Informationen über derzeit installierte Dienste und anderen Apps herausfinden.
- Telephony Manager
  - Dieser Manager liefert Informationen über Telefondienste.
- Location Manager
  - Dies ist ein wichtiger Standortdienst, welcher Apps den Zugriff auf Lage und Position des Gerätes ermöglicht.

2.1.9 JavaScript Object Notation (JSON)

JSON ist ein schlankes Datenaustauschformat, das für Menschen einfach zu lesen und zu schreiben und für Maschinen einfach zu parsen und zu generieren ist. JSON ist ein Textformat, welches unabhängig von Programmiersprachen ist, aber vielen Konventionen folgt (vgl. [14]).

JSON besteht aus 2 Strukturen:

- Name – Wert
  - kann mit einer Hash-Table verglichen werden
- Liste von Werten
  - kann mit Arrays bzw. Listen verglichen werden

2.1.9.1 JSON Typen

Objekt

Das Objekt ist eine Vielzahl von Namen und Wert Paaren, welche aber ungeordnet gespeichert sind. Ein JSON Objekt startet mit "{" und enthält eine Menge von Name – Wert Paaren, welche durch einen Beistrich getrennt sind. Innerhalb dieser Paare werden Name und Wert durch einen Doppelpunkt voneinander getrennt. Das JSON Objekt wird zum Schluss mit "}" wieder abgeschlossen.

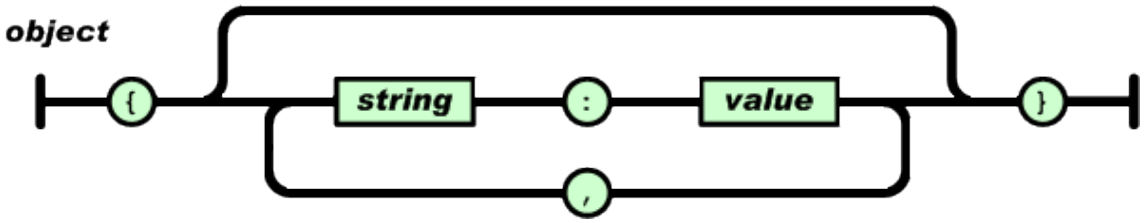

Abbildung 13: Objekt Syntax, (aus [14])

Array

Ein Array enthält eine geordnete Liste von Werten. Sie wird mit eckigen Klammern abgegrenzt und die Werte werden mit einem Beistrich getrennt.

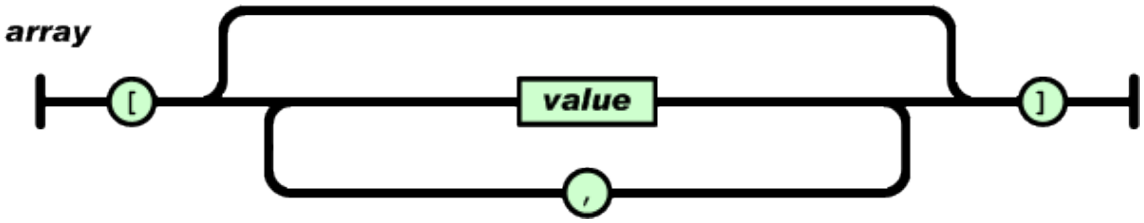

Abbildung 14: Array Syntax, (aus [14])

Wert

Ein Wert kann verschiedene Typen annehmen:

- Zeichenkette
- Nummer
- Objekt
- Array
- Boolean

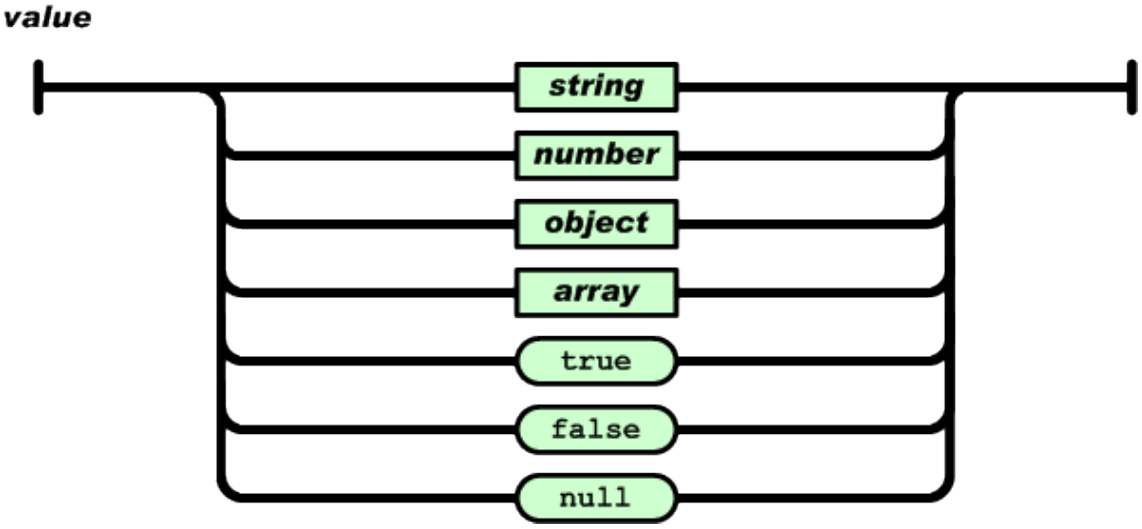

Abbildung 15: Wert Syntax, (aus [14])

Zeichenkette

Die Zeichenkette besteht aus 0 bis n Unicode-Zeichen, welche von doppelten Hochkommata eingeschlossen sind. Es werden bestimmte Operationen unterstützt, welche mit Backslash angekündigt werden und mit einem bestimmten Unicode-Zeichen bestimmt werden.

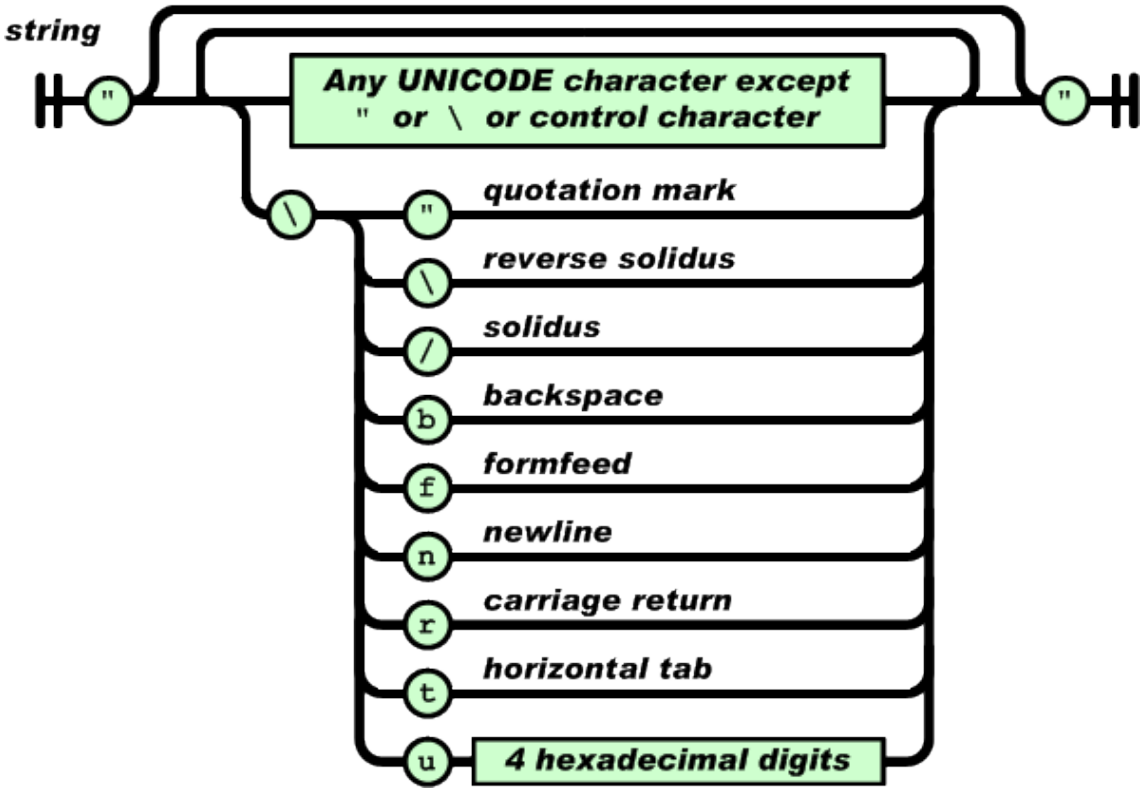

Abbildung 16: Zeichenkette Syntax, (aus [14])

Nummer

JSON unterstützt jede Art von Zahlen mit Ausnahme von oktalen und hexadezimalen Zahlen.

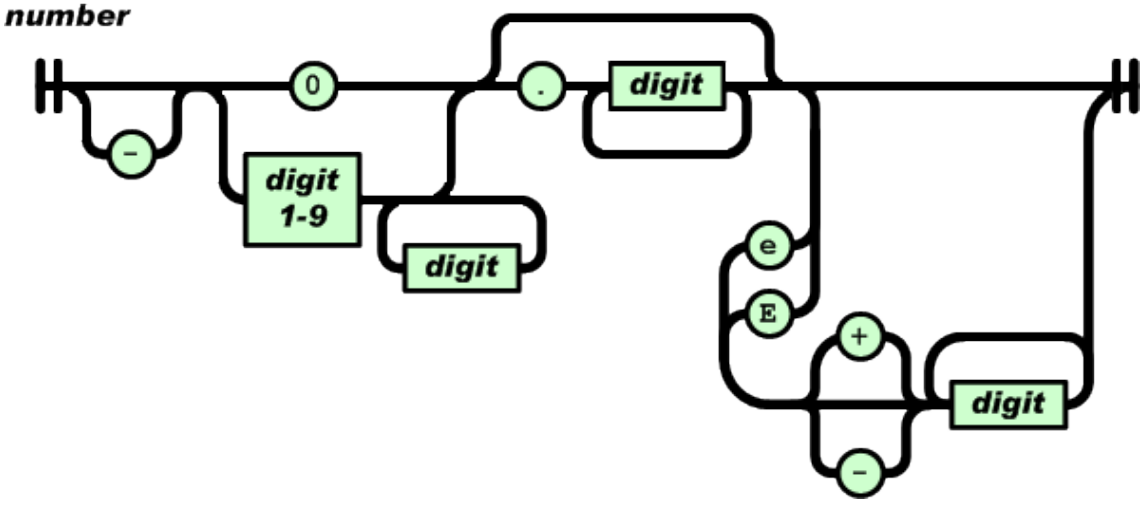

Abbildung 17: Nummer Syntax, (aus [14])

2.1.10 Google APIs

Google stellt eine Vielzahl von Schnittstellen zur Verfügung, welche das Programmieren und Zugreifen auf Google Dienste erheblich vereinfacht. Google bietet für die Verwaltung dieser APIs ein Webinterface an (siehe Abbildung 18).

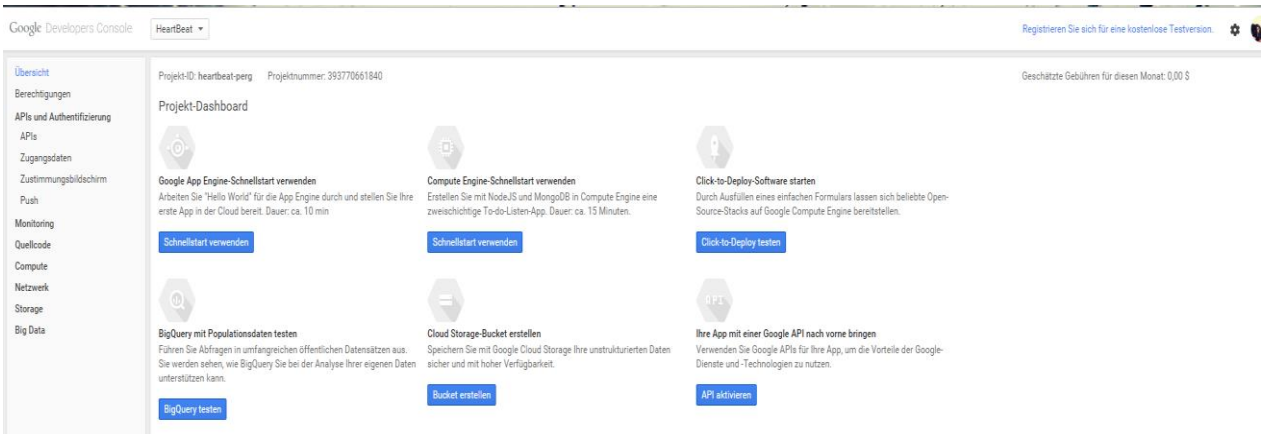

Abbildung 18: Google Entwicklerkonsole

2.1.10.1 Authentifizierung

Um diese Dienste verwenden zu können, muss sich die Anwendung bei Google registrieren. Dies geschieht unter dem Reiter „APIs und Authentifizierung“ → „Zugangsdaten“. Hier werden API-Schlüssel erstellt, mit denen die Anwendung bemächtigt wird, die Google Dienste zu nutzen.

2.1.10.2 Google Cloud Messaging (GCM)

Google Cloud Messaging ist ein wichtiger Bestandteil für die Diplomarbeit HeartBeat. Es ermöglicht Daten vom Server an Android Geräte zu senden. Damit können wir mit einer eindeutigen GCM ID, welche dem Android Gerät zugeordnet ist, das Gerät in Echtzeit ansteuern.

Aufbau

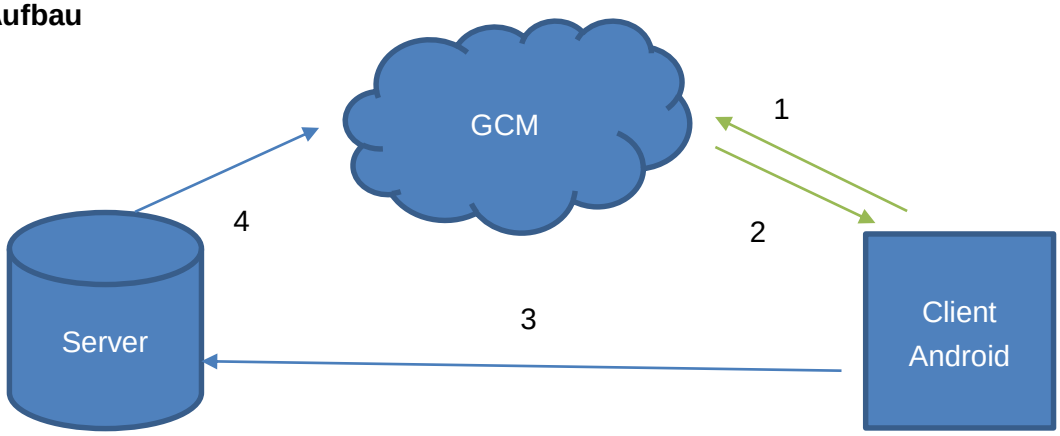

Abbildung 19: GCM Kommunikation

1. Android Clients senden Anwendungs- und Gerätinformationen an den GCM Server, um sich zu registrieren.
2. Dieser Server nimmt diesen entgegen und sendet dem Gerät eine eindeutige Identifikationsnummer zurück.
3. Diese ID wird dann an unseren Server gesendet, welcher diese in seine Datenbank speichert.
4. Falls der Server mit dem Android Gerät kommunizieren möchte, sendet der Server eine Anfrage an das GCM mit der Geräteidentifikationsnummer. Über das GCM können nun Daten an das Gerät gesendet werden.

2.1.10.3 Google Maps Android API

Die Google Maps API ermöglicht das Verwenden der Google Map Funktionen auf Android Clients. Diese API ist kostenlos und ist ebenfalls nicht durch Kontingentbeschränkungen limitiert.

2.1.10.4 Google Places API

Dies ist ein Dienst, der Informationen über bestimmte Bereiche auf der Google Map zurückgibt. Ein wichtiger Bestandteil dieses Dienstes ist, aus Koordinaten Adressen herauszufinden.

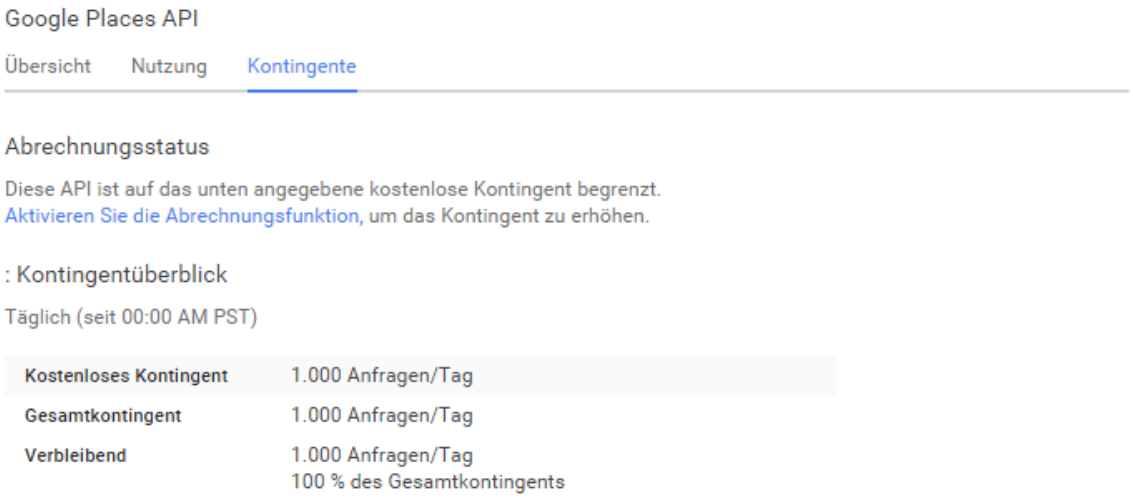

Abbildung 20: Kontingent

2.2 Entwicklungssysteme

2.2.1 Eclipse Luna

Eclipse ist eine quelloffene Entwicklungsumgebung zur Entwicklung verschiedenster Software. Das Eclipse Projekt wurde 2001 von IBM gegründet und später von der Eclipse Foundation, welche 2004 als non-profit Organisation zur Verwaltung des Projektes gegründet wurde, übernommen.

Hauptaugenmerk wurde auf die Java Entwicklung gelegt. Es werden aber auch andere Sprachen wie zB. C/C++ oder PHP unterstützt. Weiters bietet Eclipse die Möglichkeit zahlreiche Erweiterungen hinzuzufügen. Damit kann zum Beispiel Android unterstützt oder ein GUI Builder hinzugefügt werden.

Eclipse Luna bietet volle Unterstützung von Java 8 sowie für Plug-In Entwicklung oder Maven Integration. Des Weiteren bietet der Eclipse Compiler Sprachverbesserungen und Migration anonymer Klassen zu Lambda-Ausdrücken und neue Lambda-Formatoptionen (vgl. [15]).

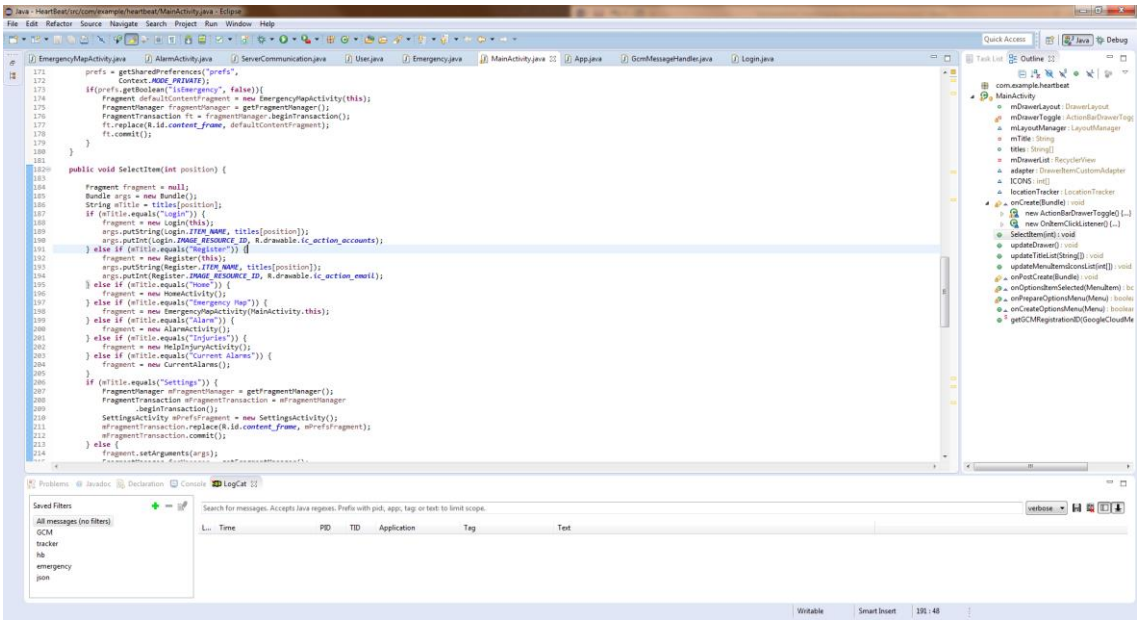

Abbildung 21: Eclipse Luna

2.2.2 Visual Studio 2013

Visual Studio ist eine Entwicklungsumgebung von Microsoft, welche auf .NET Sprachen spezialisiert ist. Es werden verschiedenste Tools angeboten, darunter Designer, Editoren, Debugger oder Profiler. Des Weiteren können Programme in einer Vielzahl verschiedener Sprachen entwickelt werden. Die gängigsten sind C#, Visual Basic, JavaScript oder HTML5. Außerdem wird umfassende Webunterstützung geboten. Dadurch können ASP.NET oder andere Seiten einfach und schnell entwickelt werden. Mithilfe des Microsoft Internet Information Servers können diese Webseiten danach einfach veröffentlicht werden.

Neben dem oben genannten Features werden noch zahlreiche andere Entwicklungswerkzeuge geboten, darunter umfassende Plattformunterstützung, welche es erlaubt, sowohl mobile Anwendungen als auch Webanwendungen, Cloud Services oder herkömmliche Windows Anwendungen zu entwickeln.

Ebenfalls werden Tools zur Entwicklung mit agilen Softwareentwicklungsmethoden geboten, welche unter anderem Burndown Charts oder Backlogs zur Verfügung stellen.

Weitere Features des Visual Studio 2013 sind Debugging und Diagnose-Tools, Tools für Software Tests oder Pläne zur Veröffentlichung der entwickelten Software (vgl. [16]).

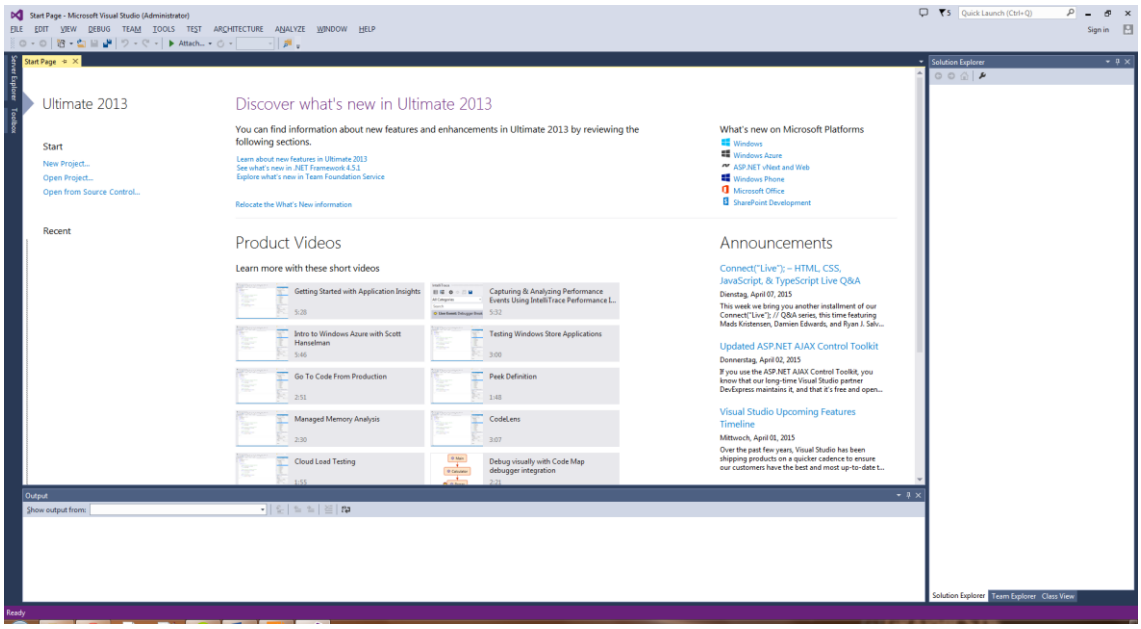

Abbildung 22: Visual Studio 2013

2.2.3 Microsoft Windows Server 2012 R2

Microsoft startete seine Serverlaufbahn mit Windows NT 3.1 Advanced Server im Jahre 1993. Damals hatte ein Server hauptsächlich die Aufgabe, einen Speicher für Daten bereitzustellen oder gemeinsames Drucken zu ermöglichen.

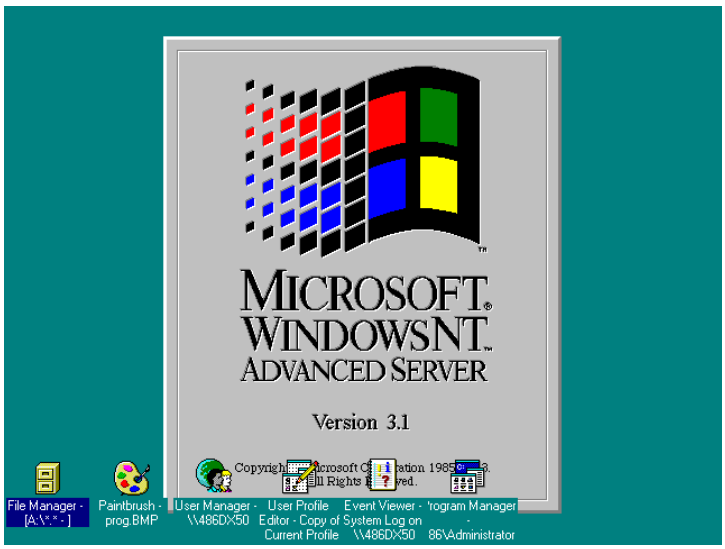

Abbildung 23: Windows NT Server 3.5 (aus [17])

1994 erschien der Windows NT Server 3.5 mit Mail Server und SQL Server. Der erste Durchbruch gelang mit Windows 2000 Server, welcher unter anderem einen neuen Internet Information Server mitbrachte.

Nach dem Windows Server 2003 folgte der Windows Server 2008, welcher sich in vielen Unternehmen durchgesetzt und als stabile Plattform bewährt hatte. Einige wichtige Neuerungen dieses Betriebssystems waren ein modularer Aufbau, wodurch nur die Features installiert wurden, die auch benötigt wurden, ein neuer Webserver (IIS) oder eine neue Servervirtualisierungslösung (Hyper-V).

Im Jahre 2012 wurde nun der Windows Server 2012 veröffentlicht. Dieser bot mit der Windows 8 Oberfläche ein komplett neues Look-and-Feel. Neben der Oberfläche gab es Neuerungen in nahezu allen Bereichen, darunter Active-Directory, DHCP oder DNS.

Im Windows Server 2012 R2 wurden dann wiederum zahlreiche Verbesserungen eingebracht, darunter Verbesserungen im Hyper-V oder Remote Access Bereich.

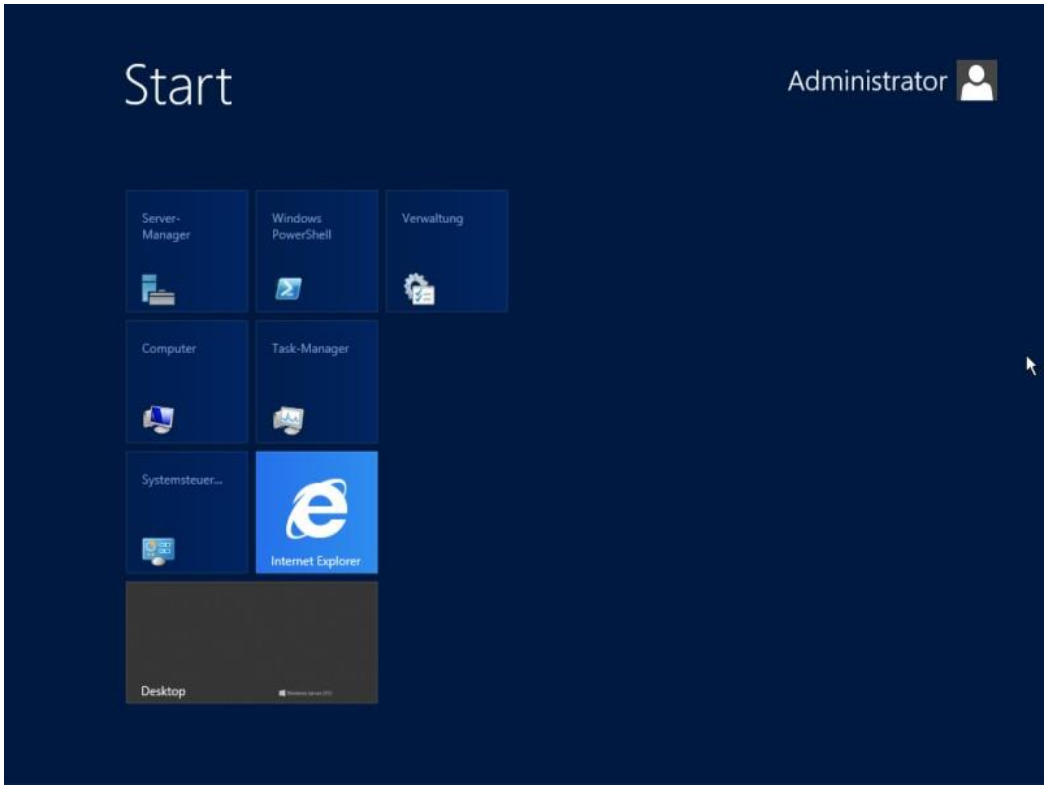

Abbildung 24: Windows Server 2012 (aus [17])

Im Windows Server 2012 gibt es verschiedene Rollen, welche je nach Bedarf installiert und durch Features ergänzt werden können. Die installierten Rollen können nun von bis zu 52 Features ergänzt werden. Wird zum Beispiel ein Fileserver erstellt, welcher in einem Cluster laufen soll, wird die Rolle „Dateidienste“ zuzüglich des Features „Failover-Clusterunterstützung“ installiert (vgl. [18]).

2.2.4 Microsoft SQL Server 2014

Der Microsoft SQL Server 2014 unterstützt Kunden beim Entwickeln von leistungsstarken, unternehmerischen Anwendungen im Big Data-, Data Warehousing-Bereich oder anderen unternehmenskritischen Bereichen.

Er bietet unter anderem gute In-Memory-Leistung, hohe Verfügbarkeit und Notfallwiederherstellung. Man verfügt über Replikationen, welche den Onlinebetrieb sichern und über Unterstützung beim Einrichten von Cloud Sicherungen.

Weiters bietet das Microsoft Produkt hohe Skalierbarkeit. Der Einsatz von bis zu 640 logischen Prozessoren ist dadurch möglich. Softwaretechnisch lässt sich der SQL Server in Visual Studio integrieren, was eine einfache Anwendungserstellung auf verschiedensten Plattformen erlaubt (vgl. [19]).

2.2.5 Microsoft SQL Server Management Studio

Das SQL Server Management Studio ist eine Umgebung für den Zugriff auf den SQL Server. Es werden Möglichkeiten zur Konfiguration, Verwaltung und Entwicklung aller SQL Server Komponenten geboten. Es werden eine Vielzahl grafischer Tools und Skript-Editoren für Entwickler und Administratoren geboten.

Im SQL Server Management Studio werden einzelne Tools wie der Query Analyzer oder der Analysis-Manager in einer Umgebung vereint.

Einige wichtige Funktionen des SSMS sind Verwaltungsaufgaben für den SQL Server, Dialogfenster zum Verwalten von Datenbankobjekten oder Aktivitätsmonitore zur Überwachung der Performance des SQL Servers (vgl. [19]).

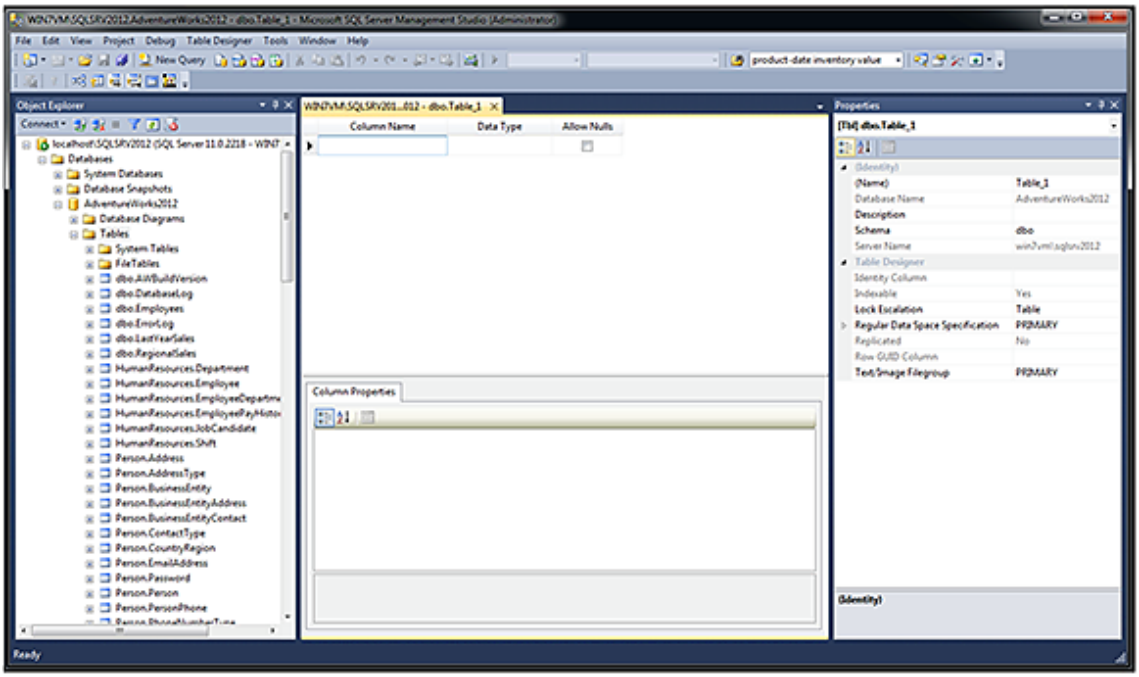

Abbildung 25: SQL Server Management Studio

2.2.6 Smart Git

Smart Git ist ein Git-Client, welcher eine graphische Oberfläche besitzt.

Bei Git handelt es sich um ein quelloffenes Versionsverwaltungstool. Git erlaubt es mehrere lokale Zweige zu besitzen, welche komplett unabhängig laufen und jederzeit zusammengefügt werden können. Im Vergleich mit anderen Versionsverwaltungen ist es kleiner und schneller. Im Gegensatz zu vergleichbarer Software wird statt einem „checkout“ des derzeit benötigten Codes das gesamte Repository geklont. Dies erhöht die Ausfallsicherheit, da jeder Nutzer ein Backup des gesamten Codes besitzt, welches jederzeit auf das Repository zurückgespielt werden kann. Außerdem besitzt Git einen Staging Bereich, der es erlaubt, einen Commit vor dem Abschluss zu bearbeiten. Dies erlaubt es auch nur einen Teil seiner modifizierten Daten zu committen (vgl. [20]).

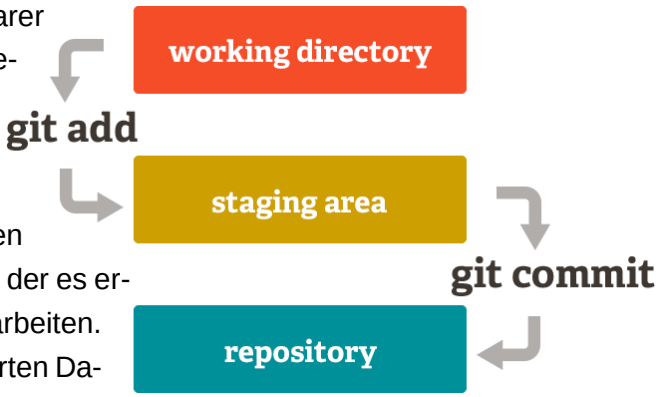

Abbildung 26: Git Staging Area

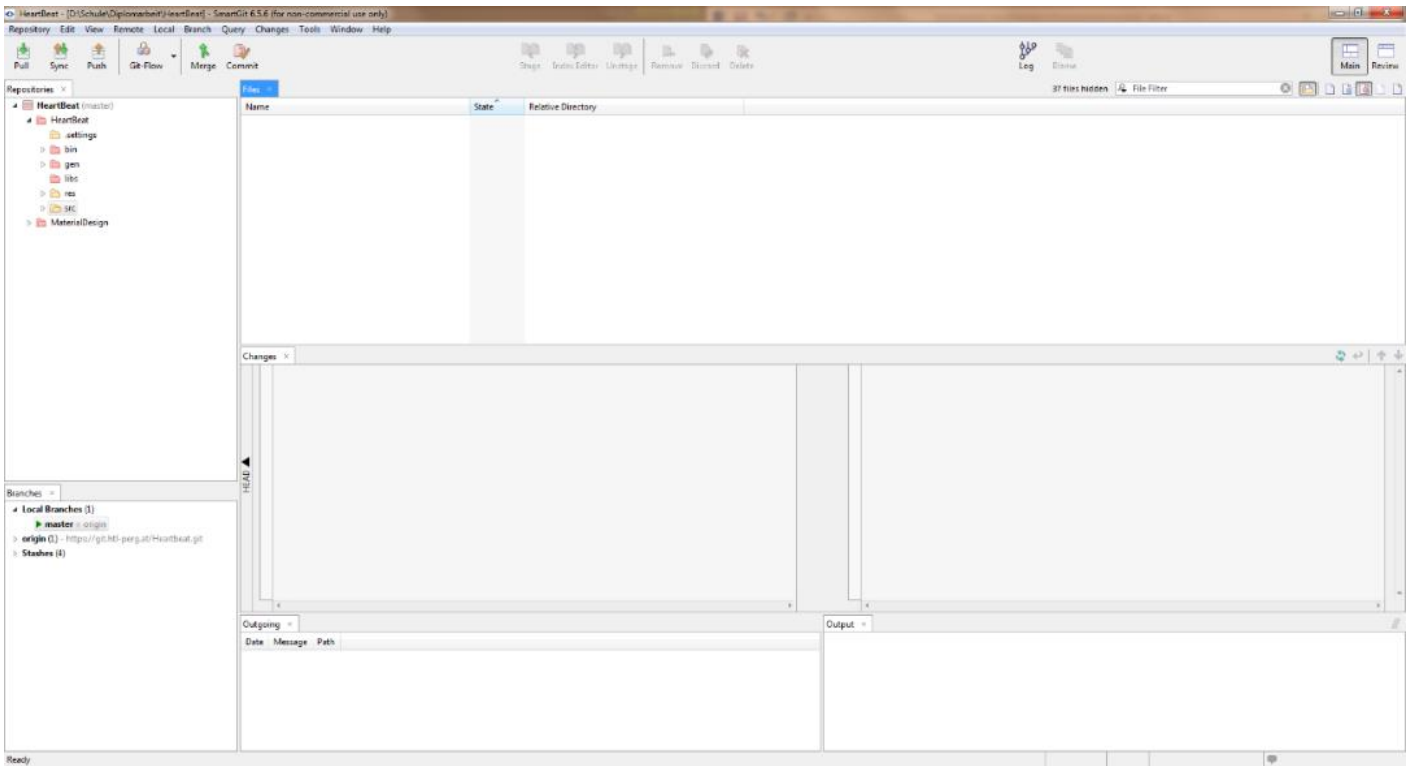

Abbildung 27: SmartGit

2.3 Development Kits

2.3.1 Android Software Development Kit (Android SDK)

Der Android SDK bietet die Grundlage zur Entwicklung von Android Anwendungen. Er enthält neben dem Android Source Code auch Entwicklertools, einen Android Emulator zum Testen der erstellten Anwendungen, auch Beispielprojekte, welche beim Erlernen neuer Technologien helfen können.

Im Sourcecode finden sich verschiedenste Klassen, auf denen Android Projekte basieren und die im eigenen Code eingebunden werden können.

Beispiele für diese Entwicklertools sind Logcat, das dem Entwickler erlaubt, auf Logeinträge am Android Gerät zuzugreifen, oder der SDK Manager, der es dem Entwickler erlaubt SDK Tools, Beispiele oder Dokumentationen zu verschiedenen API Levels herunterzuladen (vgl. [21]).

2.4 Bibliotheken

2.4.1 Android-support-v4/v7

Die Android Support Bibliotheken bieten zusätzlich zu den Standard Android APIs bestimmte Features für verschiedene Android-Versionen.

Die v4 Support-Bibliothek bietet den größten Umfang an zusätzlichen Features. Sie kann in Android Projekte ab API Level 4 (Android 1.6) verwendet werden. Die Bibliothek bietet unter anderem wichtige Klassen in vier verschiedenen Kategorien.

Unter der Kategorie „App Components“ findet man zum Beispiel Klassen, die Unterstützung für Benachrichtigungen bieten oder Anwendungen erlauben, Intents (siehe 6.2.4) innerhalb der Anwendung zu registrieren oder zu empfangen, ohne diese global zu senden.

In der zweiten Kategorie findet man Klassen, welche die Benutzeroberfläche betreffen. Diese unterstützen die Erstellung eines „Navigation Drawers“, der auch in unserer Applikation Verwendung findet, oder die Erstellung von Layouts, die verschiedene Oberflächen verwalten.

Ebenfalls findet man in der Bibliothek Klassen, welche die Zugreifbarkeit vereinfachen sollen. Ein Beispiel wäre ein Helfer zur Implementierung von Accessibility Support für benutzerdefinierte Sichten.

Die letzte der vier großen Kategorien befasst sich mit dem Inhalt. Hier findet man Klassen, die Unterstützung beim asynchronen Laden von Daten bieten oder das Teilen von privaten Dateien über Anwendungsgrenzen hinweg ermöglicht.

Die v7 Support-Bibliothek bietet Unterstützung des Action Bar Entwurfmusters und des Material Designs (vgl. [21]).

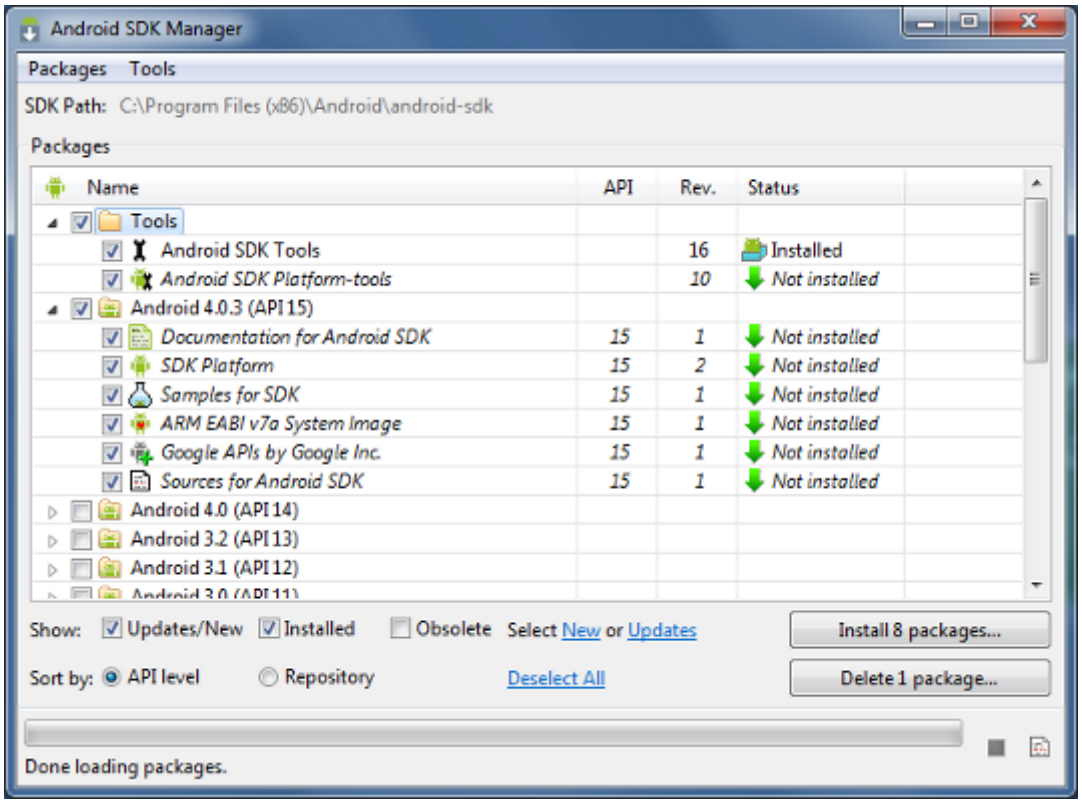

Abbildung 28: Android SDK Manager

2.4.2 Material Design

Die Material Design Bibliothek ist eine quelloffene Bibliothek, welche verschiedene Benutzeroberflächenelemente zur Verfügung stellt. Mithilfe dieser Bibliothek können Anwendungen im Material Design auch für Geräte verfügbar gemacht werden, welche kein Material Design unterstützen.

2.5 Multi-Tier-Architektur

2.5.1 Schichtenarchitektur

Die Schichtenarchitektur beschreibt eine Architektur der Softwareentwicklung, welche aus zwei bzw. drei Schichten besteht.

Die ersten Architekturen waren die Mainframe-Architektur, bei der Benutzer über Terminals auf die Anwendungslogik zugriffen, und die Filesharing-Architektur, bei der gemeinsame Dateien auf den Rechner übertragen und anschließend wieder zurückgeschrieben wurden.

Aus den Nachteilen der vorangegangenen Modelle entstand die Client/Server-Architektur. Hierbei wird der Fileserver durch eine relationale Datenbank ersetzt. Es wurden darauf hauptsächlich „Remote Procedure Calls“ oder SQL-Ausdrücke ausgeführt, um auf Daten zuzugreifen.

Daraus entwickelte sich wiederum die Zwei-Schichten-Architektur. Darunter versteht man die Aufteilung des Systems in einen Datenbankserver und mehrere Clients. Die Anwendungslogik ist zwischen dem Datenbanksystem (Prozeduren) und der Client-Anwendung aufgeteilt. Der Nachteil dieser Architektur ist die verstärkte Abhängigkeit von den Werkzeugen des Datenbankherstellers.

Bei der Drei-Schichten-Architektur wurde eine zusätzliche Schicht zwischen dem Client und dem Datenbankserver eingefügt. Diese Schicht steuert die Kommunikation mit dem Client und verarbeitet die Daten der Datenbank, bevor sie an den Client weitergeleitet werden. Starke Skalierbarkeit und hohe Flexibilität sind die größten Stärken dieses Modells.

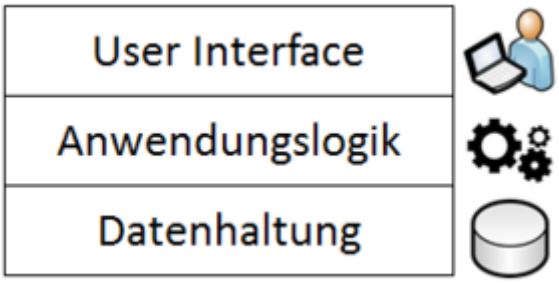

Abbildung 29: Drei-Schichten Modell (aus [22])

Die erste Schicht, die Datenhaltungsschicht, speichert und verwaltet die Daten der Anwendung. Heutzutage besteht diese Schicht hauptsächlich aus objektorientierten, relationalen oder anderen Datenbanksystemen. Zur Kommunikation mit der Anwendungsschicht werden meist standardisierte Schnittstellen wie SQL oder ODBC verwendet.

In der Anwendungsschicht werden die Daten aus der Datenhaltungsschicht angefordert und verarbeitet. Dies ermöglicht die Ausführung von verschiedenen Geschäftsprozessen unabhängig von den Tools der Datenbankhersteller.

Die Präsentationsschicht steuert die Anzeige der Daten der Anwendungsschicht und interpretiert Benutzereingaben. Durch diese Eingaben werden auf der Anwendungsschicht Prozesse ausgelöst, welche wiederum für Veränderungen an der Datenhaltungsschicht sorgen (vgl. [23]).

2.5.2 Konkreter Aufbau

In unserem System verwenden wir die Drei-Schichten-Architektur.

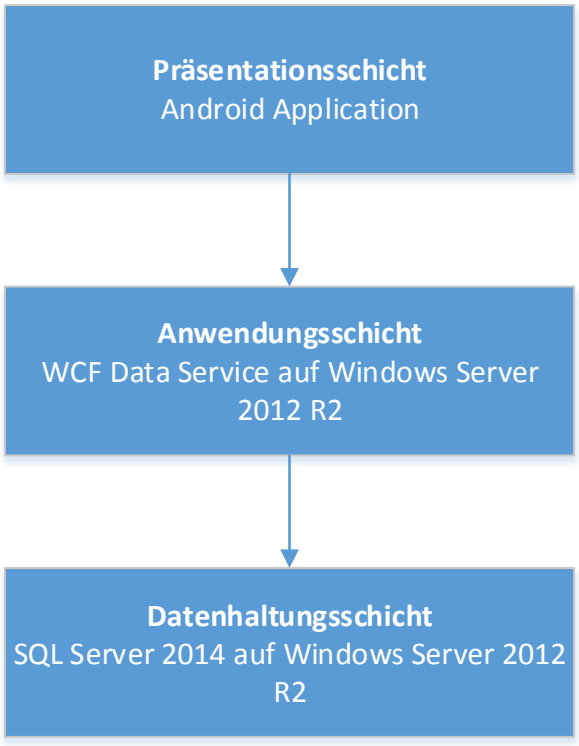

Abbildung 30: Aufbau unseres Systems

Wie in der vorangegangenen Grafik zu sehen, verwenden wir für die Präsentationsschicht eine Android-Applikation, welche die gesamte Oberfläche für unser System bietet. Hier werden sowohl die Notfälle angezeigt, als auch alle anderen Features grafisch dargestellt.

Als Anwendungsschicht benutzen wir einen WCF Data Service, welcher die Businesslogik beinhaltet. Hier werden Daten in die Datenbank geschrieben, gelesen und verarbeitet. Ein Beispiel für diese Logik ist das Auslösen eines Notfalls. Dabei werden Daten in die Datenbank gespeichert und weitere Clients benachrichtigt.

In der Datenhaltungsschicht kommt ein SQL Server zur Anwendung. Darin werden alle Datenbanken und deren enthaltene Daten gespeichert und verwaltet. Er reagiert auf Anfragen aus der Anwendungsschicht und speichert Daten, welche durch diese gesendet werden.

3 Systemanalyse

Diese Kapitel beschäftigen sich mit der Analyse des Systems und dem dazugehörigen Datenmodell.

3.1 Notfallsoftware

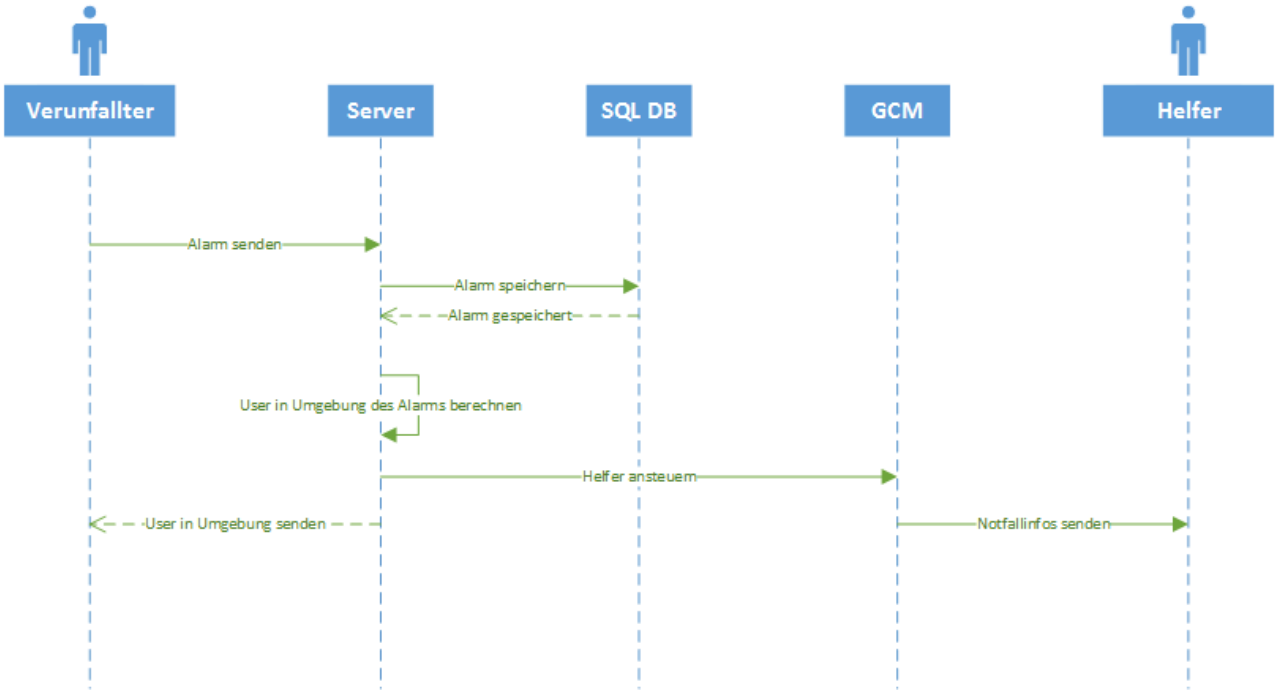

Abbildung 31: Sequenz Diagramm des Ablaufes

3.2 Datenmodell

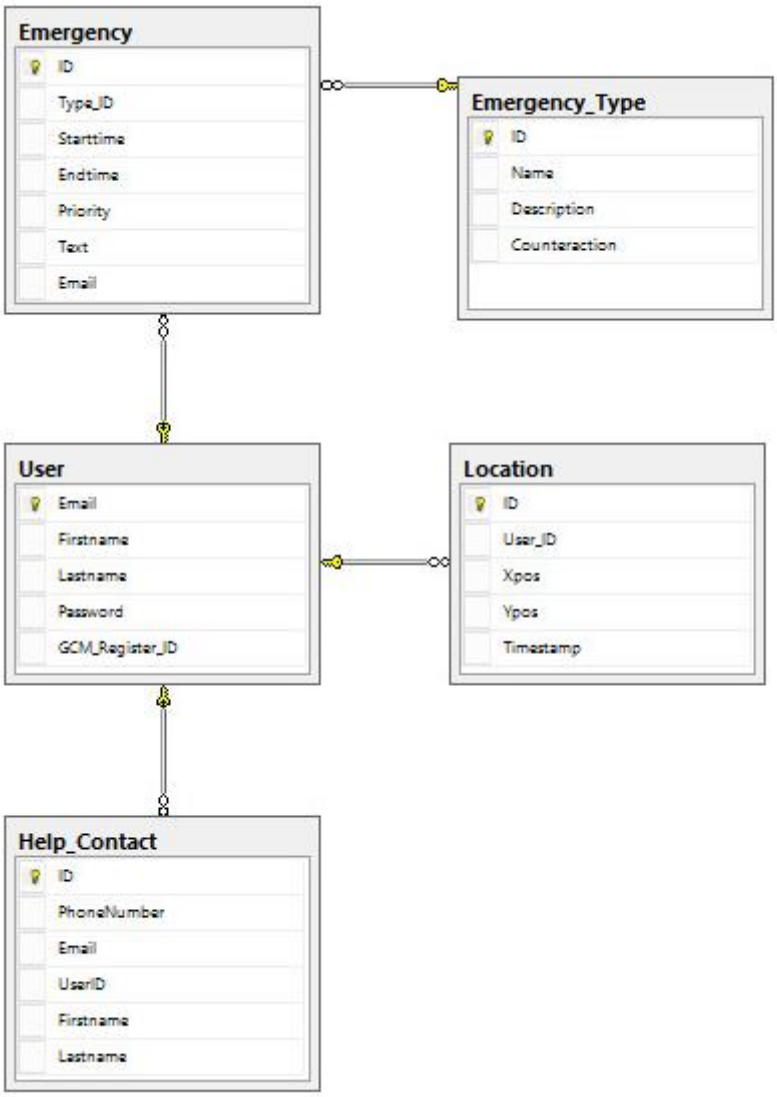

Abbildung 32: Datenmodell HeartBeat

3.2.1 User

Die Tabelle „User“ speichert jeden Nutzer unserer Applikation. Es werden persönliche Informationen wie Vorname, Nachname, Email und Passwort gespeichert. In jedem User wird die GCM ID mitgespeichert, um mit der Client Applikation vom Server aus zu kommunizieren (siehe 2.1.10.2).

	Email	Firstname	Lastname	Password	GCM_Register_ID
4	d.o@test.com	David	Test	a152e841783914146e4bcd4f39100686	APA91bH8Es9-7jWF

Abbildung 33: Datensatz User

3.2.2 Location

Jeder User hat 0 bis n Positionen. Diese Positionen werden mit einem Zeitstempel in der Tabelle „Location“ gespeichert. Im Falle eines Notfalles kann somit die Position des Verunfallten genau nachverfolgt und analysiert werden.

	ID	User_ID	Xpos	Ypos	Timestamp
687	7...	test.user@hb.com	48,418872833252	14,6128530502319	1425673927

Abbildung 34: Datensatz Location

3.2.3 Help_Contact

Jeder User hat bestimmte soziale Kontakte, die er benachrichtigen kann, wenn es zu einem Unfall kommt. Dies geschieht automatisch mit den in der „Help_Contact“ gespeicherten Telefonkontakten. Bei einem Unfall werden aus dieser Tabelle die von dem User angelegten Kontakte herausgenommen und kontaktiert.

	ID	PhoneNumber	Email	UserID	Firstname	Lastname
1	2	06503432423	test.user@hb.com	test.user@hb.com	Michael	Mayr

Abbildung 35: Datensatz Help_Contact

3.2.4 Emergency

Der eigentliche Unfall eines Nutzers wird in dieser Tabelle gespeichert. Ein Unfall hat eine Startzeit und eine Endzeit. Die Endzeit ist höchstens 30 Minuten später als die Startzeit, damit den Helfern keine inkonsistenten Daten angezeigt werden. Die Tabelle „Emergency“ enthält ebenfalls optionale Parameter wie die Priorität und einen Zusatztext. Da wir uns als Entwickler durchaus bewusst sind, dass verunfallte Personen nicht immer in der Lage sind, diese optionalen Parameter auszufüllen, kann ein Notruf auch ohne diese abgesendet werden.

	ID	Type_ID	Starttime	Endtime	Priority	Text	Email
14	13	1	42065	1430289478	6	hallo	test.user@hb.com

Abbildung 36: Datensatz Emergency

3.2.5 Emergency_Type

Um einen Notruf und deren Verletzung besser zuordnen zu können, gibt es die Tabelle „Emergency_Type“. Hier werden verschiedene Verletzungen mit deren Gegenmaßnahmen gespeichert, um den User bestmögliche Hilfestellung zu bieten.

	ID	Name	Description	Counteraction
1	0	Sonstige Verletzung	NULL	NULL

Abbildung 37: Datensatz Emergency_Type

4 Produktdokumentation

4.1 Ansichten der Android Applikation

4.1.1 Login

In der Ansicht Login findet man Felder zum Eingeben der E-Mail Adresse und des Passwortes und ein Button um sich einzuloggen. Außerdem befindet sich ein Button zum Teilen eines Links zur Applikation und ein Button zur Ansicht der Registrierung (siehe Abbildung 38).

4.1.2 Menü

Im Menü, welches man mittels Linkswisch öffnen kann, findet man oben eine Übersicht zu seinem Account. Weiter unten befinden sich die einzelnen Menüpunkte (siehe Abbildung 39).

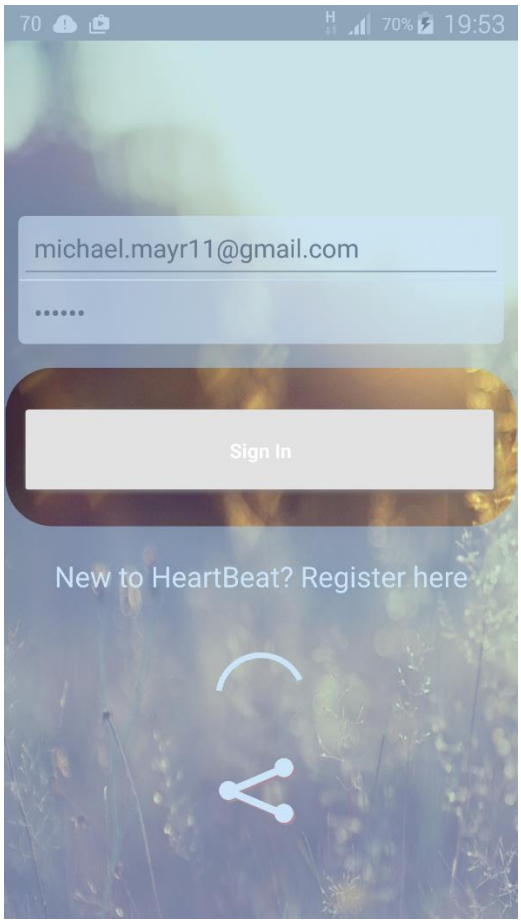

Abbildung 38: Login

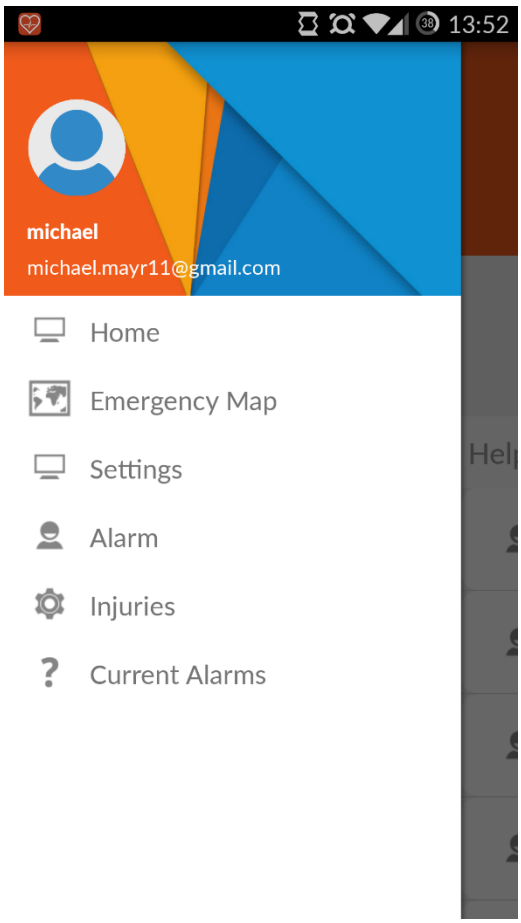

Abbildung 39: Menü

4.1.3 Notfallkontakte

Hier findet man eine Übersicht über alle bereits hinzugefügten Notfallkontakte. Man kann diese mittels des Kreuzes neben dem Kontakt löschen oder auch mittels des Float-Button am rechten unteren Rand Kontakte aus der Kontaktliste hinzufügen (siehe Abbildung 40).

4.1.4 Notfall auslösen

In dieser Ansicht findet sich lediglich ein Button zum Auslösen eines Notfalles (siehe 4.2.3)(siehe Abbildung 41).

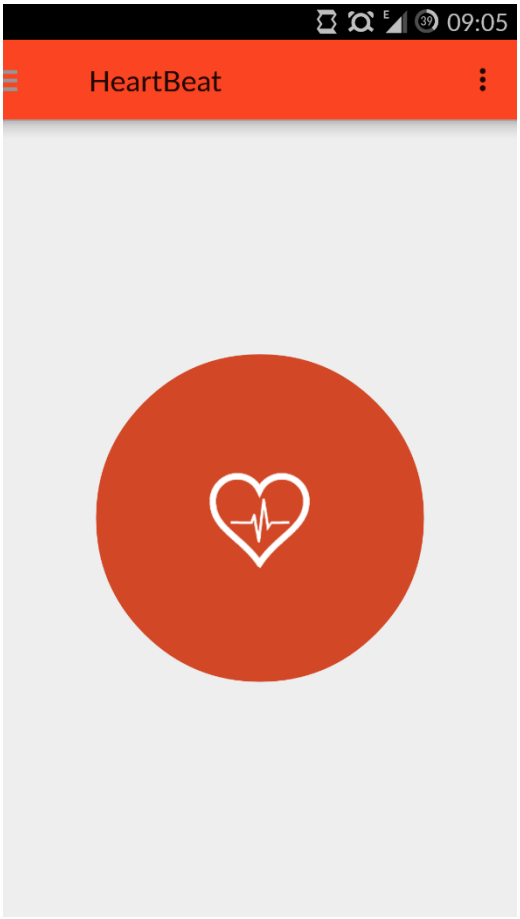

Abbildung 41: Alarm auslösen

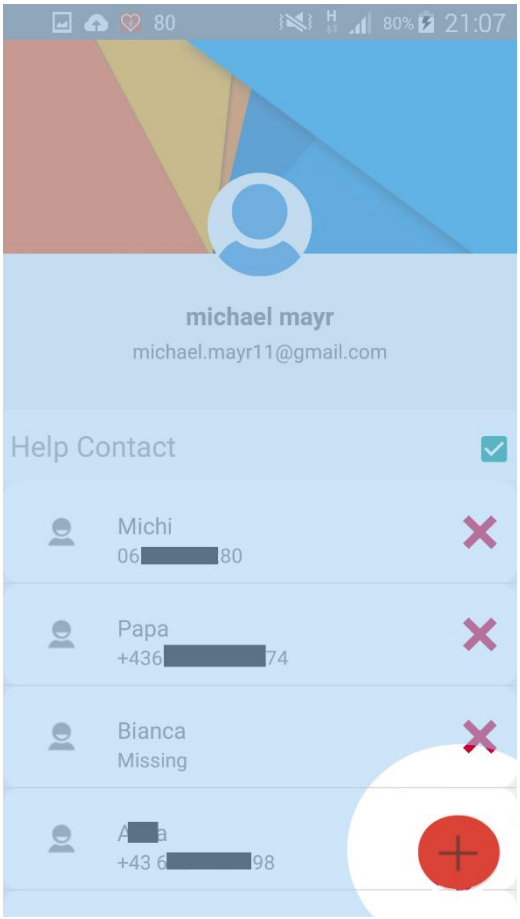

Abbildung 40: Notfallkontakte

4.1.5 Notfallkarte

Auf der Karte werden sowohl Notfälle angezeigt, bei denen man benachrichtigt wurde, als auch naheliegende Gesundheitszentren, sofern diese mittels Dropdown und Find Button oben links angefordert wurden (siehe Abbildung 43).

4.1.6 Übersicht Notfälle

In dieser Ansicht sieht man alle derzeit aktuellen Notfälle. Diese werden in einer Liste angezeigt. Oben sieht man die Anzahl der Notfälle in der Liste. Pro Notfall wird die E-Mail Adresse des Verunglückten angezeigt und dessen Koordinaten. Mittels des Plus rechts unten kann man direkt zum Notfallbutton navigieren (siehe Abbildung 42).

Abbildung 43: Notfallkarte

Abbildung 42: Notfallübersicht

4.1.7 Erste-Hilfe-Maßnahmen

Im Menüpunkt „Injuries“ findet man eine Übersicht über wichtige Erste-Hilfe Maßnahmen. Klickt man auf einen Punkt der Liste, wird man auf eine detailliertere Ansicht weitergeleitet (siehe Abbildung 44).

Abbildung 44: Erste Hilfe

4.2 Programmablauf – Android Applikation

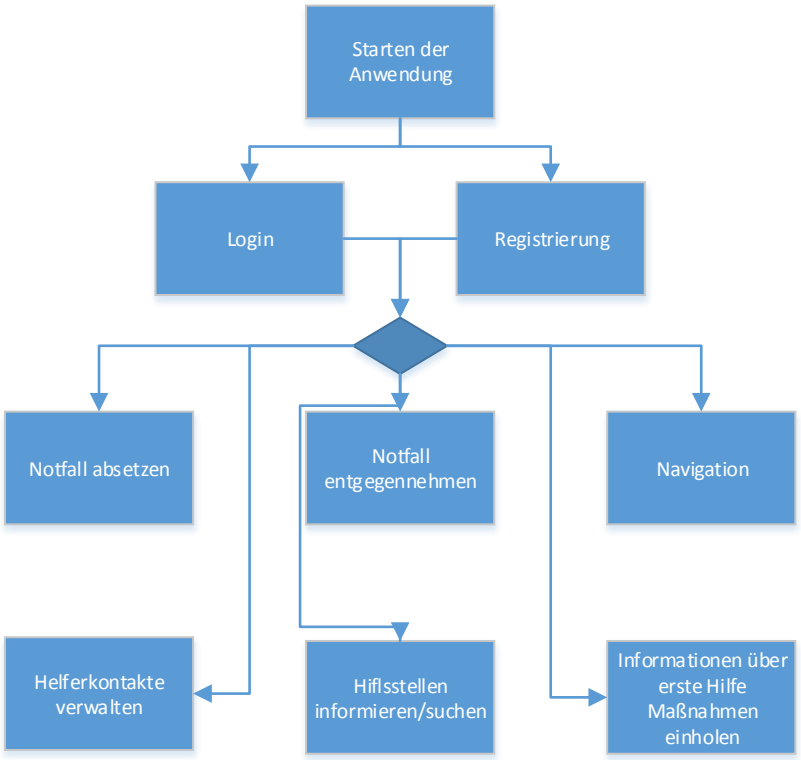

Abbildung 45: Programmablauf

4.2.1 Login

Nachdem der Benutzer die App gestartet hat, findet er vor sich die Login-Seite. Auf dieser kann er seinen Nutzernamen und sein Passwort eingeben, sich damit einloggen, oder mittels Registrierungslink zur Registrierungsseite navigieren.

Will sich der Benutzer einloggen und drückt den Login-Button, wird zuerst geprüft, ob die E-Mail Adresse und das Passwort dem richtigen Format entsprechen. Danach wird eine asynchrone Aufgabe gestartet, welche die Zugangsdaten mit dem Server abgleicht. Dazu wird mittels Anonymous Authentication eine Methode aufgerufen, die Daten vom SQL Server abruft und mit den von der Applikation gesendeten Daten vergleicht.

In der asynchronen Aufgabe wird zuerst eine URL erstellt, mit welcher eine Login Methode am Server aufgerufen wird. War der Login erfolgreich, bekommt man das entsprechende Benutzer Objekt zurück, ansonsten wird eine Fehlermeldung ausgegeben.

Anschließend wird auch noch die GCM Registration ID mit der am Server gespeicherten abgeglichen und gegebenenfalls aktualisiert.

War der Login erfolgreich, werden die E-Mail und der Vor- und Nachname als global zugreifbare Variablen gespeichert. Danach wird das Menü aktualisiert und der Nutzer auf die Startseite weitergeleitet. Außerdem werden die Zugangsdaten lokal in einer Datei gespeichert, um den nächste Login zu erleichtern.

Zu guter Letzt wird noch das Service gestartet, welches die Position des Nutzers verfolgt.

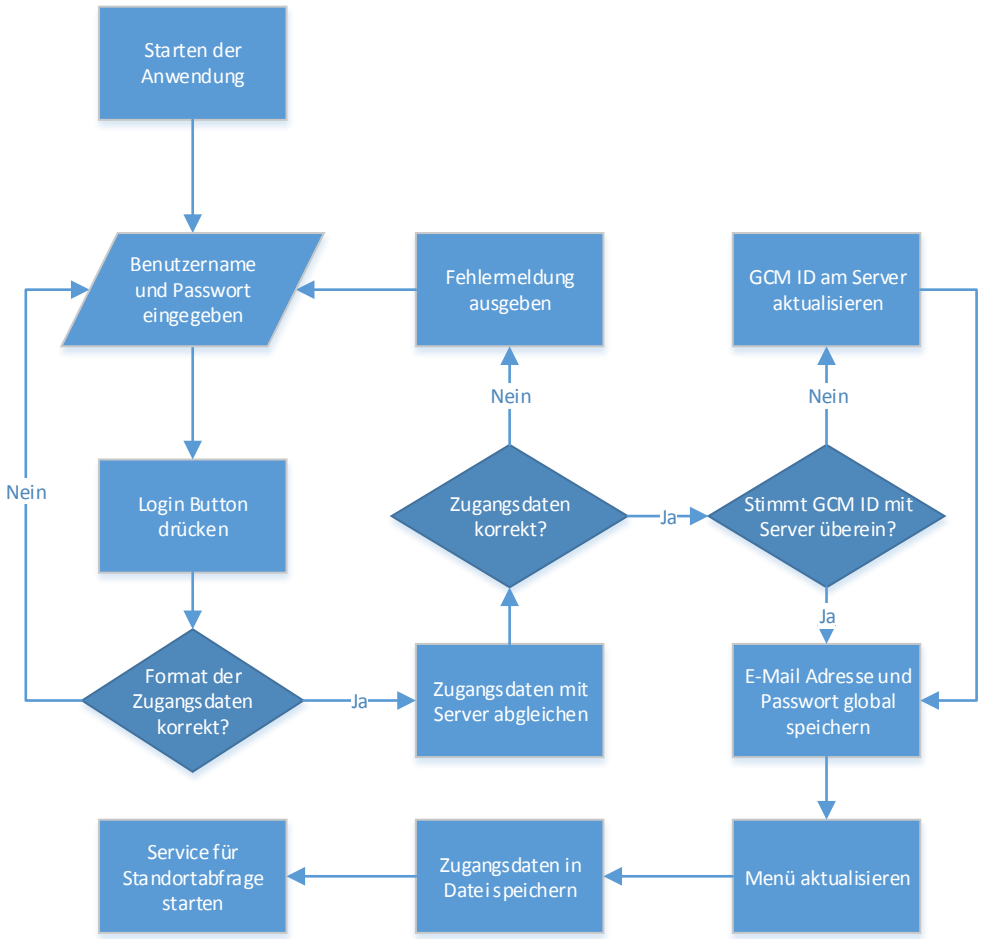

Abbildung 46: Login

4.2.2 Registrierung

Nachdem der Nutzer den Button „Registrieren“ auf der Login Seite gedrückt hat, wird er auf die Registrierungsseite weitergeleitet. Hier findet er Felder zum Eingeben seiner E-Mail Adresse, seines Vor- und Nachnamens sowie seines gewünschten Passwortes.

Drückt er nun den Registrierungsknopf, wird zuerst geprüft, ob alle Eingaben den Anforderungen entsprechen. Danach wird eine asynchrone Aufgabe gestartet, welche das Anlegen des Users am Server managt.

Dazu wird zuerst die GCM Registration ID ausgelesen und eine URL zusammengestellt, welche wiederum eine Methode am Server aufruft. War das Anlegen des Users erfolgreich, wird das angelegte User Objekt vom Server zurückgegeben, ansonsten wird eine Fehlermeldung ausgegeben.

Danach wird der Nutzer wieder auf die Login-Seite umgeleitet.

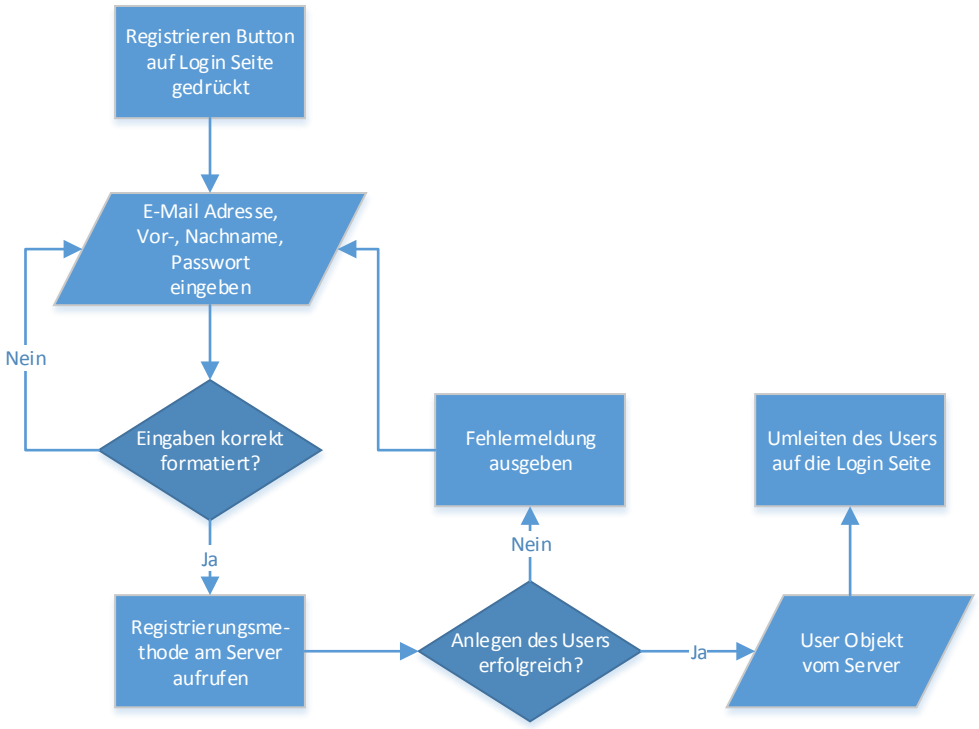

Abbildung 47: Registrierung

4.2.3 Notfall absetzen

Der Nutzer kann einen Notruf absetzen, sobald er eingeloggt ist. Dazu wählt er im Menü den Punkt „Alarm“ und drückt den pulsierenden Knopf mit dem HeartBeat Logo.

Danach wird automatisch eine asynchrone Aufgabe gestartet, welche den Notfall am Server anlegt.

In der asynchronen Aufgabe wird wieder ein GET-Request erstellt, mit dem eine Methode am Server aufgerufen wird. Danach wird der Nutzer auf die Emergency Map Seite umgeleitet.

Am Server wird zuerst ein Notfall angelegt und in der Datenbank gespeichert. Danach werden alle Nutzer im Umkreis von zwei Dezimalgraden gesucht und per Google Cloud Messaging benachrichtigt.

In der GCM Methode wird zuerst ein JSON String angelegt, welcher dann als POST Parameter übergeben wird. Danach wird dieser mittels HTTP Request an Google gesendet, wo automatisch jedes darin enthaltene Smartphone benachrichtigt wird.

Am Smartphone wird unterdessen dem Nutzer eine Karte mit allen potenziellen Helfern angezeigt. Dazu wird wieder eine asynchrone Aufgabe verwendet, in der eine Methode am Server aufgerufen wird, welche alle vorher benachrichtigten Nutzer zurückgibt. Diese werden dann mittels Marker auf der Karte angezeigt.

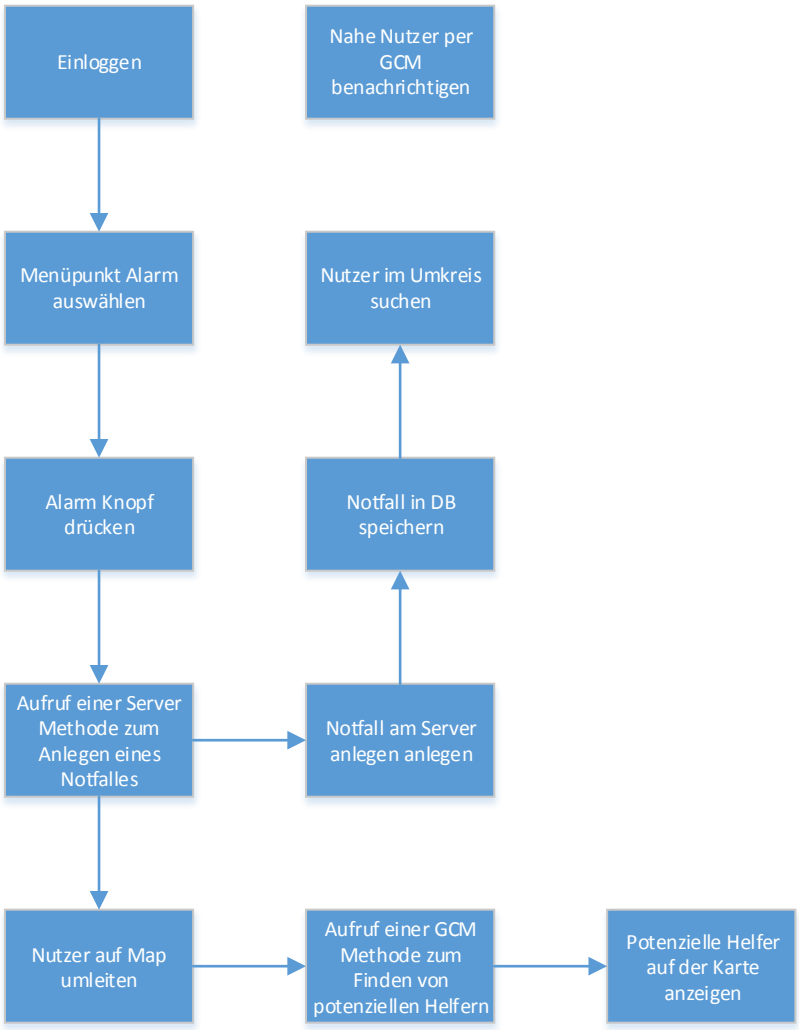

Abbildung 48: Notfall absetzen

4.2.4 Notfall entgegennehmen

Sobald der Nutzer eine Benachrichtigung via Google Cloud Messaging bekommt, wird diese mithilfe der Android Benachrichtigungen angezeigt. Klickt er diese an, öffnet sich die Anwendung und er wird automatisch mit dem zuletzt verwendeten User eingeloggt. Optional kann er die Anwendung auch manuell öffnen. Danach wird eine Karte mit dem Standort des Verletzten angezeigt. Dazu wird ein neuer Thread erstellt, welche für die Dauer des Notfalles die Standortdaten des Notfallusers lädt. Diese Informationen werden alle 30 Sekunden aktualisiert.

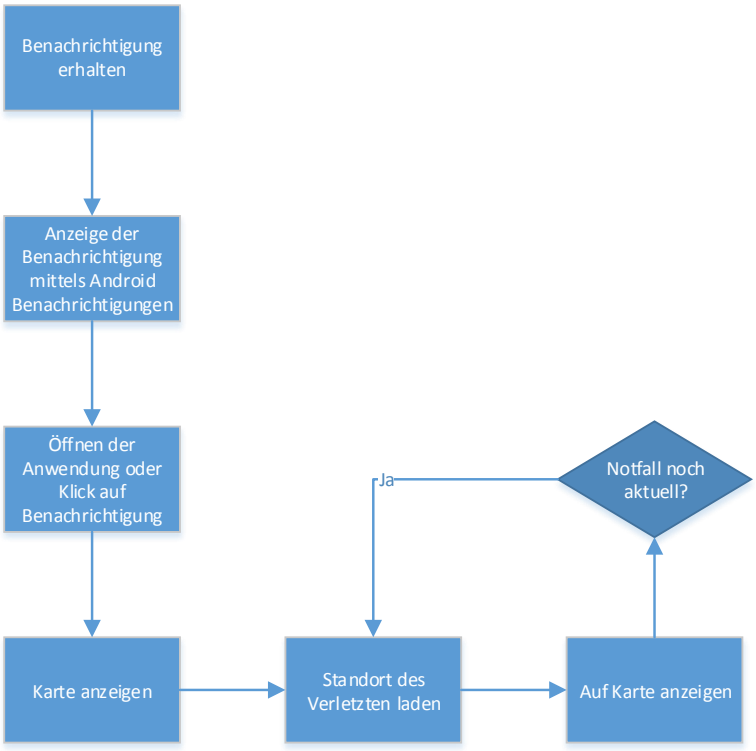

Abbildung 49: Notfall bekommen

4.2.5 Hilfsstelleninformationen

Der Nutzer kann sich in der Nähe befindliche Hilfezentren wie Ärzte oder Krankenhäuser anzeigen lassen. Dazu wählt er am oberen Kartenrand der Emergency Map aus, was er finden will und drückt danach auf den Find Button. Damit werden automatisch Daten vom Google Server geladen und auf der Karte angezeigt.

Dies funktioniert wieder mit einer asynchronen Klasse, wo JSON Daten zuerst vom Server geladen werden und anschließend in eine Liste von Plätzen umgewandelt werden. Diese Liste wird anschließend durchlaufen und jedes Element auf der Karte mittels Marker angezeigt.

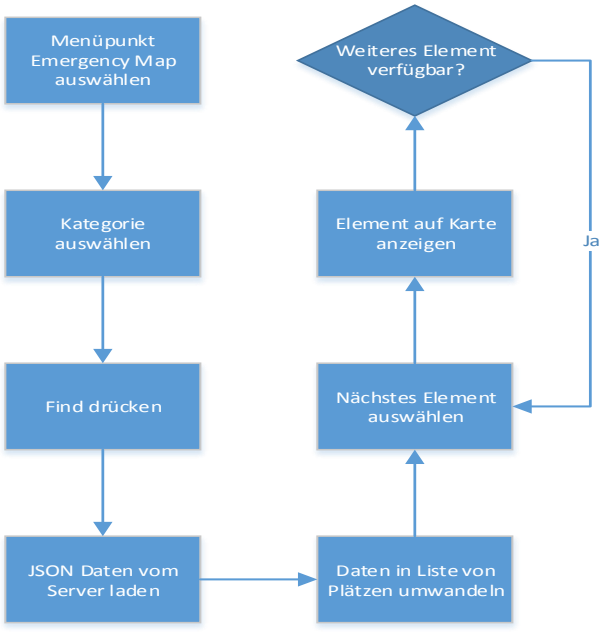

Abbildung 50: Hilfsstelleninfos

5 Konfiguration der Entwicklungsumgebung

Dieses Kapitel behandelt die benötigten Entwicklungsumgebungen und deren Konfiguration. Hauptaugenmerk wurde hierbei auf Eclipse und Visual Studio gelegt.

5.1 Android

Für die Android Programmierung wurde die Entwicklungsumgebung „Eclipse“ verwendet. Für nähere Informationen über diese Umgebung siehe 2.2.1.

Eclipse benötigt einige Tools und Plugins, um damit in der Sprache Android programmieren zu können.

Java JDK

Der Java Development Kit kann auf der Oracle Website heruntergeladen werden (siehe Abbildung 51). Nach dem Herunterladen muss nur die *.exe ausgeführt und dem Wizard gefolgt werden.

Java SE Development Kit 8u45

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	146.89 MB	jdk-8u45-linux-i586.rpm
Linux x86	166.88 MB	jdk-8u45-linux-i586.tar.gz
Linux x64	145.19 MB	jdk-8u45-linux-x64.rpm
Linux x64	165.24 MB	jdk-8u45-linux-x64.tar.gz
Mac OS X x64	221.98 MB	jdk-8u45-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	131.73 MB	jdk-8u45-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	92.9 MB	jdk-8u45-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	139.51 MB	jdk-8u45-solaris-x64.tar.Z
Solaris x64	95.88 MB	jdk-8u45-solaris-x64.tar.gz
Windows x86	175.97 MB	jdk-8u45-windows-i586.exe
Windows x64	180.42 MB	jdk-8u45-windows-x64.exe

Abbildung 51: JDK

Eclipse IDE

Die Entwicklungsumgebung Eclipse kann auf [Link 1] heruntergeladen werden. Nachdem der Download abgeschlossen ist, befindet sich eine *.zip Datei im Downloadordner. Dieser kann mit verschiedenen Programmen extrahiert werden (z.B. 7Zip, WinZip, usw.). Eclipse muss nicht installiert werden, darum ist das Programm direkt nach dem Extrahieren voll funktionsfähig.

Android SDK

Der Android SDK kann auf der Website [Link 2] heruntergeladen werden. Dieser SDK bringt wiederum einen Wizard mit sich, der Speicherposition und Shortcutbezeichnungen als Eingabe verlangt. Im Hintergrund des Wizard wird der Android SDK Manager

mitinstalliert. Hier können SDKs nachinstalliert und verschiedene Android-Versionen hinzugefügt werden.

5.2 .NET

5.2.1 Visual Studio

Für die serverseitige Programmierung wurde hauptsächlich die Entwicklungsumgebung Visual Studio verwendet (siehe 2.2.2).

Für die Entwicklung wurde die Version Visual Studio 2013 Update 3 verwendet. Diese Entwicklungsumgebung kann von der offiziellen Website [Link 3] heruntergeladen werden. Hier wird ein Installer heruntergeladen, welcher durch optionale Eingaben das Programm installiert. Gleich nachdem der Installer fertig ist, ist das Programm ausführbar und es muss nur noch mit dem Microsoft Account verbunden werden, um das Produkt zu aktivieren. Das Projekt HeartBeat ist im Hintergrund mit einer SQL Datenbank verbunden, um verschiedene Benutzerdaten zu speichern. Diese Datenbank ist aber noch nicht in diesem Paket enthalten und muss zusätzlich nachinstalliert werden (siehe 5.2.2).

5.2.2 SQL Server

Um Daten zu speichern und zu verwalten, bietet sich für die .NET Umgebung der SQL Server 2014 an. Dieser ist unter [Link 4] zu erreichen. Um mehr Infos über die Architektur und die Funktionsweise dieses Servers zu erfahren, verweisen wir auf das Kapitel 2.1.2. Der SQL Server wird mithilfe eines SQL Server Installation Centers installiert und konfiguriert. Da noch keine Serverinstanz auf dem Rechner vorhanden ist, wird der Punkt „New SQL Server stand-alone installation or add features to an existing installation“ im Reiter „Installation“ ausgewählt (siehe Abbildung 52).

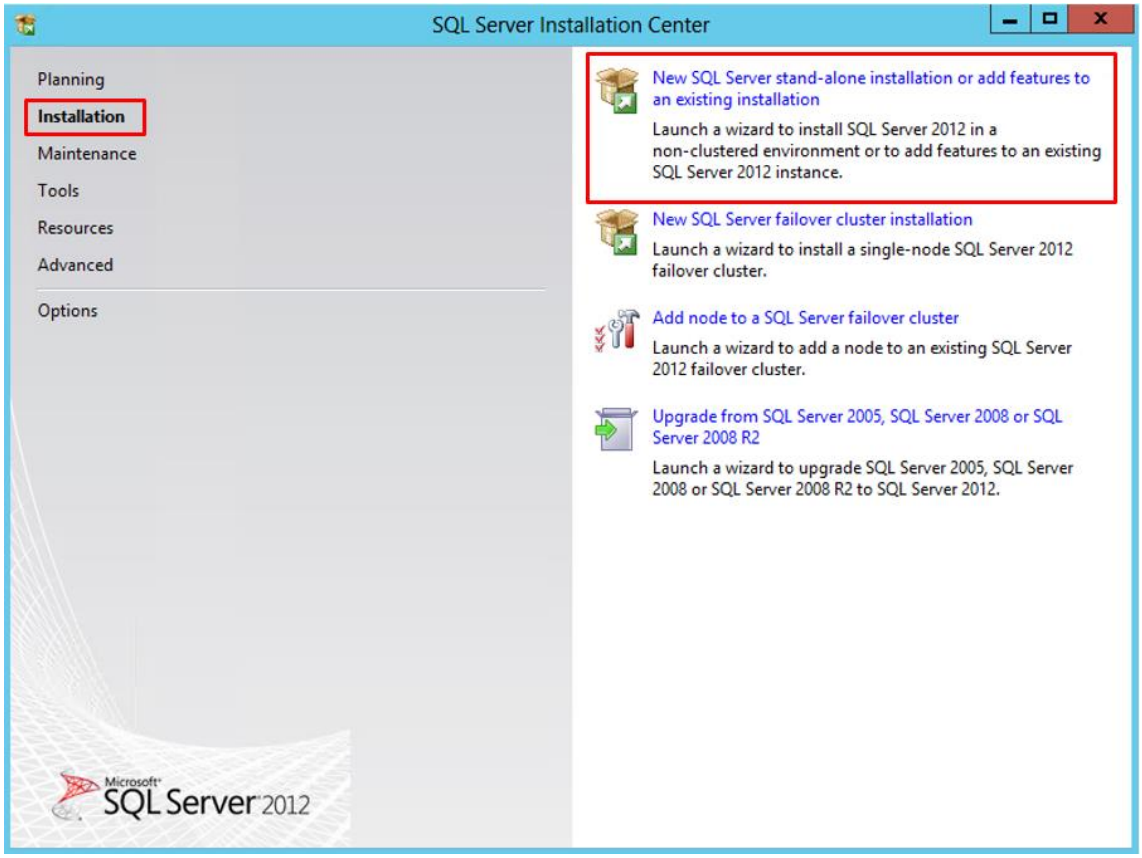

Abbildung 52: SQL Server Installation Center

Nach dieser Auswahl wird vom Installer der Produktschlüssel abgefragt, um das Produkt zu identifizieren und zu aktivieren. Es ist möglich, dass nach Abschluss dieser Eingabe ein Neustart des Computers notwendig ist, um diesen auf die Installation vorzubereiten. Der SQL Server bietet eine Vielzahl von Funktionen, die aber nicht alle benötigt werden. Unter dem Reiter „Setup Role“ und „Features Selection“ kann der Benutzer angeben, welche Funktionen er wie konfiguriert haben möchte (siehe Abbildung 53). Das Programm HeartBeat benötigt die grundlegenden Funktionen, welche es ermöglichen, die Daten zu überwachen, zu verwalten und sicher zu speichern. Darum liegt es nahe, den Punkt „SQL Server Feature Installation“ auszuwählen, da hier die wichtigsten Services installiert werden, die genau diese Funktionen beinhalten.

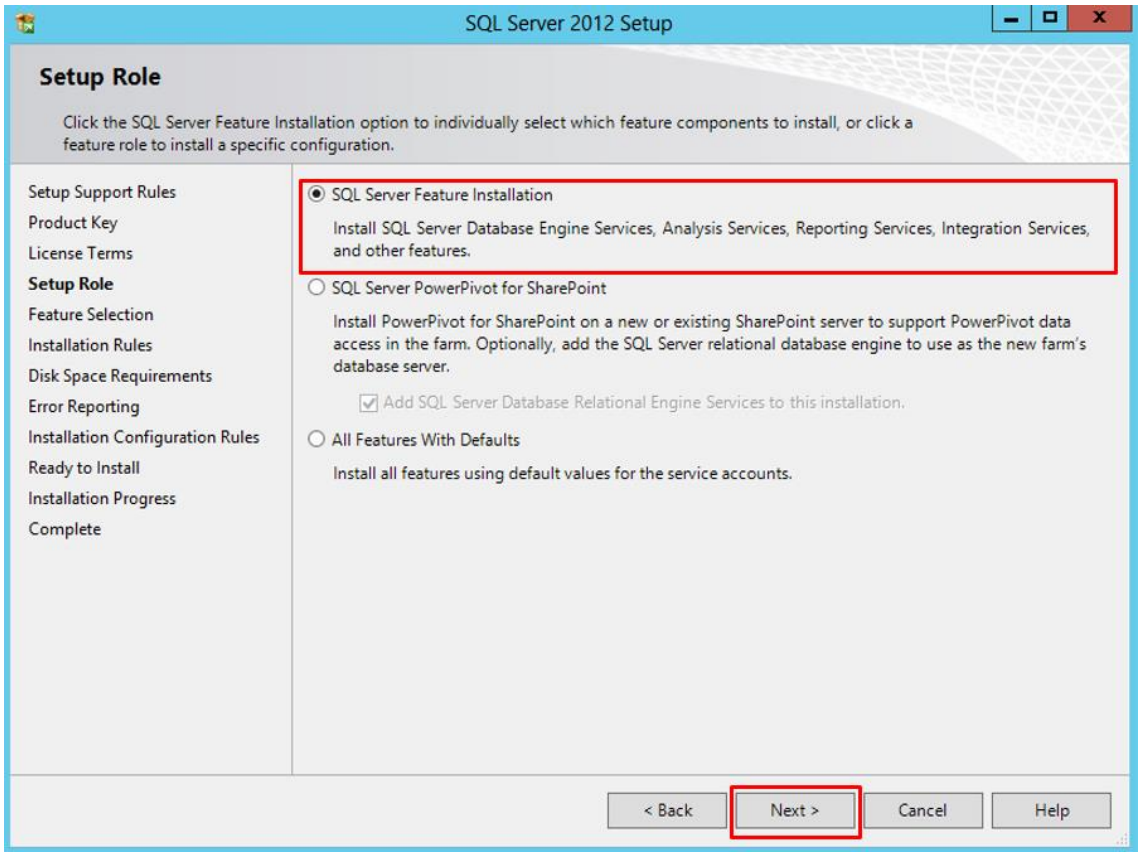

Abbildung 53: SQL Server 2012 Setup Role

Nachdem diese Funktionen definiert wurden, benötigt der Server eine Instanz. Diese wird mit einer Instanz ID und dem Pfad, wo diese gespeichert werden soll angelegt. Hier können die Standardwerte belassen und zum nächsten Reiter gesprungen werden. “Disk Space Requirements“ berechnet nun aus der vorher getroffenen Auswahl den benötigten Speicherplatz für das Programm. Nicht jeder Benutzer dieser Datenbankinstanz soll aber auf jede Funktion Zugriff haben. Dies wird im nächsten Schritt definiert. Im Bereich „Server Configuration“ kann für jede ausgewählte Funktion ein autorisierter Benutzer angegeben werden, welcher vollständigen Zugriff auf diese Funktionen hat. Auch kann angegeben werden wie ein Service starten soll. Es wird empfohlen, den „Startup Type“ mit „Automatisch“ zu definieren, um keine Komplikationen am Server zu erzeugen (siehe Abbildung 54).

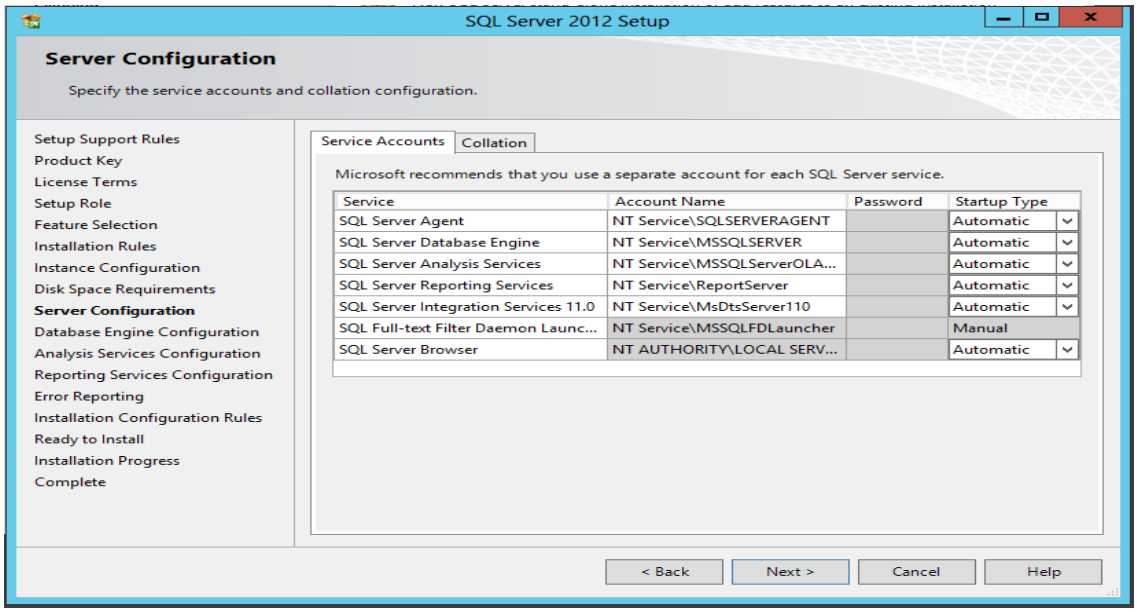

Abbildung 54: SQL Server 2012 Server Configuration

Um den Zugriff auf die eigentliche Datenbank Engine einzuschränken, kann im Installer ein Administrator angelegt werden. Dieser kann sich entweder mit der Windows Authentifizierungsmethode oder dem sogenannten Mixed Mode authentifizieren. Meist wird die Windowsmethode verwendet und erspart somit einiges an Konfiguration. Die nächsten Reiter befassen sich mit den Analyse- und Berichtfunktionen, welche mit den Standardwerten belassen werden können. Am Ende dieses Setups sollte es in etwa wie in Abbildung 55 aussehen.

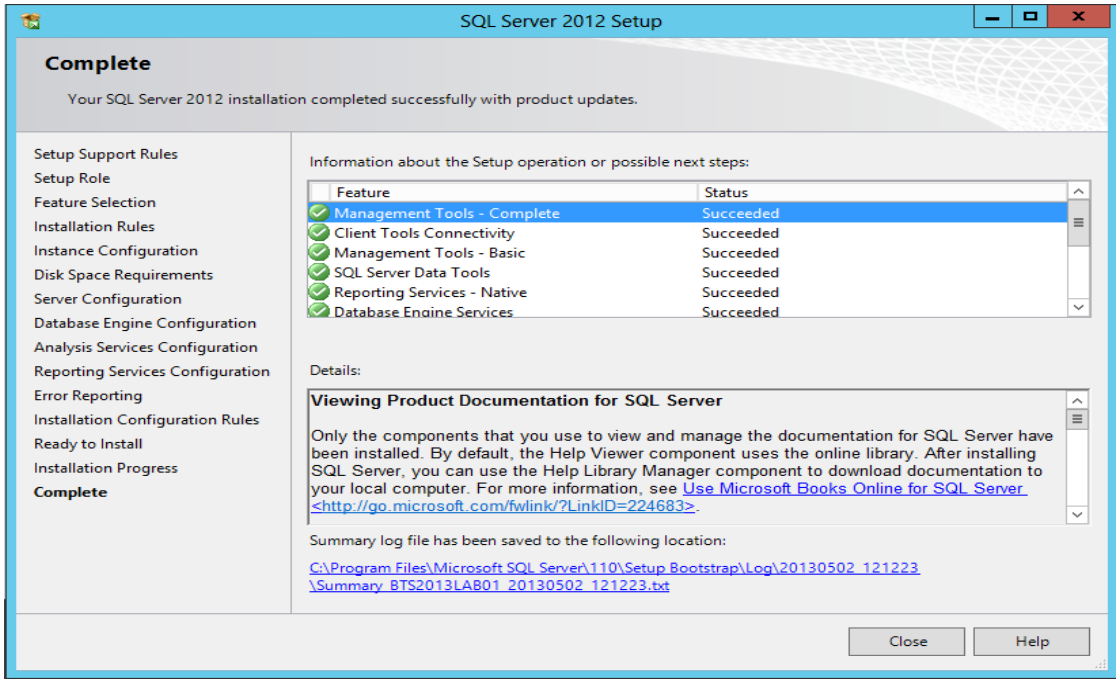

Abbildung 55: SQL Server 2012 Setup Complete

6 Implementierung

6.1 WCF Data Service

Zur Erstellung und Konfiguration des WCF Data Services haben wir die IDE Microsoft Visual Studio 2013 verwendet.

6.1.1 Erstellung

Um ein Data Service am Internet Information Server zu erstellen, wird zuerst eine leere Webseite angelegt. Wie bei jedem anderen Projekt wählt man „File“, dann „New“ und zuletzt „Web Site“.

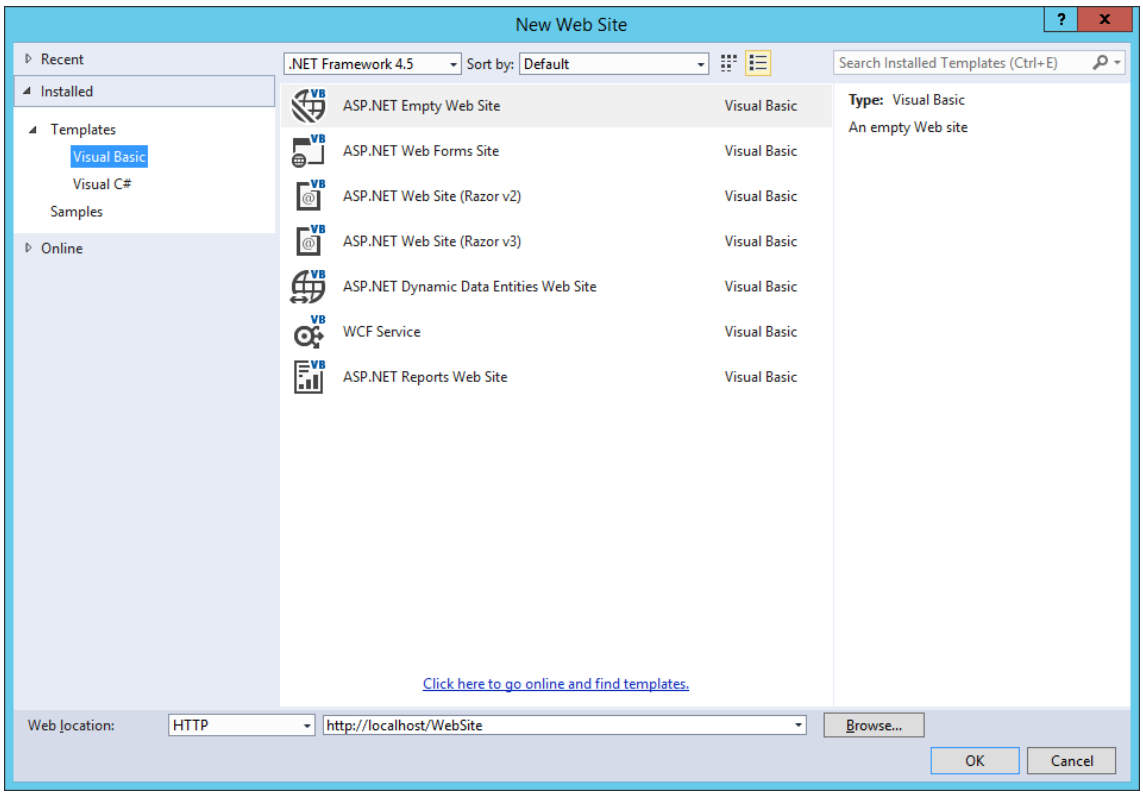

Abbildung 56: Erstellen einer Webseite

Im folgenden Fenster wählt man „ASP.NET Empty Web Site“, um eine leere Webseite zu erstellen. Unter „Web Location“ ist „HTTP“ auszuwählen. Unter „Browse“ kann dann noch der endgültigen Pfad auswählt werden. Hier wird oftmals der bereits installierte IIS gewählt. Zudem kann noch die IIS Seite für die Applikation gewählt werden. Den Namen der Webseite schreibt man nach der URL des Webservers. Bestätigt man die Eingaben nun mit OK, wird eine neue Webseite angelegt.

Zuerst werden die benötigten NuGet-Packages installiert. Für unseren Service wird nur das Entity Framework benötigt (siehe Beispiel 3).

```
Install-Package EntityFramework
```

Beispiel 3: Installieren des Entity Frameworks

Zur Installation dieses Pakets kann auch der grafische Paketverwalter unter „Tools“, „Library Package Manager“ und „Manage NuGet Packages for Solution…“ verwendet werden. Hier sucht man entweder nach „Entity Framework“ oder wählt dieses in den Vorschlägen aus.

Danach wird das Entity Framework Datenmodell erstellt. Dazu fügt man mit Rechtsklick auf seine Solution ein neues Element hinzu. Im folgenden Dialog wählt man „ADO.NET Entity Data Model“ aus und gibt dem Modell einen Namen. Nun wird nach der Erstellung eines „App_Code“ Ordners gefragt. Im folgenden Fenster wird „Code First from Database“ gewählt. Dabei wird auf Basis einer existierenden Datenbank ein zugehöriges Modell erstellt.

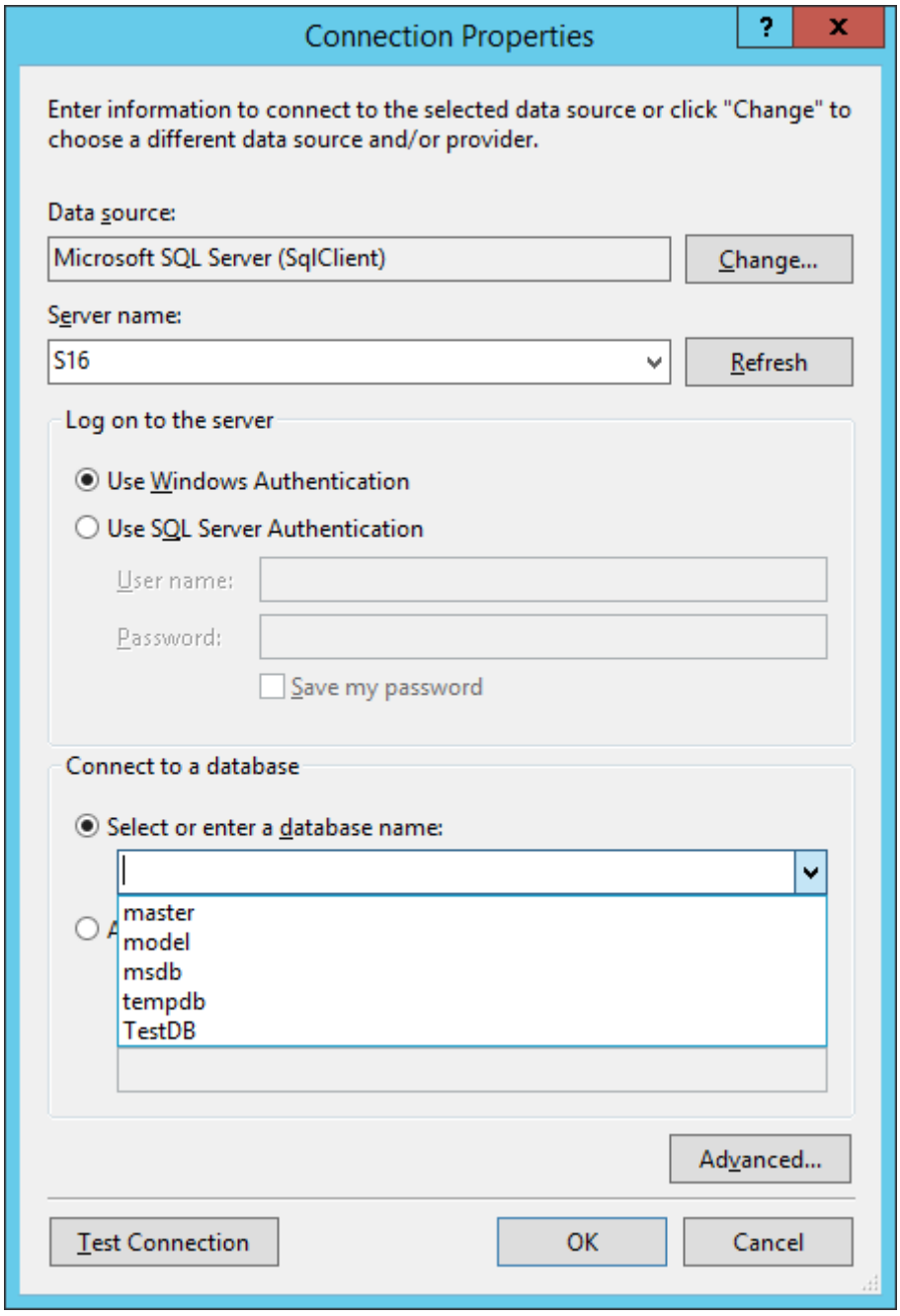

Abbildung 57: Verbindung zum SQL Server herstellen

Im nächsten Schritt muss die Datenbank, aus der das Modell erzeugt werden soll, gewählt werden. Dazu wählt man entweder eine bestehende Verbindung zu einer Datenbank aus oder erstellt eine neue Verbindung(siehe Abbildung 57Abbildung 57). Hier wird der Servername des SQL Servers eingetragen und im darunterliegenden Dropdown kann die gewünschte Datenbank ausgewählt werden.

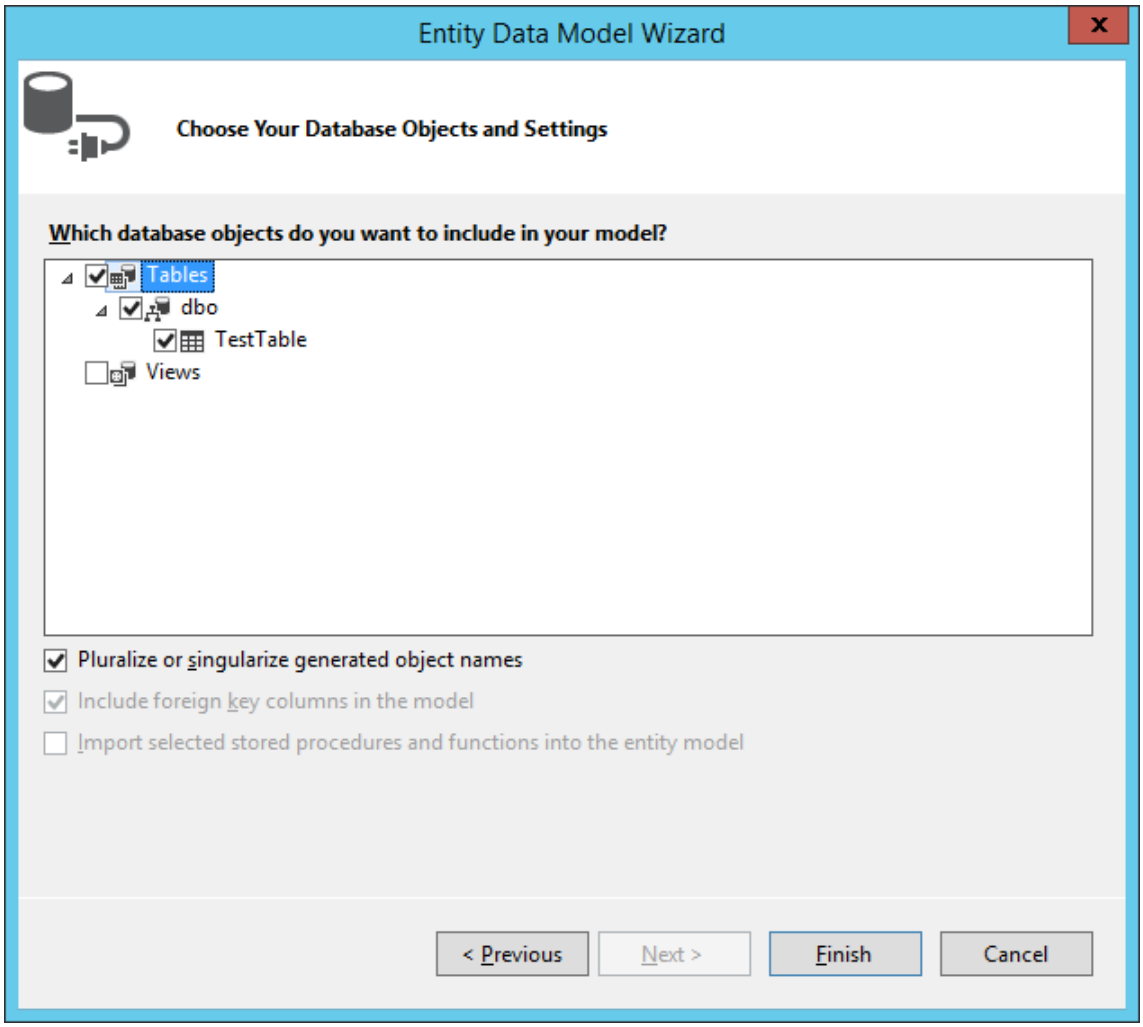

Abbildung 58: Auswahl der Tabellen oder Views

Danach werden die gewünschten Tabellen oder Ansichten der Datenbank ausgewählt (siehe Abbildung 58). Wird nun mit Finish bestätigt, wird das Modell erzeugt. Im letzten Schritt wird noch das Service erzeugt und konfiguriert. Dazu wird wiederum ein neues Element mittels Rechtsklick auf die Solution das Fenster zum Hinzufügen von Elementen geöffnet und „WCF Data Service“ ausgewählt.

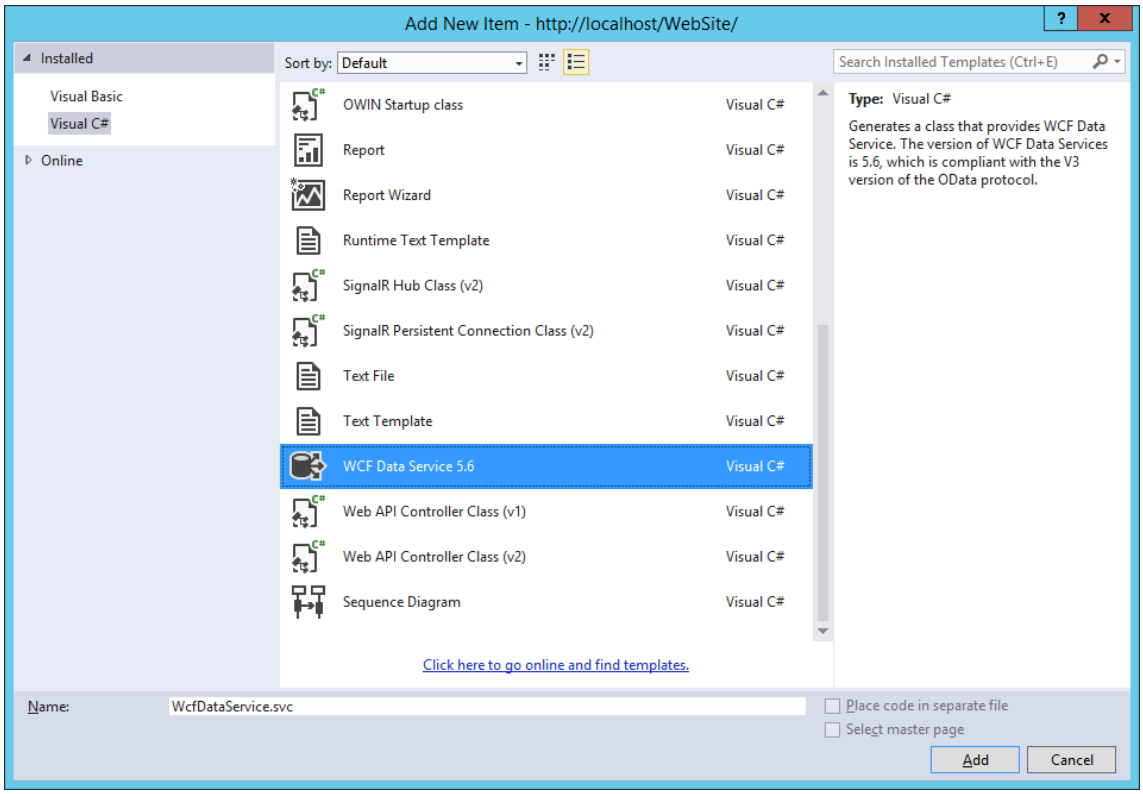

Abbildung 59: Hinzufügen neuer Elemente

Dadurch wird automatisch ein Service erzeugt. Hier muss in der Deklaration der Klasse noch das entsprechende Modell, das durch das Service zugreifbar sein soll, eingetragen werden (siehe Beispiel 4).

```
public class WcfDataService : DataService< /* TODO: put your data
source class name here */ >
```

Beispiel 4: Klasseninitialisierung WCF Service

Anschließend muss noch der Zugriff auf die Daten erlaubt werden (siehe Beispiel 5).

```
config.SetEntitySetAccessRule("*", EntitySetRights.AllRead);
```

Beispiel 5: Zugriffsregeln auf die Entitäten

Damit erlaubt man den Lesezugriff auf alle vorhandenen Entitäten.

Somit ist die Erstellung des Data Services abgeschlossen. Erreichbar ist dieser nun unter der IP-Adresse oder URL der IIS Webseite (z.B. http://localhost/TestService/TestService.svc).

Zusätzlich gibt es noch eine Reihe weiterer nützlicher Konfigurationsmöglichkeiten. Eine davon ist das Anzeigen detaillierter Fehlerbeschreibungen. Dazu werden zwei weitere Codezeilen benötigt (siehe Beispiel 6, Beispiel 7).

```
[ServiceBehavior(IncludeExceptionDetailInFaults = true)]
```

Beispiel 6: Detaillierte Fehlermeldungen anzeigen lassen

```
config.UseVerboseErrors = true;
```

Beispiel 7: Detaillierte Fehlermeldungen anzeigen lassen

6.1.2 Webmethoden erstellen

Um spezielle Abfragen oder das Einfügen von Daten über das Service zu erlauben, können Webmethoden erstellt werden, welche dann über eine URL und Get-Parameter aufgerufen werden können.

Es werden normale Methoden mit Rückgabewert in der Service C#-Datei erstellt, welche dann als Webmethoden gekennzeichnet werden.

```
[WebGet(UriTemplate ="/getNearUserLocations/{email}")]
public List<Location> getNearUserLocations(String email)
{
}
```

Beispiel 8: Webmethode erstellen

Hier wird am Beispiel einer Methode zur Berechnung naher Benutzer veranschaulicht, wie so eine Methode aussieht. Das Uri-Template Attribut definiert die URL, womit die Methode aufgerufen wird. Nach dem ersten Schrägstrich steht der Methodenname und nach dem folgenden in geschwungenen Klammern die Attribute, welche für die Methode benötigt werden. Siehe Beispiel 9 für einen Methodenaufruf.

```
http://SERVER-ADRESSE/SERVICE.svc/getNearUserLocations?email='test@email.com'
```

Beispiel 9: Beispielaufruf einer Webmethode

Anschließend wird die Methode ausgeführt und eine Liste von Standorten zurückgegeben.

6.2 Android App

Zur Entwicklung der Android-Applikation wird Eclipse mit dem ADT Plugin verwendet.

6.2.1 Erstellung einer Applikation

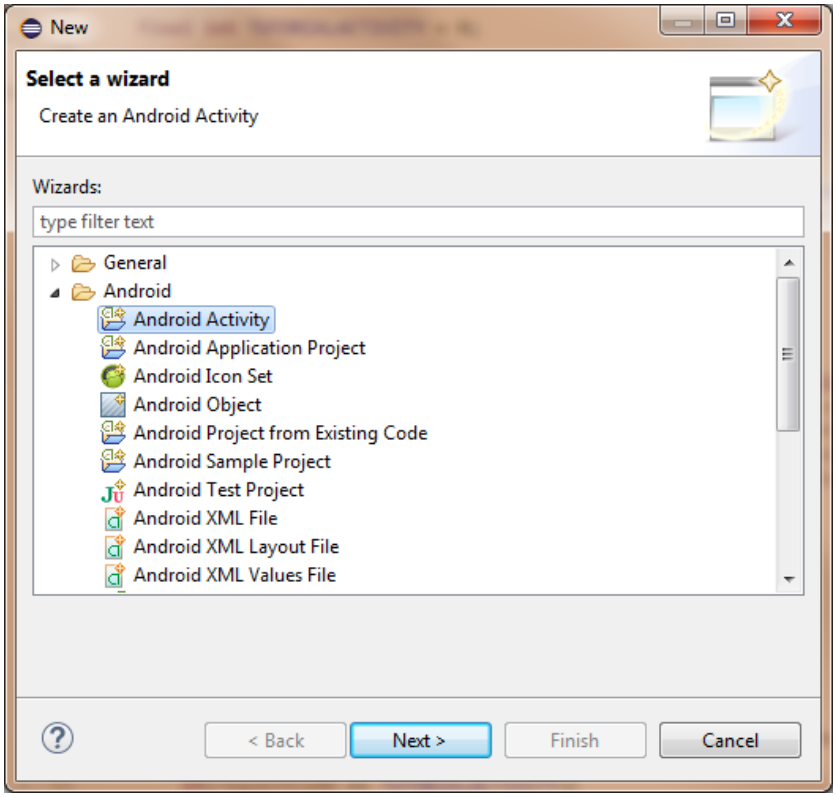

Abbildung 60: Erstellen einer Android Applikation

Um eine Android-Anwendung zu erstellen wird zuerst ein neues Projekt angelegt. Hierzu wählt man unter „File“, „New“ und „Other“ im Ordner Android „Android Application Project“ aus (Abbildung 60). Projektname und Package Namen werden im nächsten Schritt definiert. Ebenfalls kann angegeben werden, welches SDK zur Kompilierung der Android-Applikation herangezogen wird. Im nächsten Schritt kann ein eigenes Icon eingefügt und die Aktivität gewählt werden.

Hiermit steht das Grundgerüst einer Android-Applikation. Auf dieses aufbauend können nun weitere Klassen und Ansichten hinzugefügt werden.

6.2.2 Hinzufügen von Aktivitäten

Mit einem Rechtsklick auf das Projekt können weitere Elemente hinzugefügt werden. Eclipse bietet einen eigenen Wizard zur Erstellung weiterer Aktivitäten (siehe Abbildung 61).

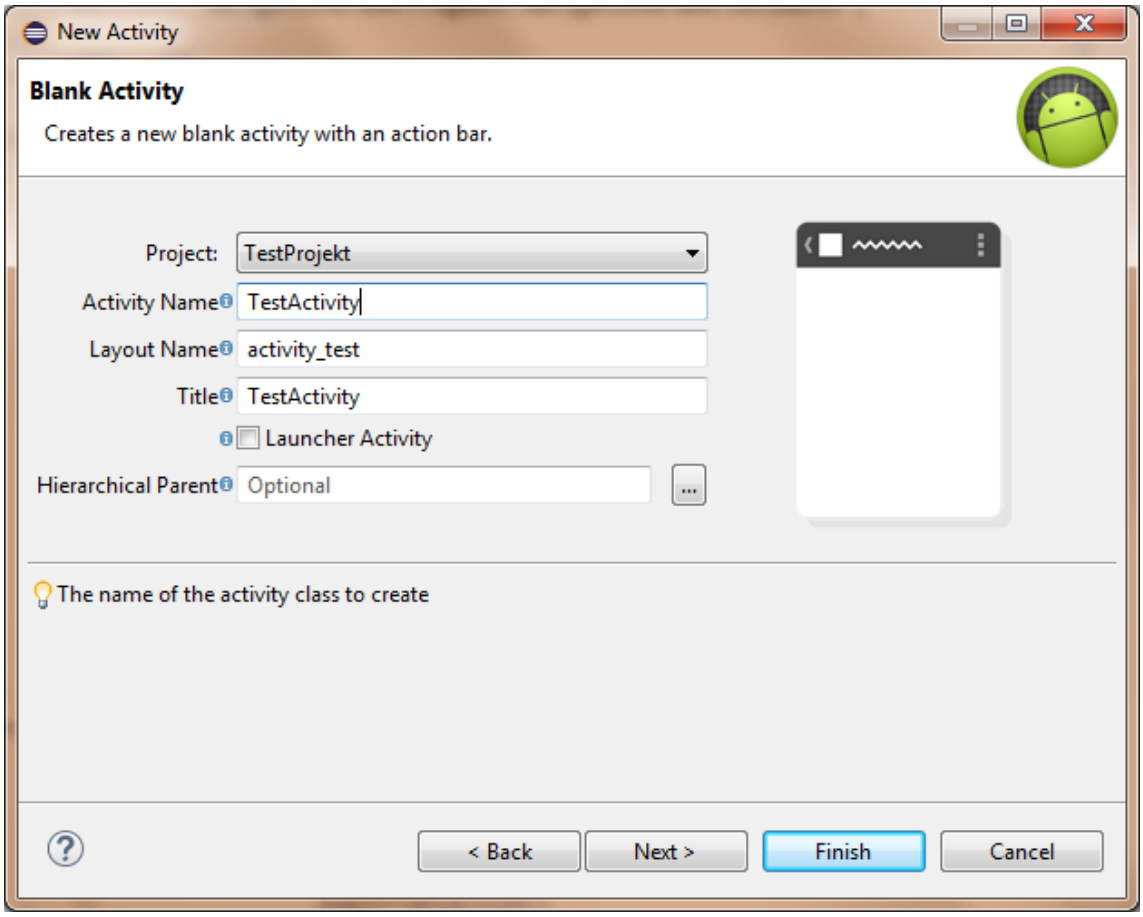

Abbildung 61: Erstellen einer neuen Aktivität

Abschließend wird eine Übersicht über die potentiellen Änderungen gezeigt, falls die Aktivität erstellt wird. Hauptsächlich werden zwei Dateien erzeugt. In einer XML Datei wird das Layout und die Elemente der Aktivität gespeichert. In einer Java Datei wird der Code, welcher beim Aufruf der Aktivität oder beim Druck auf ein Element ausgeführt wird, gespeichert.

6.2.3 Fragment

Eine Aktivität ist ein Container für verschiedene Benutzeroberflächen und diese kann viele Fragmente beinhalten. Ein Fragment kann mit einer „Mini-Aktivität“ verglichen werden. Jedes Fragment hat eine eigene Benutzeroberfläche und kann dynamisch verändert und hinzugefügt werden.

6.2.3.1 Lebenszyklus eines Fragments

Ein Fragment hat einen geregelten Ablauf im Programm. Wenn ein Fragment hinzugefügt wird, werden verschiedene Methoden nacheinander aufgerufen, welche bestimmte Funktionen erfüllen. Ein Fragment „lebt“ in einer sogenannten „ViewGroup“, welche die Hierarchie einer Aktivität darstellt. Alle unsere Benutzeroberflächen in der Applikation HeartBeat bauen auf diesen Fragmenten und deren Schemata auf (siehe Abbildung 62).

6.2.3.2 Fragmente verwalten

Fragmente werden mit dem „FragmentManager“ verwaltet. Dieser Manager ermöglicht es, verschiedene Fragmente zu löschen bzw. hinzuzufügen. Der Manager verwaltet die verschiedenen Fragmente und greift wenn nötig darauf zurück, ohne diese wieder komplett neu erstellen zu müssen. Um zwischen den einzelnen Fragmenten zu navigieren, ist in der Applikation eine eigene Methode implementiert worden (siehe Beispiel 10).

6.2.3.3 Aufbau eines Fragments

Um ein Fragment zu nutzen, muss es aber vorher noch eine Klasse geben, die von solchem Fragment erbt und verschiedene Methoden überschreibt. Der Grundaufbau solch einer Klasse beinhaltet die eigentliche Vererbung der Überklasse „Fragment“ und anschließend werden die wichtigsten Methoden überschrieben. In der Methode „onCreateView()” wird die Oberfläche des Fragments zurückgegeben (siehe Beispiel 11).

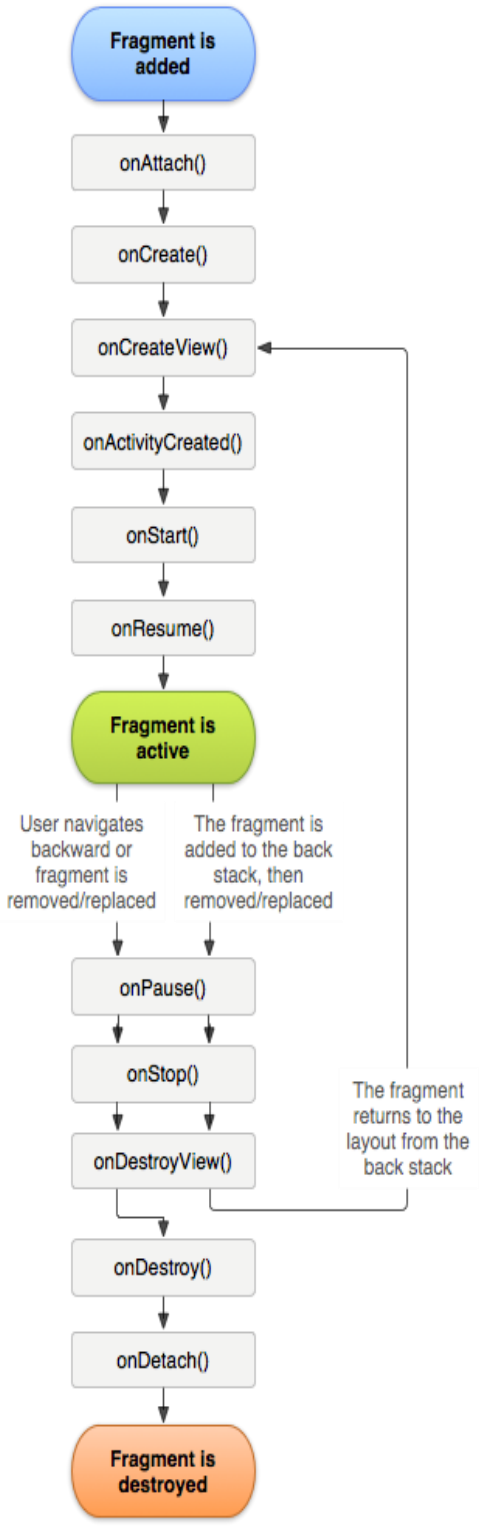

Abbildung 62: Lebenszyklus eines Fragments, (aus [11])

```
public void changeFragment(Fragment f, Bundle args) {
   f.setArguments(args);
   FragmentManager frgManager = getFragmentManager();
   FragmentTransaction ft = frgManager.
   beginTransaction();
   ft.setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right);
   ft.replace(R.id.content_frame, f);
   ft.commit();
}
```

Beispiel 10: Fragmente verändern

```
public class CurrentAlarms extends Fragment {
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
      View view = inflater.inflate(R.layout.activity_alarm,container,false);
      return view;
   }
   @Override
   public void onActivityCreated(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
   }
}
```

Beispiel 11: Fragment

6.2.4 Intent

Mit „Intents“ können Aktivitäten und Services in verschiedenen Anwendungen gestartet werden. Mit diesen „Intents“ können zum Beispiel auf die Anruferliste zugegriffen, SMS versendet und Emails geschickt werden, ohne dass die eigentliche Applikation diese Funktionen implementiert hat. Die Klasse „Intent“ fungiert sozusagen als Datenübergeber zwischen Aktivitäten. Beispiel 12 zeigt, dass eine Botschaft ganz einfach zu anderen Aktivitäten geschickt werden kann. In diesem Fall ist die andere Aktivität ein Service, der uns ermöglicht, mit verschiedenen sozialen Netzwerken zu kommunizieren und unsere App zu teilen.

```
String message = "Hi there! Check out the new app HeartBeat or visit our
website for further information http://heartbeat.htl-perg.ac.at";
Intent share = new Intent(Intent.ACTION_SEND);
share.setType("text/plain");
share.putExtra(Intent.EXTRA_TEXT, message);
startActivity(Intent.createChooser(share, "Share HeartBeat"));
```

Beispiel 12: Intent für soziale Netzwerke

6.2.5 JSON

Da wir vom Server Daten im JSON Format erhalten, wird zur Weiterverarbeitung der Daten der JSON String in POJOs umgewandelt. Dies geschieht mittels den Standard Java Klassen JSON-Object und JSON Array.

```
public static List<EmergencyType> parseJson(String json){
     List<EmergencyType> emergencyTypes = new ArrayList<EmergencyType>();
     try {
       JSONObject jsonobj = new JSONObject(json);
       JSONArray values;
       try{
         values = jsonobj.getJSONArray("value");
       } catch(JSONException ex){
         values = null;
       }
       if(values!=null){
         for(int i = 0; i < values.length(); i++){
           JSONObject emergencyType = (JSONObject)values.get(i);
           EmergencyType e = new EmergencyType();
           e.id = emergencyType.getInt("ID");
           e.name = emergencyType.getString("Name");
           e.description = emergencyType.getString("Description");
           e.counteraction = emergencyType.getString("Counteraction");
           emergencyTypes.add(e);
           Log.d("JSON", "More than one object");
         }
       } else {
         EmergencyType e = new EmergencyType();
         e.id = jsonobj.getInt("ID");
         e.name = jsonobj.getString("Name");
         e.description = jsonobj.getString("Description");
         e.counteraction = jsonobj.getString("Counteraction");
         emergencyTypes.add(e);
         Log.d("JSON", "One Object");
       }
     } catch(Exception e){
       Log.d("JSON", Log.getStackTraceString(e));
     }
     return emergencyTypes;
   }
```

Beispiel 13: JSON Parser

Die statische Methode „parseJson“ jedes POJOs erstellt aus einem JSON String eine Liste von Objekten des jeweiligen POJOs. In diesem Beispiel wird eine Liste von EmergencyType, also den Arten von Notfällen, erstellt.

Zuerst wird ein JSON Object aus dem String erstellt und danach geprüft, ob dieses ein Array von Objekten enthält. Wenn ein Array enthalten ist, bedeutet dies, dass mehrere Objekte enthalten sind. Ansonsten wurde nur ein Objekt zurückgegeben.

Danach wird entweder das Array durchlaufen und für jedes Element ein Objekt erstellt, die Attributwerte ausgelesen und zur Liste hinzugefügt, oder nur ein neues Objekt erstellt, die Attributwerte direkt aus dem ersten JSON Object ausgelesen und zur Liste hinzugefügt.

6.3 Client-Server Kommunikation

Zur Kommunikation mit dem Server werden vom Client HTTP Anfragen mit GET-Parametern gesendet und vom Server JSON codierte Daten zurückgeschickt.

```
URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)
obj.openConnection();

        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", "Android");

        int responseCode = con.getResponseCode();
        BufferedReader in = new BufferedReader(
                new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
          response.append(inputLine);
        }
        in.close();
```

Beispiel 14: Client-Server Kommunikation

Dazu wird zuerst ein Objekt der Klasse „HttpURLConnection“ erstellt und die Verbindung mithilfe vom URL Objekt geöffnet. Danach wird die Methode der Parameterübergabe und der User Agent definiert. Sobald die Antwort vom Server eintrifft, wird diese mithilfe eines gepufferten Lesers eingelesen. Zum Schluss wird die Verbindung geschlossen und der JSON String zurückgegeben.

7 Zusammenfassung

Das Kapitel beinhaltet eine Zusammenfassung und ein Resümee unserer erbrachten Leistungen. Ebenfalls werden Verbesserungsvorschläge und auch Erfahrungen festgehalten.

7.1 Erbrachte Leistungen

Da es bei der Diplomarbeit HeartBeat keinen externen Projektauftraggeber gibt, ist es uns seit Anfang der Entwicklungsarbeiten freigestanden, welche Funktionen wir wie und wann implementieren.

Kernfunktionen:

- Login
- Registrierung
- Notfälle absetzen / löschen
- Notfalldaten angeben / verändern
- Benutzerdaten angeben / verändern
- Unfälle anzeigen
- Navigation
- Erste Hilfe Maßnahmen anzeigen
- Notfallkontakte anlegen / löschen
- Notfallkontakte kontaktieren
- Position laufend speichern
- Benachrichtigungen anzeigen
- Menünavigation

7.2 Verbesserungen

Es wurden alle grundlegenden Funktionen implementiert, aber im Laufe der Entwicklungsarbeiten sind wir vermehrt auf verschiedene optionale Funktionen aufmerksam geworden, welche sich aber aus zeitlichen Gründe nicht implementieren lassen.

- Bild vom Unfallort
- Bild vom Unfallopfer
- Anbindung zu Rettungsdiensten
- HTTPS Verbindung zu unserem Server

8 Verzeichnisse

8.1 Internetverzeichnis

Link 1: https://eclipse.org/downloads/

Link 2: https://developer.android.com/sdk

Link 3: www.visualstudio.com

Link 4: http://www.microsoft.com/de-at/download/details.aspx?id=42299

8.2 Tabellenverzeichnis

8.3 Quelltextverzeichnis

8.4 Literaturverzeichnis

[1] Microsoft, “MSDN - Übersicht über ADO.NET,” 2015. [Online]. Available: https://msdn.microsoft.com/de-de/library/e80y5yhx(v=vs.110).aspx.

[2] Microsoft, “MSDN - Onlinedokumentation für SQL Server,” 2015. [Online]. Available: https://msdn.microsoft.com/de-de/library/ms130214(v=sql.110).aspx.

[3] Entity Framework Tutorial, “EntityFrameworkTutorial - entity framework introduction,” 2015. [Online]. Available: http://www.entityframeworktutorial.net/EntityFramework5/entity-framework5-introduction.aspx.

[4] S. S. Mumbare, “Codeproject - Windows Communication Foundation Basics,” 2011. [Online]. Available: http://www.codeproject.com/Articles/255114/Windows-Communication-Foundation-Basics. [Accessed 2015].

[5] Sachin Somanath Mumbare, “Codeproject - Windows Communication Foundation Basics,” 2015. [Online]. Available: http://www.codeproject.com/Articles/255114/Windows-Communication-Foundation-Basics.

[6] Microsoft, “MSDN - Erste Schritte mit LINQ,” 2015. [Online]. Available: https://msdn.microsoft.com/de-de/library/bb397933.aspx.

[7] B. A. Joseph Albahari, C# 5.0 in a Nutshell: The Definitive Reference, 2012.

[8] Microsoft, “MSDN - Übersicht über XAML,” 2015. [Online]. Available: https://msdn.microsoft.com/de-de/library/ms752059%28v=vs.110%29.aspx.

[9] P. Schmidt, “Windowsnetworking - Introduction to Internet Information Service 7.0,” 2007. [Online]. Available: http://www.windowsnetworking.com/articles-tutorials/windows-server-2008/Introduction-Internet-Information-Services.html. [Accessed 2015].

[10] Microsoft, “Technet - Konfigurieren der Authentifizierung in IIS,” 2015. [Online]. Available: https://technet.microsoft.com/de-de/library/cc731244(v=ws.10).aspx.

[11] Google, “http://android.com/,” 2015. [Online].

[12] Wikipedia - Android Betriebssystem, 2015. [Online]. Available: http://de.wikipedia.org/wiki/Android_%28Betriebssystem%29.

[13] Payload Media,
“http://www.techotopia.com/index.php/Android_4_App_Development_Essentials,” 2014. [Online].

[14] JSON Org., “http://www.json.org/json-de.html,” 2015. [Online].

[15] The Eclipse Foundation, “Eclipse,” 2015. [Online]. Available: https://www.eclipse.org/ide/.

[16] Microsoft, “Visual Studio,” 2015. [Online]. Available: www.visualstudio.com.

[17] Winhistory.de, 2015. [Online]. Available: winhistory.de.

[18] U. B. Boddenberg, Windows Server 2012 R2 - Das umfassende Handbuch, Rheinwerk Computing, 2014.

[19] Microsoft, “Microsoft - SQL Server,” 2015. [Online]. Available: www.microsoft.com.

[20] S. F. Conservancy, “Git,” 2015. [Online]. Available: http://git-scm.com/.

[21] Android Community, 2015. [Online]. Available: developer.android.com.

[22] S. Wenig, “dab-europe - Allgemeines zur Datenanalyse,” [Online]. Available: http://www.dab-europe.com/de/blog/allgemeines-zur-datenanalyse/ccm-teil-1-5-dinge-die-sie-wissen-sollten-und-2-dinge-die-sie-besser-nicht-vergessen-beim-aufbau-einer-ccm-datenanalyseumgebung.html#c956. [Accessed 2015].

[23] D. K. Bergner, P. D. M. Broy, D. A. Rausch and D. M. Sihling, *Betriebliche Informationssysteme,* Technische Universität München, 2002.

[24] Microsoft, “MSDN - ADO.NET Entity Framework,” 2015. [Online]. Available: https://msdn.microsoft.com/de-de/library/vstudio/bb399572(v=vs.100).aspx.

[25] Neil Smyth, “Techotopia,” 2015. [Online]. Available: http://www.techotopia.com/index.php/Android_4_App_Development_Essentials.

[26] syntevo GmbH, “Git Client SmartGit,” 2015. [Online]. Available: http://www.syntevo.com/smartgit/.

[27] P. Mehra, “c-sharpcorner - What is ADO.NET,” 2009. [Online]. Available: http://www.c-sharpcorner.com/uploadfile/puranindia/what-is-ado-net/. [Accessed 2015].

[28] J. Lerman, “thedatafarm - Data Access,” 2015. [Online]. Available: http://thedatafarm.com/data-access/. [Accessed 2015].