Technische Universität Braunschweig

Institut für Nachrichtentechnik

X-Map

Android-App zur qualitativen Erfassung von Mobilfunknetzen

Software-Entwicklungspraktikum (SEP)
Sommersemester 2013

Systemspezifikation

Auftraggeber
Technische Universität Braunschweig
Institut für Nachrichtentechnik
Prof. Dr.-Ing. Thomas Kürner
Schleinitzstraße 22
38106 Braunschweig

Betreuer: Dennis M. Rose

Auftragnehmer:

Name	E-Mail-Adresse
Sofia Ananieva	s.ananieva@tu-braunschweig.de
Andreas Bauerfeld	a.bauerfeld@tu-braunschweig.de
Ferhat Çinar	f.cinar@tu-braunschweig.de
Andreas Hecker	a.hecker@tu-braunschweig.de
Julia Kreyßig	julia@kreyssig.com
Timo Schwarz	t.schwarz@tu-braunschweig.de
Julian Troegel	j.troegel@tu-braunschweig.de
Deniz Yurtseven	d.yurtseven@tu-braunschweig.de

Braunschweig, 15. Mai 2013

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
0.0.1	01.05.2013	Timo Schwarz	abgenommen	Kapitel 2, 5 angepasst
0.0.3	09.05.2013	Julian Troegel	abgenommen	Kapiteln 2 erweitert
0.0.4	10.05.2013	Timo Schwarz	abgenommen	Kapiteln 1.1, 2, 3 erweitert
0.0.7	11.05.2013	Julian Troegel	abgenommen	Kapiteln 2, 3 überarbeitet
0.0.9	11.05.2013	Ferhat Cinar, Deniz Yurtseven	abgenommen	Kapitel 2 erweitert
0.1.0	11.05.2013	Julian Troegel	abgenommen	Kapitel 5 & Glossar erweitert
0.1.1	11.05.2013	Timo Schwarz	abgenommen	Kapitel 1 erweitert
0.1.2	11.05.2013	Sofia Ananieva	abgenommen	Kapitel 3 erweitert
0.1.3	11.05.2013	Ferhat Cinar	abgenommen	Kapitel 1.1 erweitert
0.1.4	11.05.2013	Andreas Bauerfeld	abgenommen	Kapitel 2 erweitert
0.1.5	12.05.2013	Andreas Hecker, Sofia Ananieva	abgenommen	Kapitel 2 erweitert
0.1.6	12.05.2013	Andreas Bauerfeld	abgenommen	Kapitel 2 erweitert
0.1.7	12.05.2013	Andreas Hecker	abgenommen	Kapitel 2 erweitert
0.1.8	12.05.2013	Julian Troegel	abgenommen	Kapitel 2 und 3 erweitert
0.1.9	12.05.2013	Sofia Ananieva	abgenommen	Kapitel 2 und 3 erweitert
0.1.10	12.05.2013	Julia Kreyßig	abgenommen	Kapitel 2 erweitert
0.1.11	12.05.2013	Ferhat Cinar	abgenommen	Kapitel 1.1 überarbeitet
0.1.12	12.05.2013	Deniz Yurtseven	abgenommen	Kapitel 1.1 erweitert
0.1.13	12.05.2013	Julia Kreyßig, Timo Schwarz	abgenommen	Kapitel 3.1 erweitert
0.1.14	12.05.2013	Sofia Ananieva	abgenommen	Kapitel 3.1.2 erweitert
0.1.15	12.05.2013	Andreas Hecker	abgenommen	Kapitel 5 erweitert
0.1.16	13.05.2013	Julian Troegel	abgenommen	Kapitel 1, 3, 5 erweitert
0.2.0	14.05.2013	Julia Kreyßig, Timo Schwarz	abgenommen	Kapitel 1 erweitert
0.2.1	14.05.2013	Julian Troegel	abgenommen	Kapitel 2 erweitert. NFA eingefügt
0.2.2	14.05.2013	Andreas Bauerfeld	abgenommen	Kapitel 3 erweitert. Backend
0.2.3	14.05.2013	Sofia Ananieva	abgenommen	Kapitel 3.3.2 erweitert

0.2.4	14.05.2013	Andreas Bauerfeld	abgenommen	Schnittstelle eingefügt
0.2.5	14.05.2013	Andreas Hecker	abgenommen	Schnittstelle eingefügt
0.2.6	14.05.2013	Andreas Bauerfeld, Andreas Hecker, Julian Troegel, Sofia Ananieva	abgenommen	Kapitel 5 erweitert
0.2.7	15.05.2013	Julia Kreyßig, Timo Schwarz	abgenommen	Kapitel 3.2 erweitert
0.2.8	15.05.2013	Julian Troegel	abgenommen	Kapitel 4 erstellt, Dokument formatiert
0.3	15.05.2013	Julian Troegel, Timo Schwarz	abgenommen	Abgabeversion

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

X-Map ist ein verteiltes System zur Aufzeichnung, Speicherung und Auswertung von Messdaten zum Mobilfunknetz. Es setzt sich prinzipiell zusammen aus

- ... einer Android-Applikation, welche auf Mobilgeräten, auf denen sie installiert ist, Messdaten sammelt.
- ... einem WebService, der für Benutzerverwaltung und Messdatenspeicherung verantwortlich ist.
- ... einer Web-Anwendung, welche Möglichkeiten zur Visualisierung und Auswertung der Messdaten bereitstellt.

Kapitel 2 dieses Dokuments befasst sich mit der Erläuterung der im Pflichtenheft festgelegten Produktfunktionen und deren geplanter Umsetzung. Kapitel 3 beschreibt, zu welcher Architektur die Analyse der Funktionen in Kapitel 2 geführt hat. Kapitel 4 stellt mithilfe eines UML-Deployment-Diagramms dar, in welcher Weise die oben beschriebenen Strukturen verteilt sind. In Kapitel 5 wird darüber Buch geführt, welche Kriterien bereits auf welche Weise umgesetzt wurden und welche noch zu bearbeiten sind.
Der Detailgrad der einzelnen Ausführungen in den jeweiligen Unterkapiteln ist dabei durch den aktuellen Stand der Entwicklung beschränkt.

1.1 Projektdetails

Die in Kapitel 1 angeschnittenen Grundkomponenten werden in der Folge durch Aktivitätsdiagramme bzw. Statecharts detailliert. Als Grundlage dienen die geplanten Normalfälle der Benutzung der einzelnen Komponenten.

1.1.1 Android Applikation

Es wird ein typischer Ablauf der Android Applikation beschrieben.

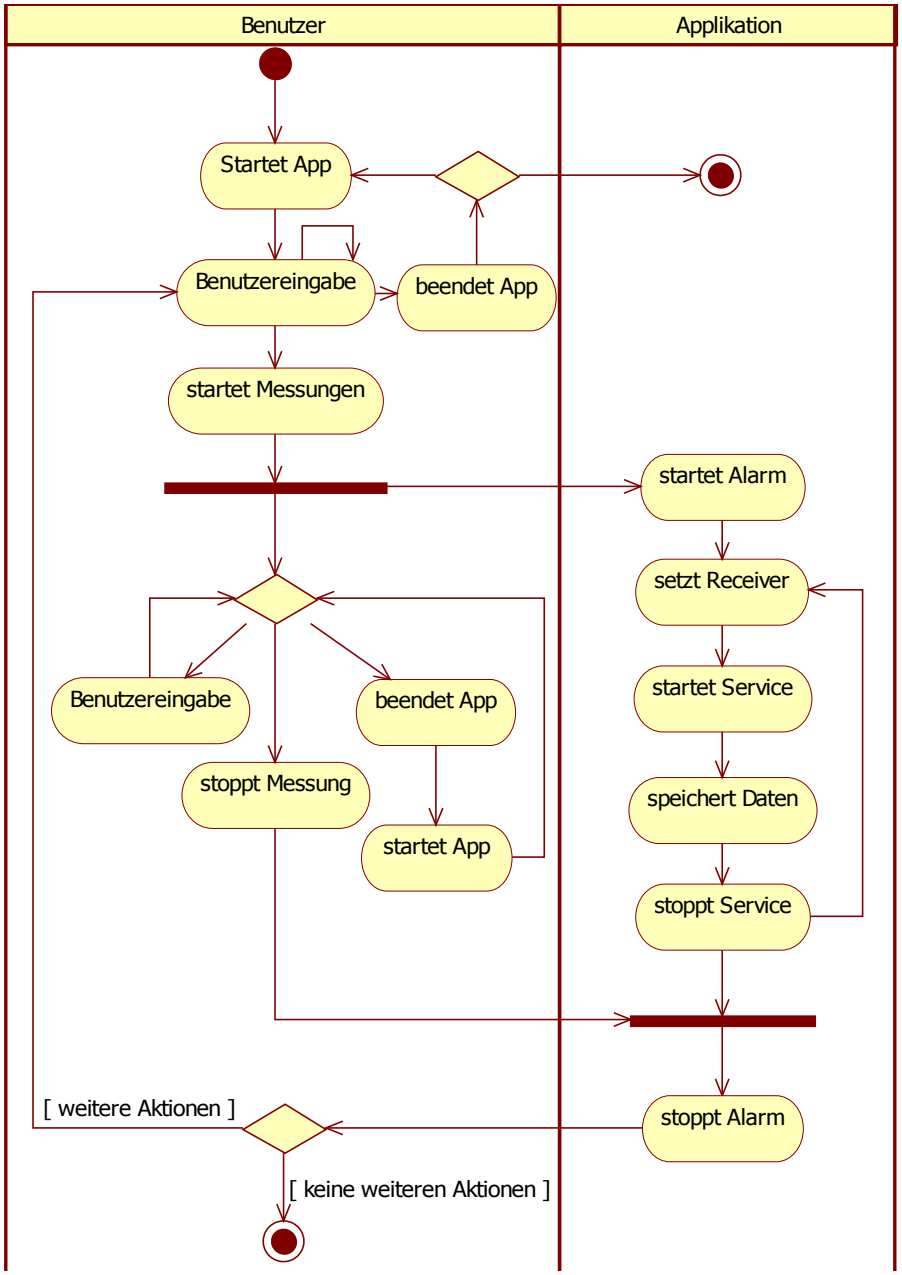

Abbildung 1.1: Aktivitätsdiagramm zur Benutzung der Android Applikation

Abbildung 1.1 stellt einen typischen Arbeitsablauf für einen Benutzer dar, welcher Messungen auf der Android Applikation durchführen lässt.
Zunächst muss der Benutzer die App auf seinem Mobilgerät starten. Dabei wird angenommen, dass der Benutzer die App vorher schon installiert hat. Daraufhin kann der Benutzer verschiedene

Benutzereingaben durchführen. Typische Benutzereingaben sind dabei der Start der Visualisierungen, die Einstellungen bearbeiten, sich anmelden oder registrieren, Übertragungen an den WebService durchführen lassen und weitere. Diese Benutzereingaben werden hier nicht weiter spezifiziert. Der Benutzer kann aber auch, ohne Messungen durchzuführen, die App beenden. Entweder ist der Ablauf dann beendet oder der Benutzer startet die App neu.
Sobald der Benutzer die Messungen starten lässt, wird innerhalb des Mobilgerätes ein Alarm gesetzt, der die periodischen Zeitpunkte für einen Receiver setzt. Der Alarm bindet sich dabei in das Android-System und kann damit auch ausgeführt werden, wenn die App nicht aktiv ist oder das Mobilgerät im Standby ist.
Der gestartete Receiver besitzt dabei eine sehr kurze Lebensdauer, da dieser vom Android-SDK bereitgestellt wird und somit vom ressourcensparenden Betriebssystem nach sehr kurzer Zeit beendet wird. Deshalb startet dieser einen Service, der länger überleben kann und die Messungen durchführt und anschließend die Daten in der Datenbank speichert. Danach beendet der Service sich selbst. Zum nächsten vom Alarm gesetzten Zeitpunkt wird der Receiver wieder gestartet und der Ablauf wiederholt sich.
Parallel zu diesem sich immer wiederholen Ablauf, kann der Benutzer verschiedene weitere Eingaben machen. Zum Einen sind die oben beschriebenen Benutzereingaben möglich und zum Anderen kann der Benutzer auch die App beenden. Dadurch, dass der Alarm sich in Android-System gebunden hat, läuft der Vorgang der Messung weiter ab. Der Benutzer muss dann die App wieder starten, um weitere Aktionen durchführen zu können. Andererseits kann der Benutzer die Messungen auch stoppen. Dadurch wird der Alarm beendet und aus dem Android-System geschrieben.
Nun können weitere Aktionen durchgeführt werden, wenn der Nutzer dieses möchte.

1.1.2 WebService

Es wird ein typischer Ablauf des WebServices beschrieben.

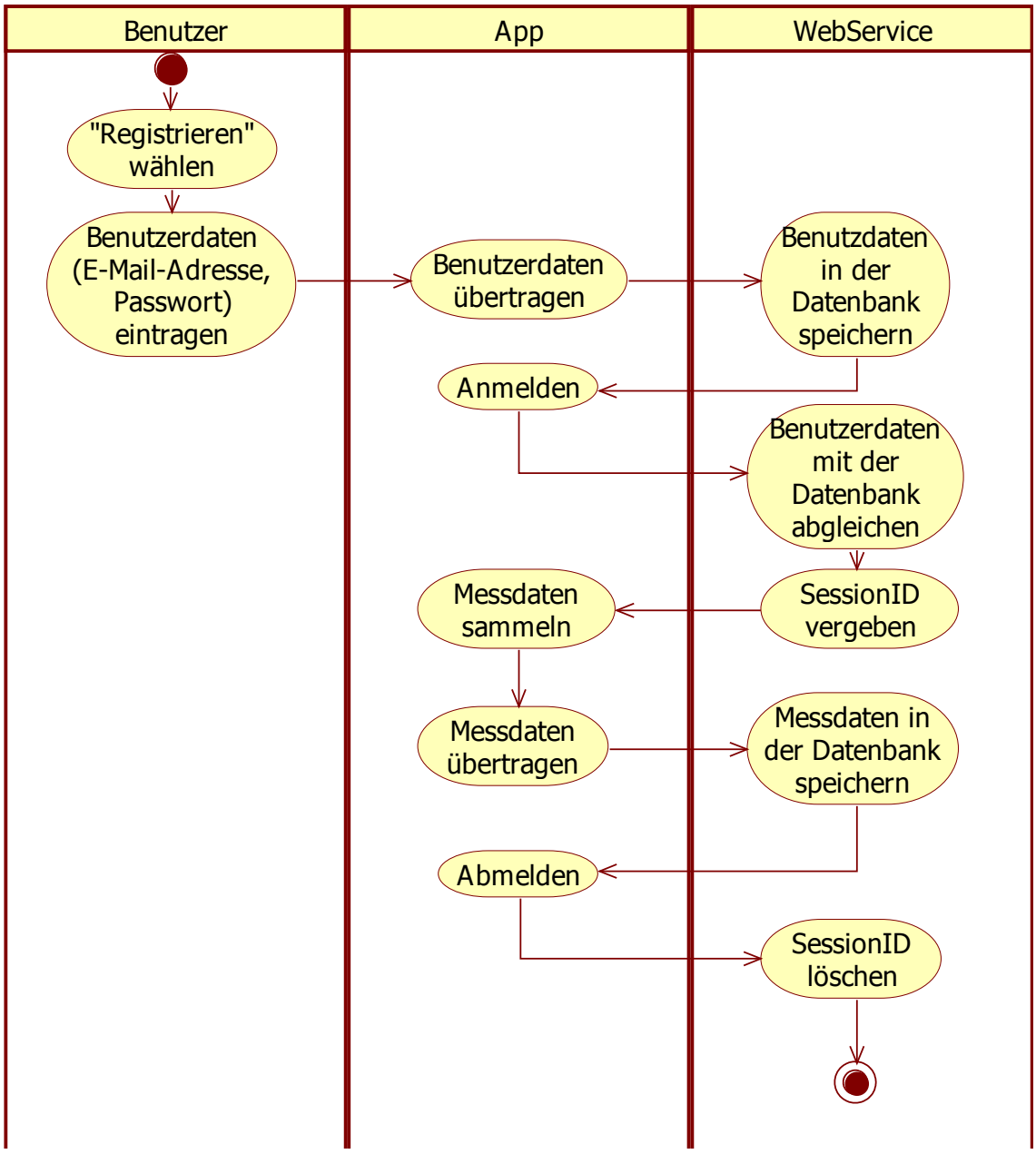

Abbildung 1.2: Aktivitätsdiagramm zur Interaktion zwischen App und WebService

Abbildung 1.2 beschreibt den typischen Ablauf, wenn ein Nutzer die App installiert hat und nun das Übertragen von Daten an den WebService ermöglichen möchte.
Hierfür startet der Nutzer zunächst die Registrierungsprozedur in der App und gibt seine Daten ein. Diese überträgt die App dann zusammen mit Informationen über das verwendete Gerät an den WebService, welcher den Datensatz in die Datenbank einträgt. (Sind die Nutzerdaten

bereits vorhanden, erhält die App hierüber Rückmeldung.) Ist die Registrierung erfolgt, meldet sich die App automatisch mit den neuen Nutzerdaten an, woraufhin sie vom WebService eine SessionID erhält.
Die App verpackt nun die Messdaten in das vorgegebene Format und übermittelt sie an den WebService, welcher sie an die Datenbank überträgt. Nun kann sie die App vom WebService abmelden, woraufhin die SessionID gelöscht wird.

1.1.3 Web Applikation

Es wird ein typischer Ablauf der Web Applikation beschrieben.

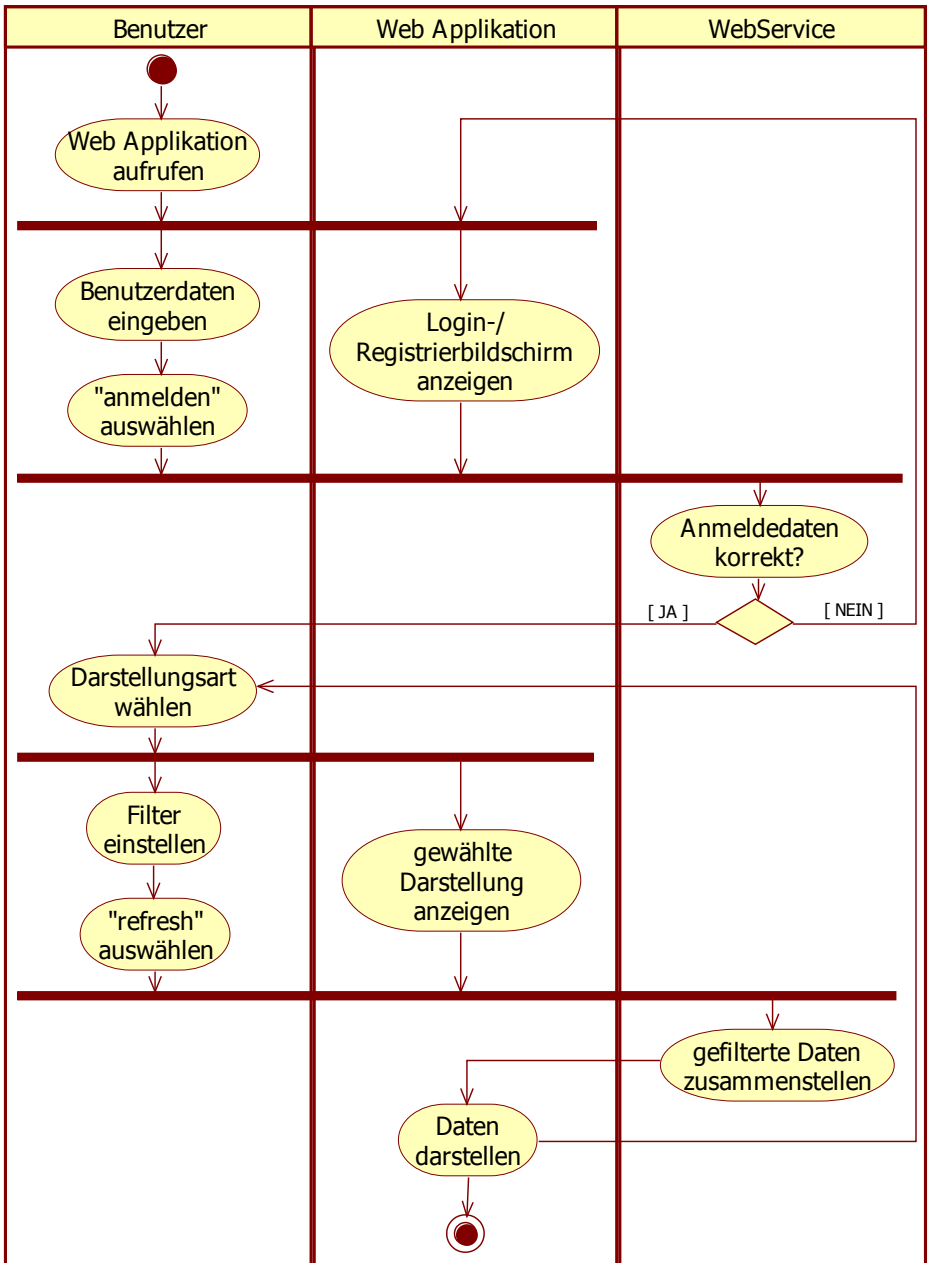

Abbildung 1.3: Aktivitätsdiagramm zur Benutzung der Web Applikation

Abbildung 1.3 stellt einen typischen Arbeitsablauf für einen Forscher dar, welcher sich die von X-Map gesammelten Messdaten im Browser ansehen möchte.
Zunächst muss die Webseite der X-Map WebApp aufgerufen werden. Daraufhin muss sich der Benutzer anmelden, wobei hier vorausgesetzt wird, dass er bereits registriert ist. Hatte er das

nicht, ist die hier nicht abgebildete Registrierfunktion zu nutzen. Nach der Verifizierung der eingegebenen Daten wird dem Nutzer zunächst ein Willkommensbildschirm angezeigt, von welchem aus er die gewünschte Darstellungsart wählen kann. Diese wird daraufhin (zunächst ohne das Einblenden von eigentlichen Messdaten) aufgerufen. In der Folge kann der Nutzer entsprechende Filter auswählen, sodass er nur die Daten angezeigt bekommt, die er haben möchte. Schließlich werden die gefilterten Daten vom WebService an die Applikation zurückgeschickt, welche sie nun darstellen kann.

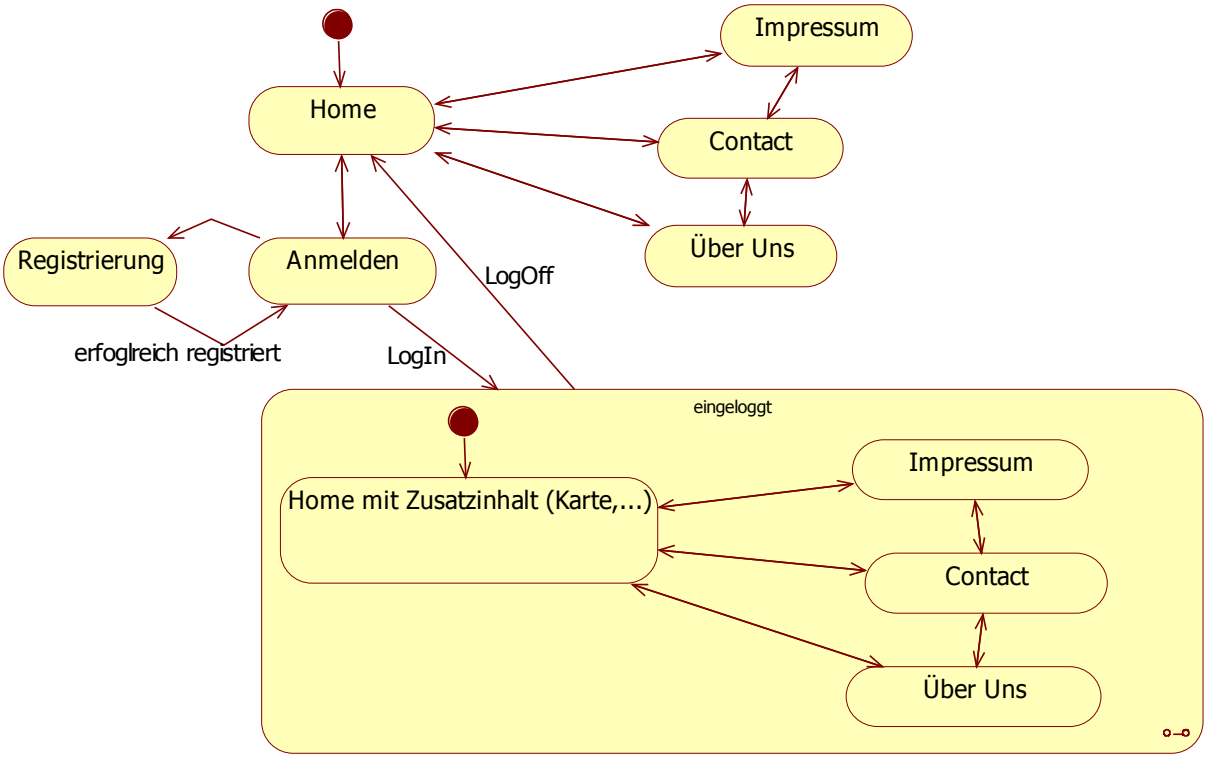

Abbildung 1.4: Statechart zur Web Applikation

Abbildung 1.4 stellt die möglichen Zustände dar, welche die Web Applikation besitzen kann. Auf der Startseite befindet man sich im Startzustand. Von dort aus kann man zur „Login“-Seite, zum „Impressum“ oder über die Menüleiste zu den Seiten „Contact“ und „Über uns“ wechseln, auf der sich nähere Informationen zu den Entwicklern befinden. Falls der Anwender noch kein Benutzerkonto besitzt, gelangt man von der „Login“-Seite zur „Register“-Seite. Wurde über diese erfolgreich ein Benutzerkonto erfolgreich erstellt, wird man zur „Login“-Seite zurückgeleitet. Falls sich ein Fehlversuch beim Login oder der Registrierung ereignet, bleibt der Zustand in den jeweiligen Seiten erhalten. Kommt es zu einem erfolgreichen Login, befindet sich die WebApp im Oberzustand „eingeloggt“ und dem Unterzustand „Startseite“, mit „Karte“ oder „Tabelle“ als Menüpunkte. Auch im eingeloggten Zustand kann der Benutzer zu den Seiten „Contact“, „Über uns“ und „Impressum“ wechseln, jedoch ohne den Oberzustand zu verlassen. Falls sich der Benutzer ausloggt, befindet man sich wieder auf der Startseite. Die Aktion „LogOff“ kann

jederzeit ausgeführt werden

2 Analyse der Produktfunktionen

In den folgenden Unterkapiteln werden die im Pflichtenheft gefundenen Produktfunktionen analysiert und die Art und Weise ihrer geplanten Lösung mithilfe von Sequenz-Diagrammen konkretisiert.

2.1 App

Die folgenden Funktionalitäten $\langle F10 \rangle$ bis $\langle F80 \rangle$ beziehen sich hauptsächlich auf die Android-Applikation.

2.1.1 Analyse von Funktionalität $\langle F10 \rangle$: Daten messen

Auf dem Mobilgerät werden die Daten des Mobilfunknetzes gemessen und in Relation zur Position des Gerätes gesetzt.

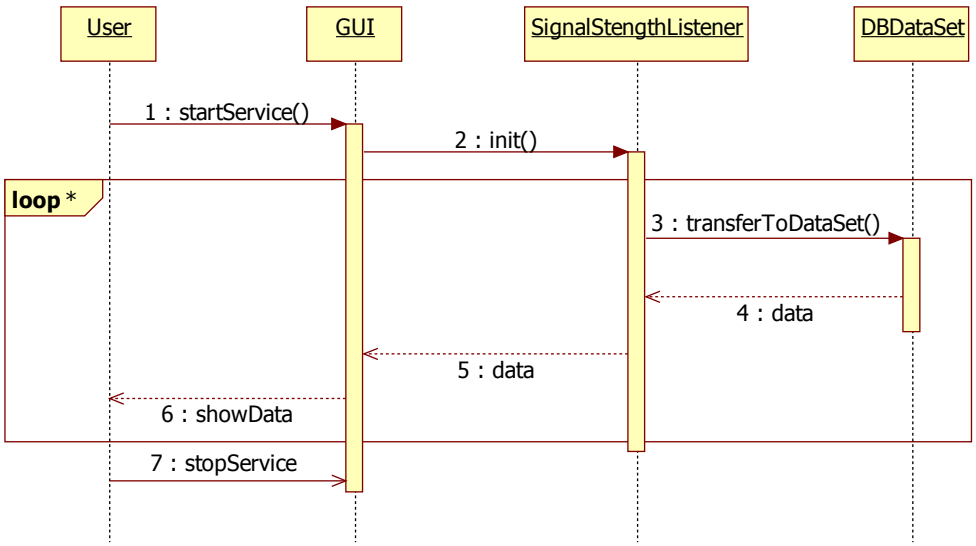

Abbildung 2.1: Sequenzdiagramm zum Messen von Daten

Der User startet die App (1). Nach dem Start wird der SignalStrengthListener initialisiert (2). Der SignalStrengthListener beinhaltet alle Methoden zum Messen der gewünschten Daten. Bei einer Änderung der Signalstärke wird automatisch eine neue Messung der Daten vorgenommen. Die gemessenen Daten werden in einen temporären Datensatz DBDataSet geschrieben (3). Anschließend wird der Datensatz an das User Interface weitergeleitet (4)+(5) und die Daten ausgegeben (6).

2.1.2 Analyse von Funktionalität $\langle F20 \rangle$: Visualisierung der Android-Applikation

Die auf dem Mobilgerät gespeicherten Messdaten werden durch einen Graphen, eine Tabelle und GoogleMaps visualisiert.

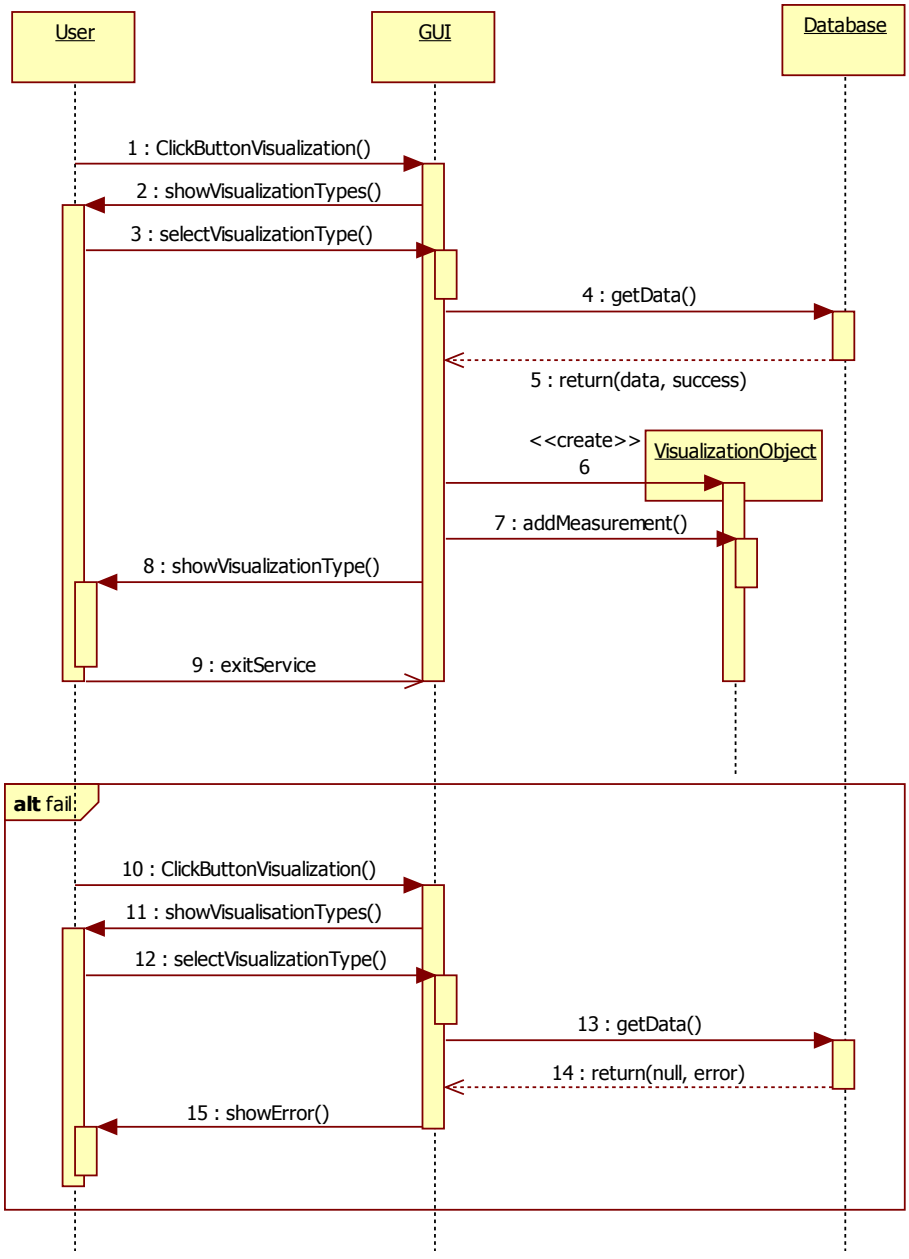

Abbildung 2.2: Sequenzdiagramm zur Visualisierung lokal gespeicherter Daten

Um seine Messdaten zu sehen, kann der Nutzer auf den Visualisierungsbutton klicken. Anschließend kann er zwischen bestimmten Visualisierungstypen wählen: Einem Graph, einer Tabelle oder GoogleMaps. Durch einen Lesezugriff der GUI auf die unterliegende Datenbankschicht,

wird das gewünschte Visualisierungsobjekt mitsamt der Messdaten von der GUI erstellt und dem Nutzer angezeigt. Sobald das Visualisierungsobjekt nicht mehr benötigt wird, wird es vom Android-System zerstört. Alternativ wird der Nutzer über eine Fehlermeldung benachrichtigt, falls aus Fehlergründen keine Visualisierungen angezeigt werden.

Sobald neue Daten gemessen werden, aktualisiert sich der Graph, die Tabelle als auch Google-Maps.

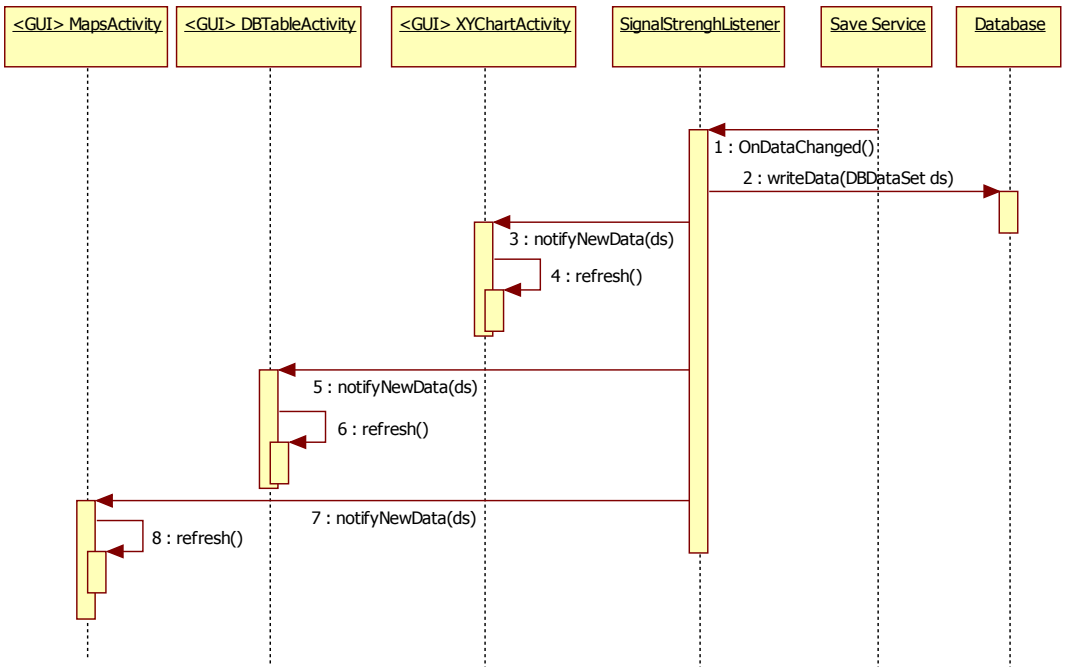

Abbildung 2.3: Sequenzdiagramm zur Kommunikation zwischen neuen Daten und GUI Activities

Das Backend wird vom Save Service in periodischen Abständen benachrichtigt, weitere Daten zu messen. Die neuen Daten werden daraufhin vom Back-end in die Datenbank geschrieben. Anschließend werden alle GUI-Listener (XYChart, DBTable, Maps) des Back-end über ein Interface mitsamt der neuen Daten benachrichtigt. Die GUI Activities aktualisieren ihre Ansicht.

2.1.3 Analyse von Funktionalität $\langle F30 \rangle$: Daten für WebService

Das Mobilgerät macht sich am Server bekannt und kündigt so das Senden von Daten an. Daraufhin werden zur Verfügung stehende Datensätze an den Server übermittelt.

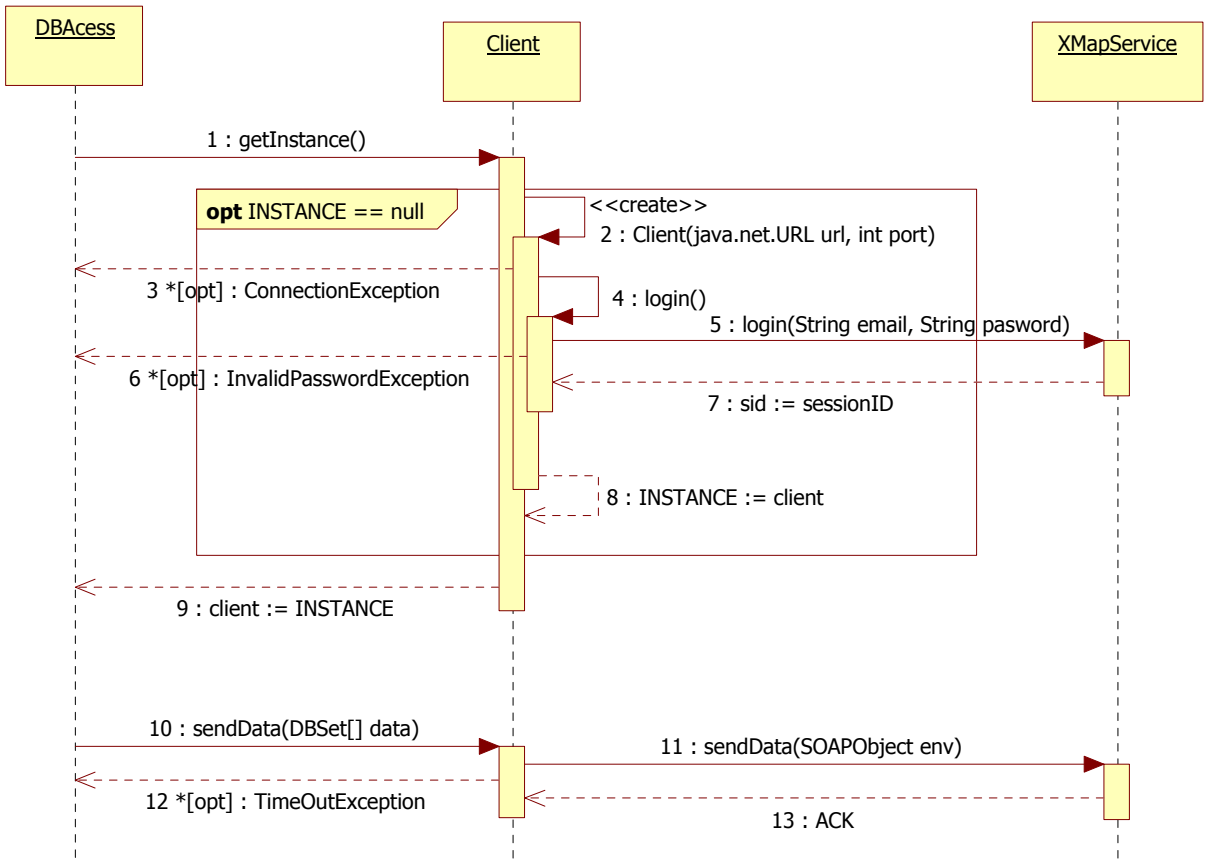

Abbildung 2.4: Sequenzdiagramm zum Anmelden und Übertragen von Daten

Client ist eine Klasse des Design Patterns 'Singleton', welche den Datenaustausch über SOAP implementiert. DBAcess braucht eine Instanz von Client zur Übertragung (1). Ist momentan kein Client-Objekt vorhanden wird dieses erzeugt und mit dem Server verbunden (2). Sollte es Verbindungsfehler geben wird eine ConnectionException zurückgegeben (3). Danach wird das Login ausgeführt (4)+(5). Sollte das Passwort falsch sein und der Server keine Antwort geben wird eine InvalidPasswordException zurückgegeben (6). Läuft alles gut bekommt der Client eine SessionID von X-MapService zurück, welche er für die künftige Verwendung speichert (7). Die Client-Klasse gibt die Referenz auf das Objekt an DBAcess (9). Mit der Referenz kann DBAcess nun sendData aufrufen welche als Parameter ein Array aus Datensätzen enthält (10). Der Client bereitet diese Daten auf und sendet sie an den X-MapService (11), dieser bestätigt den Empfang (13). Kommt es zu Fehlern wird eine TimeOutException erzeugt (12).

2.1.4 Analyse von Funktionalität $\langle F40 \rangle$: Daten löschen

Alte bzw. nicht mehr benötigte Datensätze werden vom Mobilgerät unwiederbringlich gelöscht. Dazu gibt es zwei Möglichkeiten, wie es zu einer Datenlöschung kommt. Zum Einen beim Einfügen von Daten (Abbildung 2.5) und zum Anderen bei einer Einstellungsänderung der maximalen Speichergröße (Abbildung 2.6)

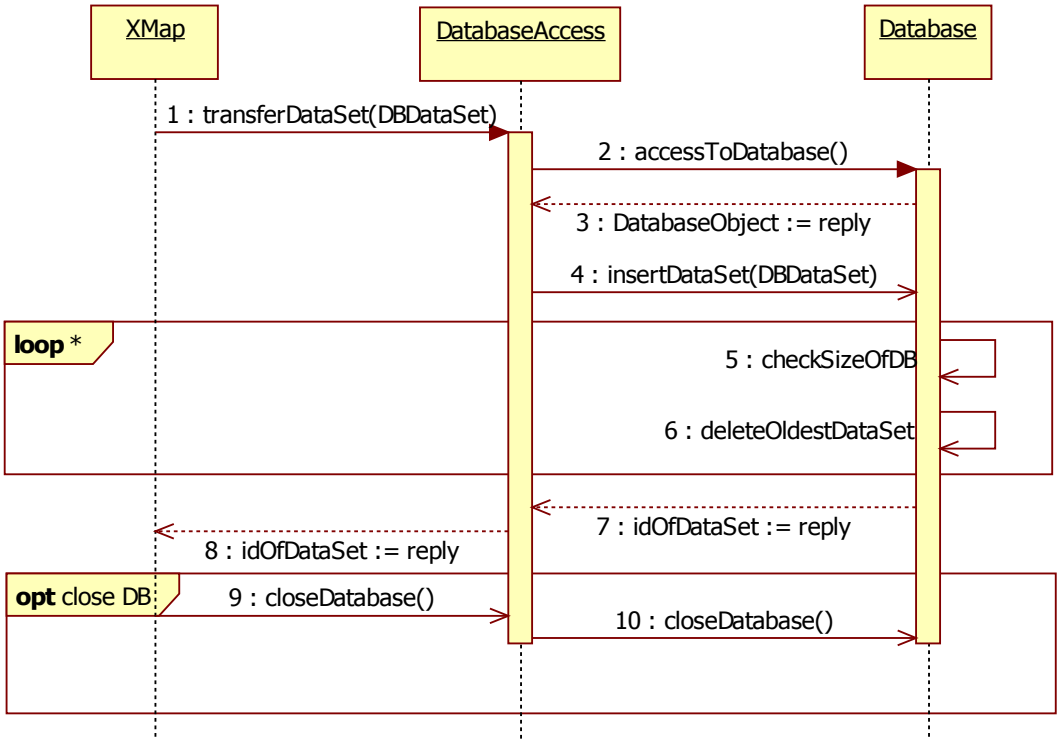

Abbildung 2.5: Sequenzdiagramm zum Löschen von Daten beim Einfügen von Daten

Die X-Map-App möchte einen Datensatz in der Datenbank speichern, wodurch die maximale Speichergröße überschritten wird.
Dazu wird dieser Datensatz an den Datenbankzugriffspunkt gesendet. (1) Danach greift dieser Zugriffspunkt auf die Datenbank zu und erhält ein Datenbankobjekt auf dem dieser Zugriffspunkt arbeitet. (2)+(3) Über dieses wird nun der Datensatz in der Datenbank eingefügt und gespeichert. (4)
In der Folge prüft die Datenbank, ob nun die maximale Speichergröße der Datenbank überschritten wurde. (5) Wenn dies der Fall ist, wird der älteste Datensatz gelöscht. (6)
Dabei werden die Schritte (5) und (6) in einer Schleife solange wiederholt, bis die aktuelle Speichergröße wieder unter dem Maximum liegt.
Danach wird von der Datenbank über dem Datenbankzugriff (7) an die X-Map-App (8) die in der Datenbank gepeicherte "id" des Datensatzes zurückgegeben.
Optional kann danach noch die Datenbank geschlossen werden, falls kein weiter Zugriff benötigt wird. (9)-(10)

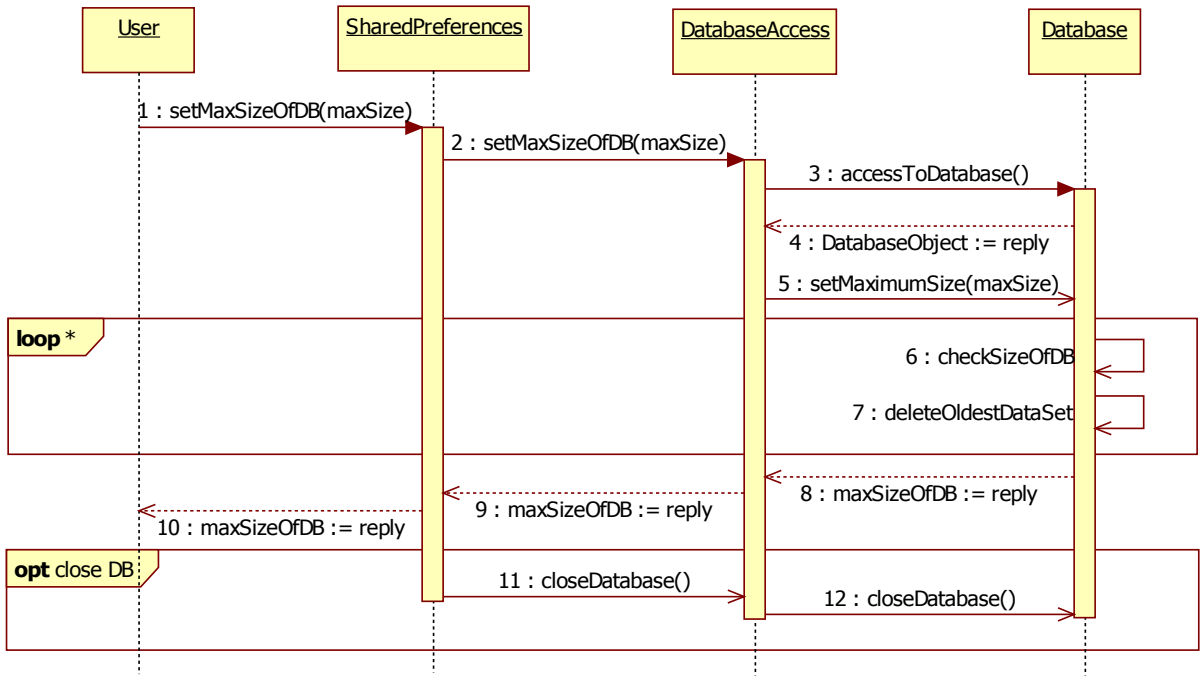

Abbildung 2.6: Sequenzdiagramm zum Löschen von Daten bei einer Einstellungsänderung

Der Benutzer stellt in den Einstellungen die maximale Größe der Datenbank ein. Die neue maximale Speichergröße ist nun kleiner als die aktuelle Datenbankgröße. (1)
Nun wird die X-Map-App über das SharedPreferences-Interface an den Datenbankzugriffspunkt die maximale Speichergröße weiterleiten (2), wodurch dieser Zugriffspunkt den Zugriff auf die Datenbank anfordert und ein Datenbankobjekt für die Weiterverarbeitung erhält. (3)+(4) Über dieses wird die Datenbank Speichergröße neu gesetzt. (5)
Hierfür wird in einer Schleife geprüft, ob die aktuelle Größe der Datenbank noch größer als die maximal zulässige Größe ist. (6) Ist dies der Fall, wird der älteste Datensatz gelöscht. (7) Die Schleife über (6) und (7) wird verlassen, sobald die aktuelle Speichergröße kleiner oder gleich der maximal eingestellten ist. Danach wird dem Benutzer die neu gesetzte maximale Datenbankgröße zurückgegeben. (8)-(10)
Optional kann danach noch die Datenbank geschlossen werden, falls kein weiter Zugriff benötigt wird. (11)+(12)

2.1.5 Analyse von Funktionalität $\langle F50 \rangle$: Daten lokal speichern

Gemessene Daten werden in einer Datenbank gespeichert.

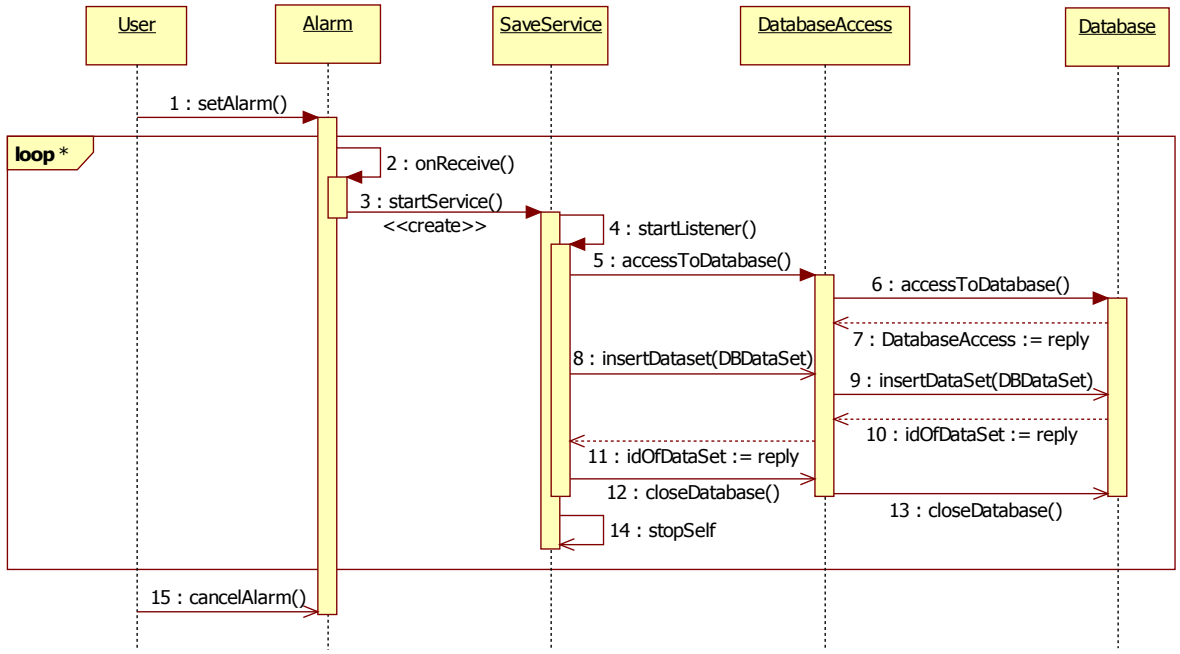

Abbildung 2.7: Sequenzdiagramm zum Lokalen speichern der Daten

Der Benutzer startet einen Alarm, im Hintergrund (1). Der Alarm schreibt sich dadurch ins Android-System und wird damit auch gestartet, wenn die App nicht aktiv ist oder das Mobilgerät sich im Standby-Modus befindet. Dieser Alarm ruft sich in bestimmten Zeitabständen selbst als Receiver auf (2). Dieser hat nur eine sehr kurze Lebensdauer, da er für sehr kurze Aktionen vorgesehen ist und vom Android-System selbst kurz nach Aufruf beendet wird. Deshalb startet dieser Receiver einen Service, bzw. kreiert diesen, falls er noch nicht existiert. (3)
Der Service startet den Listener (4), der für das Auslesen des Handynetzes und der GPS-Daten fungiert. Dabei wird über den Datenbankzugriff ein Zugriff auf die Datenbank ermöglicht. (5)+(6) Der Datenbankzugriff erhält hierfür ein Datenbankobjekt. (7) Die gesamte Kommunikation zwischen dem Service und der Datenbank erfolgt nun über die Schnittstelle des Datenbankzugriffs. Wenn der Listener Daten gesammelt hat, werden diese in der Datenbank gespeichert (8)+(9) und wird in der Folge durch die "id" des gespeicherten Datensatzes, über den Erfolg des Speicherns informiert. (10)+(11)
Zwischen den Schritten (9) und (10) kann optional, die in Abbildung 2.5 beschriebene Schleife mit den dortigen Schritten (5)+(6), zum Löschen von Datensätzen, eintreten. (Wird hier nicht näher spezifiziert.)
Danach hat der Service seine Arbeit beendet, schließt die Datenbank (12)+(13) und stoppt sich selbst. (14)

Dabei werden die Schritte (2)-(14) immer periodisch in einer Schleife wiederholt bis der Benutzer den Alarm beendet. (15)

2.1.6 Analyse von Funktionalität $\langle F60 \rangle$: Einstellungen für lokal zu speichernde Daten

Einstellungen für lokal zu speichernde Daten werden vom Nutzer festgelegt.

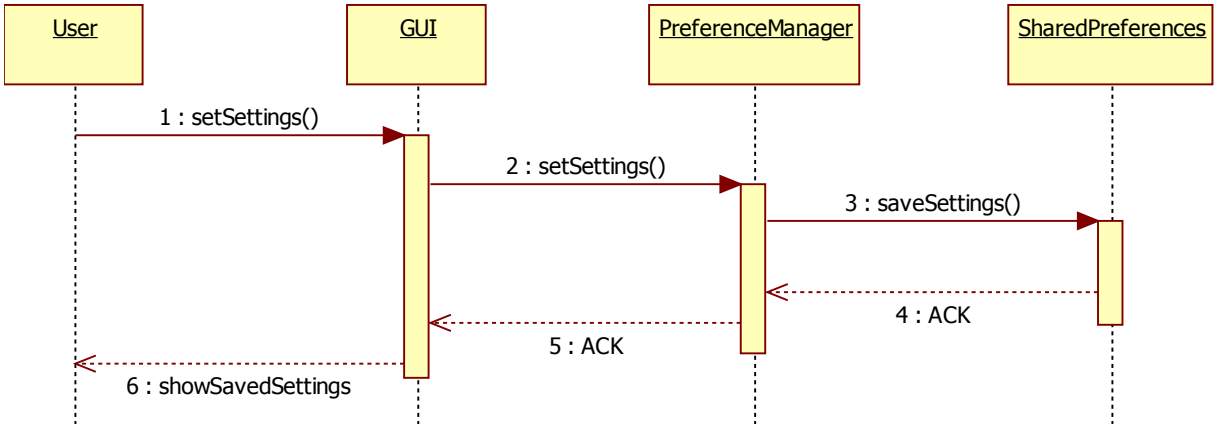

Abbildung 2.8: Sequenzdiagramm für Einstellungen lokal zu speichernder Daten

Der Nutzer wählt zu speichernde Parameter auf der App aus:

- Messdaten, die lokal gespeichert und an den WebService übertragen werden (inkl. Datenschutzeinstellungen bzgl. Gerätebezug (ehemalig $\langle F70 \rangle$)
- Die Häufigkeit der Messdatenübertragung
- Maximale Speichergröße der Datenbank

Sind die gewünschten Parameter ausgewählt, werden sie von der GUI an den PreferenceManager übergeben. Dies ist eine von Android zur Verfügung gestellte Klasse zur Verwaltung von Nutzereinstellungen. Der PreferenceManager speichert die Einstellungen mit Hilfe des Interfaces SharedPreferences in einer XML-Datei. SharedPreferences stellt einen Mechanismus für Speicherung von und Zugriff auf Einstellungen bereit.

2.1.7 Analyse von Funktionalität $\langle F80 \rangle$: Registrierung eines Benutzers

Ein Benutzer gibt auf dem Mobilgerät seine E-Mail-Adresse und ein Passwort ein. Diese Daten werden dann an den Server übermittelt, welcher die Daten mit der bestehenden Benutzdatenbank abgleicht und entsprechend die Daten einträgt, oder nicht.

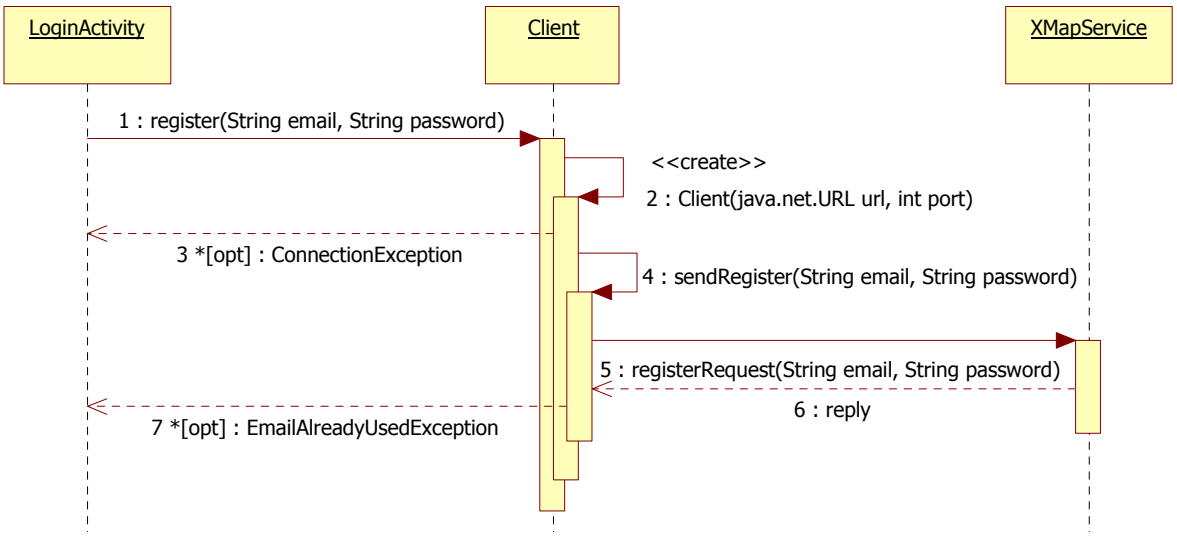

Abbildung 2.9: Sequenzdiagramm zum Registrieren eines Benutzers

Die statische Methode register wird mit Email-Adresse und Passwort als Parameter aufgerufen (1). Danach wird der Client-Konstruktor gestartet (2). Dabei kann wie beim Login eine ConnectionsException ausgelöst (3) werden, wenn es Verbindungsprobleme geben sollte. Intern wird die private Methode sendRegister aktiviert (4), welche dann ein RegisterRequest (5) an den Server sendet. Das Ergebnis des Servers (6) kann entweder eine Erfolgsnachricht sein, oder ein Fehler. Im letzteren Fall wird eine EmailAlreadyUsedException (7) erstellt. Entstehen keine Exceptions war die Registrierung erfolgreich.

2.2 Server

Die folgenden Funktionalitäten $\langle F90 \rangle$ bis $\langle F130 \rangle$ sind dem Server zugeordnet.

2.2.1 Analyse von Funktionalität $\langle F90 \rangle$: Anmeldung

Ein Benutzer gibt seine Anmeldedaten auf dem Mobilgerät ein. Diese Daten werden an den Server übermittelt, mit der Benutzerdatenbank abgeglichen, woraufhin entweder die Anmeldung zurückgewiesen oder das Mobilgerät am Server bekanntgemacht wird.

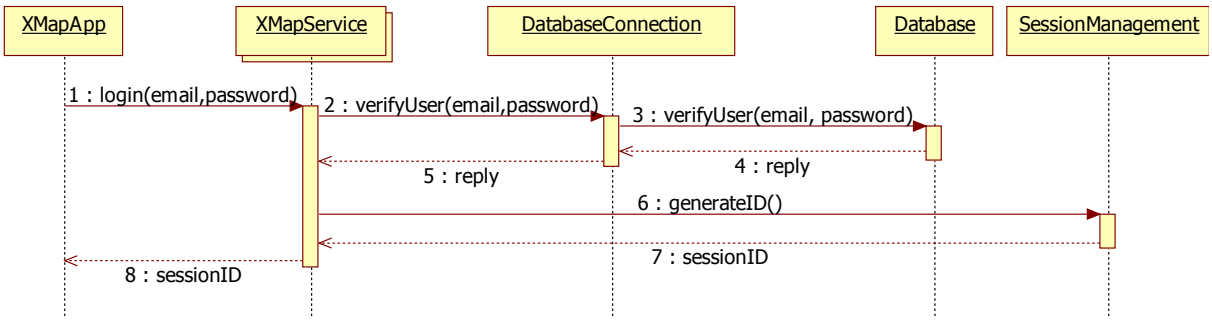

Abbildung 2.10: Sequenzdiagramm zum Anmeldevorgang

Die App ruft über einen Fernaufruf auf dem (mutlithreaded) WebService die Login-Prozedur mit E-Mail-Adresse und Passwort als Parameter auf. (1) Der Service fragt nun über die Datenbankverbindungsschicht bei der Datenbank an, ob eingegebene Werte korrekt sind, woraufhin eine entsprechende Antwort erfolgt. (2)-(5) Ist der Login korrekt, wird mithilfe des SessionManagements eine SessionID generiert und der App als Antwort übergeben. (6)-(8) Ist der Login inkorrekt wird keine ID generiert und eine negative SessionID zurückgeliefert.

2.2.2 Analyse von Funktionalität $\langle F100 \rangle$: Registrierung eines neuen Mobilgeräts

Ein Benutzer meldet sich mit einem bisher unbekannten Mobilgerät an. Das Mobilgerät wird, sofern im Datenschutz erlaubt, dem Nutzer zugeordnet und kann von nun an Daten übermitteln.

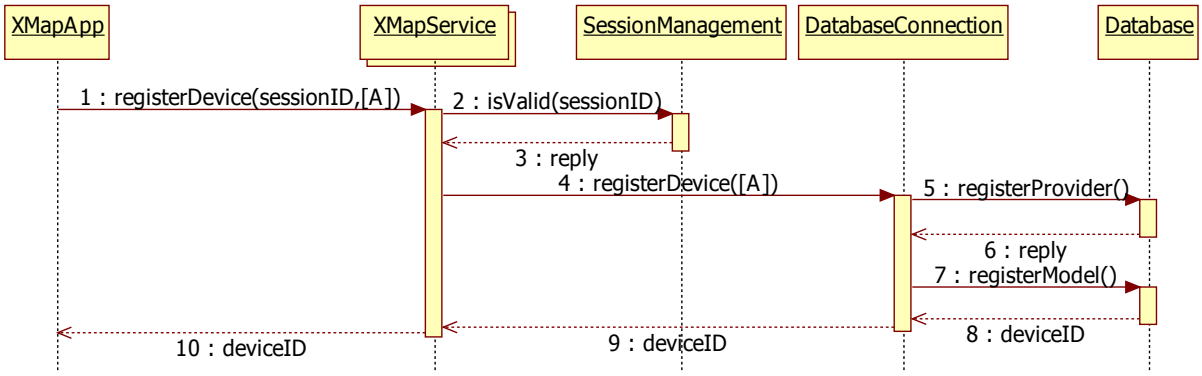

Abbildung 2.11: Sequenzdiagramm zum Geräte-Registrierungsvorgang

[A] := manufacturer, device, phoneType, networkOperator, networkOperatorName, countryCode

Die App ruft über einen Fernaufruf auf dem (multithreaded) WebService die Geräte-Registrierungsprozedur mit ihrer aktuellen sessionID sowie den Parametern aus [A] auf. (1)
Es wird überprüft, ob die SessionID gültig ist. (2)+(3)
Im Erfolgsfall wird die DatabaseConnection mit der Registrierung des Gerätes betraut. (4)-(8)
Die zurückgegebene deviceID wird dem Gerät dann übergeben. (9)+(10)
Das Gerät ist nun registriert.

2.2.3 Analyse von Funktionalität $\langle F110 \rangle$: Speicherung der Daten

Ein neu erhaltener Messdatensatz wird bei gültiger Session-ID in der Datenbank gespeichert.

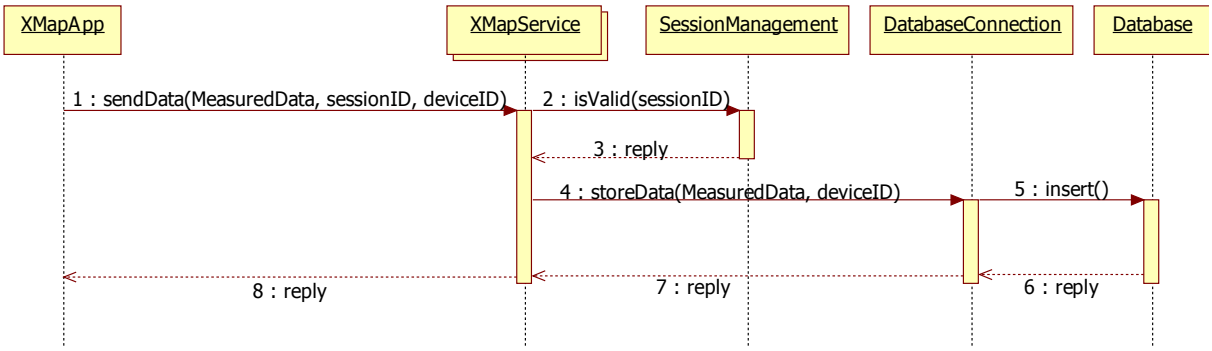

Abbildung 2.12: Sequenzdiagramm zur Speicherung der Daten

Die App sendet einen neuen Messdatensatz mit der aktuellen SessionID und der DeviceID an den WebService. Dieser überprüft, ob die SessionID gültig ist und speichert wenn ja, die neuen Messdaten mit der DeviceID in der Datenbank.

2.2.4 Analyse von Funktionalität $\langle F120 \rangle$: Auslesen der Daten

Zu Auswertungszwecken werden über den WebService Datensätze aus der Datenbank ausgelesen.

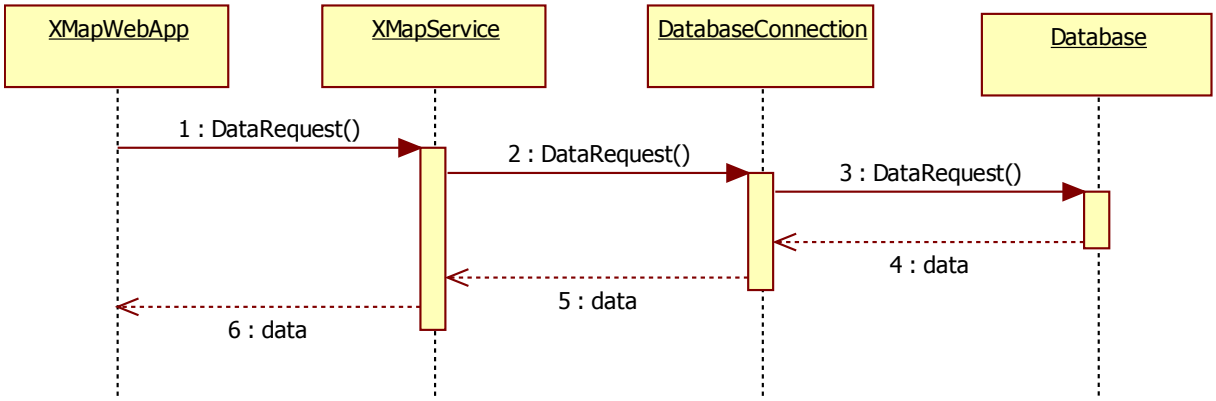

Abbildung 2.13: Sequenzdiagramm zum Auslesen der Daten

Die Web-Applikation sendet eine Datenanfrage über den WebService an die Datenbank. Diese gibt die benötigten Daten auf selbem Wege zurück an die WebApp.

2.2.5 Analyse von Funktionalität $\langle F130 \rangle$: Visualisierung der Daten

Die Web-Applikation benutzt den WebService um Daten aus der Datenbank zu erhalten, welche daraufhin visualisiert werden können.

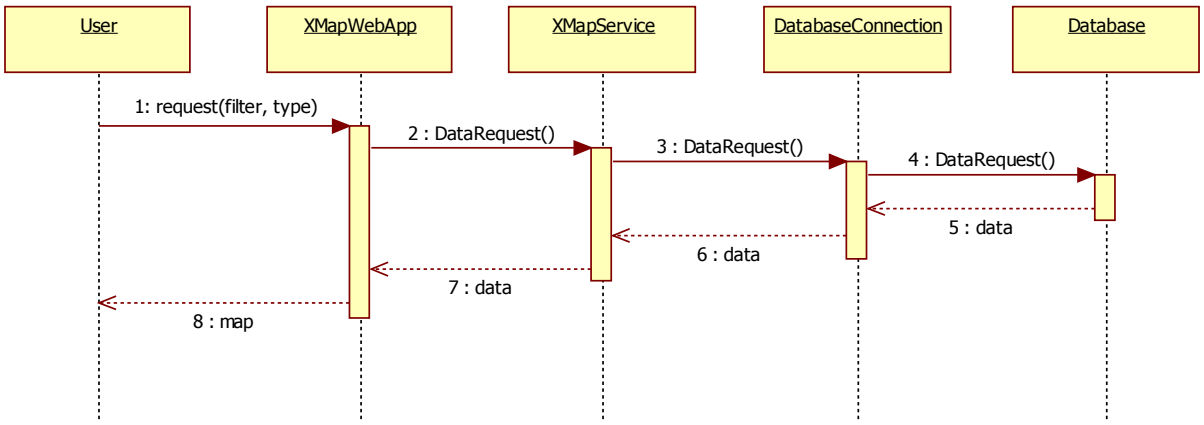

Abbildung 2.14: Sequenzdiagramm zum Visualisieren der Daten

Die Web-Applikation erhält Informationen vom Benutzer welche Daten er auf welche Art visualisiert bekommen möchte und sendet dementsprechend eine Datenanfrage raus. Sobald diese Daten bei der Web-Applikation eingetroffen sind werden diese dem Benutzer angezeigt.

2.3 Nicht-Funktionale Anforderungen

Dieser Abschnitt beschreibt die Nicht-Funktionale Anforderungen in Sequenzdiagrammen. Dabei werden $\langle Q10 \rangle$, $\langle Q20 \rangle$ und $\langle Q30 \rangle$ sowie $\langle Q40 \rangle$ und $\langle Q50 \rangle$ in jeweils einem Sequenzdiagramm zusammengefasst. $\langle Q60 \rangle$ besitzt ein eigenständiges Sequenzdiagramm.

2.3.1 Analyse der Nicht-Funktionalen Anforderungen $\langle Q10 \rangle$, $\langle Q20 \rangle$ und $\langle Q30 \rangle$

Der Teilabschnitt beschreibt mittels Sequenzdiagramm in Abbildung 2.15, wie der Benutzer auf eine zu lange Antwortzeit hingewiesen wird. Die hier benutzten Anforderungen sind:

- $\langle Q10 \rangle$ Die Antwortzeit darf für jegliche Benutzeranfrage nicht mehr als drei Sekunden betragen.
- $\langle Q20 \rangle$ Der Benutzer wird informiert, sobald die Antwortzeit länger als drei Sekunden ist.
- $\langle Q30 \rangle$ Eine Fortschrittsanzeige wird bei einer Verzögerung zur Verfügung gestellt.

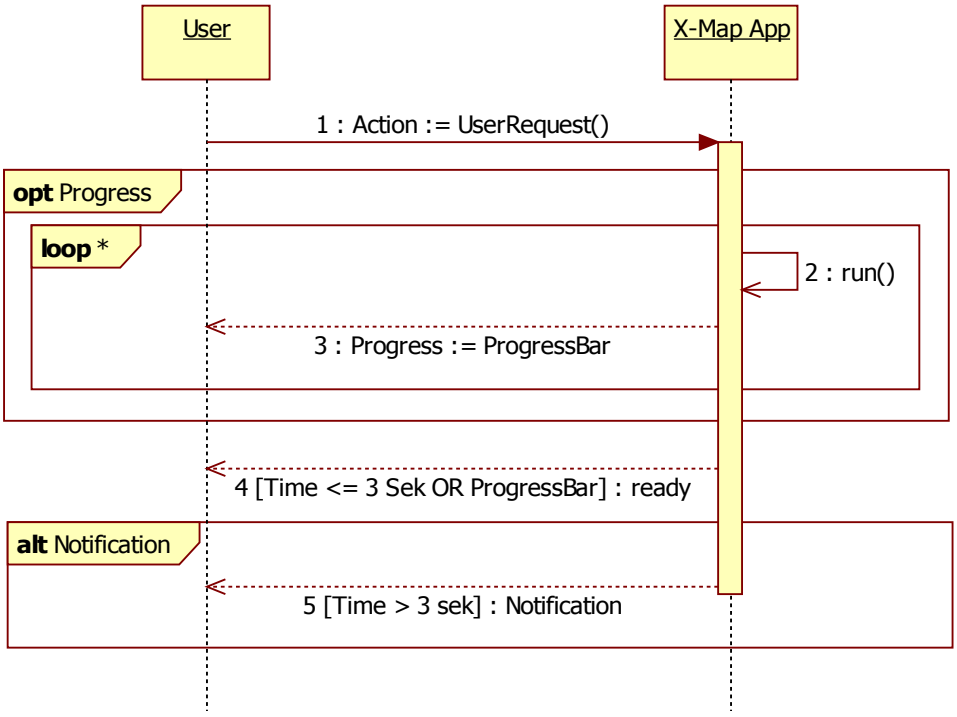

Abbildung 2.15: Sequenzdiagramm zu $\langle Q10 \rangle$, $\langle Q20 \rangle$ und $\langle Q30 \rangle$

Der Benutzer startet eine Benutzeranfrage an die X-Map App (1), welche eine längere Berechnung ausführen muss. Dazu kann die App dem Benutzer optional während sie arbeitet (2) eine Fortschrittsanzeige ausgeben (3), wenn schon bei Beginn der Berechnung bekannt ist, dass folgende Berechnung länger dauern kann. Welche Berechnung eine Fortschrittsanzeige, von Beginn

an erhält, wird von den Entwicklern durch gezieltes Testen bestimmt.
Am Ende der Berechnung arbeitet die App ohne weitere Information an den Benutzer weiter. Dazu hat die App entweder die Berechnung innerhalb von 3 Sekunden vollendet oder der Benutzer wurde mittels Fortschrittsanzeige über den Berechnungsfortschritt auch über die 3 Sekunden hinaus informiert. (4)
Wenn kein Fortschritt angezeigt wird und die Berechnung länger als 3 Sekunden benötigt, wird dem Benutzer alternativ nach 3 Sekunden eine Benachrichtigung angezeigt, dass die Berechnung noch nicht vollendet ist.

2.3.2 Analyse von den Nicht-Funktionalen Anforderung $\langle Q40 \rangle$ und $\langle Q50 \rangle$

Der Teilabschnitt beschreibt die Fehlerbehandlung durch den Benutzer. Dazu soll der Benutzer bei Fehlern möglichst nicht eingreifen müssen. Dies wird im Sequenzdiagramm auf Abbildung 2.16 dargestellt. Die hier benötigten Nicht-Funktionalen Anforderungen sind:

- $\langle Q40 \rangle$ Bei Aussetzen von Datenübertragung, Positionsbestimmung oder Messungen ist keine Interaktion des Benutzers nötig.
- $\langle Q50 \rangle$ Fehler bei Hintergrundfunktionen werden dem Anwender durch ein Fehlerprotokoll zugänglich gemacht.

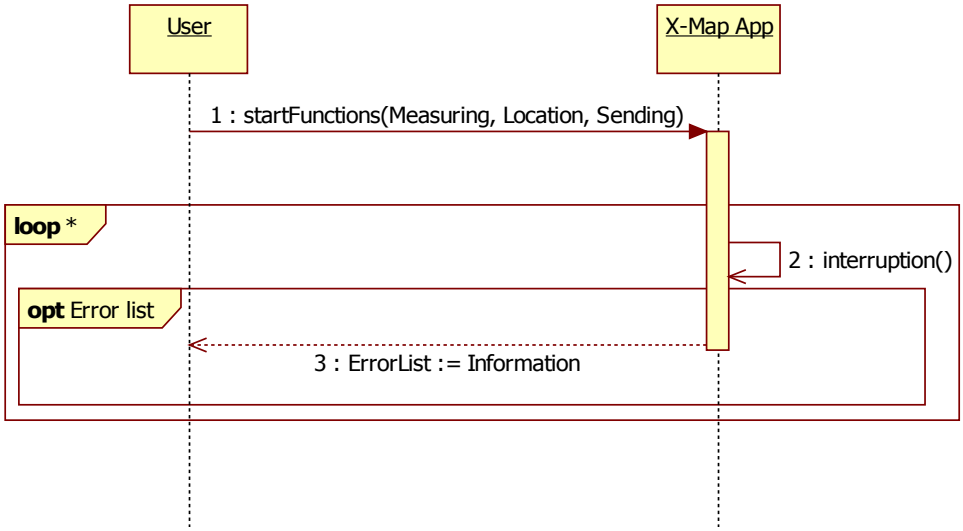

Abbildung 2.16: Sequenzdiagramm zu $\langle Q40 \rangle$ und $\langle Q50 \rangle$

Der Benutzer startet eine oder mehrere der Funktionen, die für die Datenübertragung, Positionsbestimmung und Messungen vorhanden sind. (1)
Während die Funktionen ihre Aufgaben erfüllen, kann es zu Aussetzern verschiedener Formen kommen. (2) Da diese Aussetzer mehrfach auftreten können, wird dieses in einer loop-Schleife dargestellt.
Der Benutzer kann dabei optional ein Fehlerprotokoll einsehen, um sich über mögliche Aussetzer und Fehler zu informieren. (3) Das Fehlerprotokoll wird dabei als optional gekennzeichnet, da dieses ein Kann-Kriterium $\langle RC1 \rangle$ ist und somit noch nicht mit in die bisherige Planung einbezogen wurde.

2.3.3 Analyse von der Nicht-Funktionalen Anforderung $\langle Q60 \rangle$

In diesem Teilabschnitt wird die Spracheinstellung der App in einem Sequenzdiagramm in Abbildung 2.17 beschrieben. Die hier beschriebene Nicht-Funktionale Anforderung ist:

- $\langle Q60 \rangle$ Die Verkehrssprache ist Englisch und Deutsch.

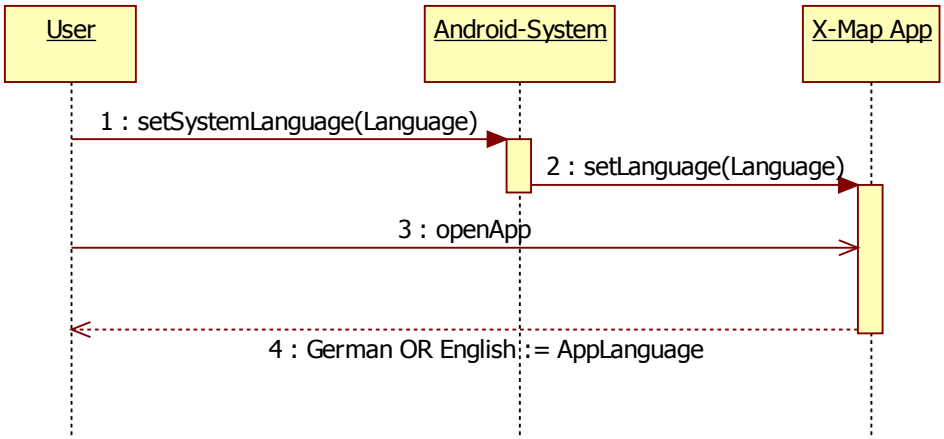

Abbildung 2.17: Sequenzdiagramm zu $\langle Q60 \rangle$

Die Sprachauswahl der App, wird über die Android-Systemsprache festgelegt. Demnach setzt der Benutzer die Sprache des Mobilgerätes auf eine von ihn gewünschte Sprache. (1)
Dadurch passt die App sich der eingestellten System-Sprache an, indem das Betriebssystem der App die eingestellte Sprache übermittelt. (2)
Danach kann der Benutzer die App öffnen (3), wodurch ihm die Sprache der App angezeigt wird. (4) Dazu wird bei deutscher Systemsprache, die App ebenfalls in Deutsch dargestellt. Bei allen anderen Systemsprachen, wird die App auf Englisch dargestellt.

3 Resultierende Softwarearchitektur

Durch eine vorhergehende Analysephase wird für die X-Map App sowohl client- als auch serverseitig eine 3-Schichten-Architektur verfolgt. Unterschieden wird dabei zwischen einer GUI- bzw. Service-Schicht, einer Fachkonzeptschicht und einer Datenhaltungsschicht.
Es folgt ein etwas genauerer Einblick in dieses System.

3.1 Komponentenspezifikation

Im Folgenden wird die Komponentenstruktur von X-Map näher beschrieben. Das zentrale Element dieses Kapitels ist Abbildung 3.1, welche einen Überblick über die Komponenten des Systems liefert.

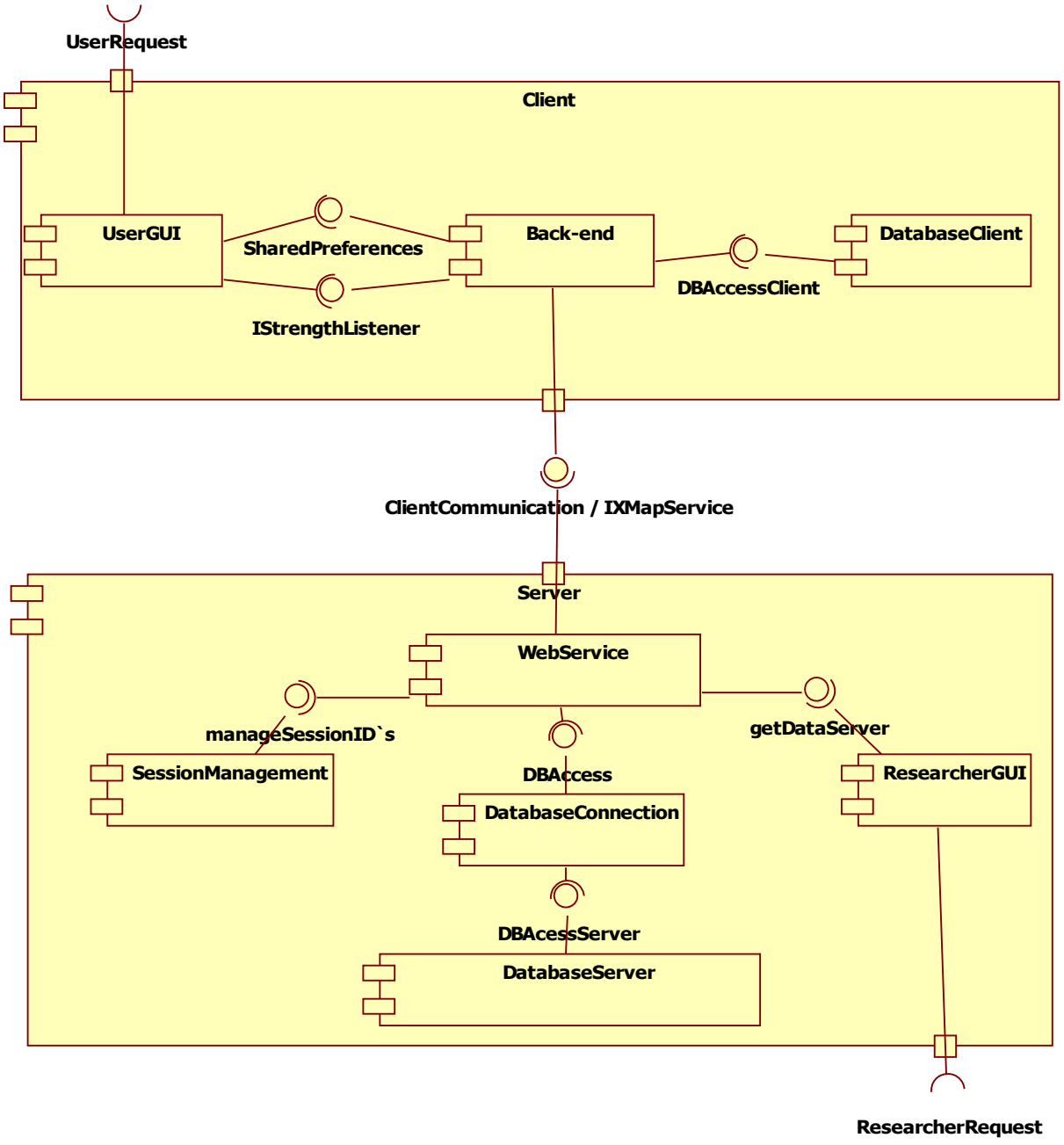

Abbildung 3.1: Komponentendiagramm

3.1.1 Client

In diesem Kapitel werden die Komponenten der Client-Oberkomponente besprochen:

Komponente $\langle C10 \rangle$: Client
Der Client beschreibt ein Mobilgerät. Der Client ermöglicht alle Operationen die notwendig sind. Dabei arbeitet der Client auf einer 3-Schichten-Architektur, um eine erweiterbare Architektur zu gewährleisten.
Der Client kommuniziert mit der Komponente Server ($\langle C20 \rangle$), um Daten an diesen zu übertragen.

Komponente $\langle C11 \rangle$: UserGUI
Die UserGUI dient zur direkten Kommunikation mit dem Nutzer. Die Auswertung und Anzeige der lokal gemessenen Daten kann durch eine Tabelle, einen Graphen oder eine Google Maps erfolgen. Bei der Google Maps wird die Empfangsqualität durch ein farbiges Overlay dargestellt.

Komponente $\langle C12 \rangle$: Backend
Das Backend ist eine zentrale Steuerungs- und Kontrolleinheit. Es übernimmt u.a. die Datenmessung, Speicherung sowie Kommunikation mit dem Server.

Komponente $\langle C13 \rangle$: DatabaseClient
Der DatabaseClient stellt die Datenbank des Clients dar. Es stellt auch Methoden zu Verfügung, um die Operationen auf der eigentlichen Datenbank zu vereinfachen.

3.1.2 Server

Dieses Kapitel enthält die Beschreibungen der Komponenten des Servers:

Komponente $\langle C20 \rangle$: Server
Der Server bietet die Basis für die Ausführung des WebServices und der WebApplikation.

Komponente $\langle C21 \rangle$: WebService
Der WebService stellt Methoden zum Registrieren und Anmelden von Usern, sowie zum Annehmen und Auslesen von Messdaten bereit.

Komponente $\langle C22 \rangle$: SessionManagement
Das SessionManagement ist ein statisches Objekt, das über alle Instanzen des WebService eine threadsichere Liste von SessionID's verwaltet. SessionID's werden bei der Anmeldung an Geräte übergeben und dienen dazu, User zu authentifizieren, ohne die Login-Daten mehrfach übertragen zu müssen.

Komponente $\langle C23 \rangle$: DatabaseConnection
Die DatabaseConnection stellt dem Webservice einfache Methoden zur Verfügung, in welchen z.T. komplexe Operationen auf der eigentlichen Datenbank gekapselt sind.

Komponente $\langle C24 \rangle$: DatabaseServer
DatabaseServer symbolisiert die Datenbank, welche User- und Messdaten verwaltet und bereitstellt.

Komponente $\langle C25 \rangle$: ResearcherGUI
Die Komponente ResearcherGUI stellt die Web-Applikation dar, welche zur Anzeige und Auswertung der gemessenen Daten dient.

3.2 Schnittstellenspezifikation

Die in Abbildung 3.1 abgebildeten Schnittstellen der einzelnen Komponenten und ihre zugehörigen Methoden werden in den folgenden Tabellen näher erläutert.

Schnittstelle $\langle I10 \rangle$: DBAccessClient

Operation	Beschreibung
accessToDatabase()	Wird von einer Klasse des Clients aufgerufen, die Zugriff auf die Datenbank benötigt. Die Datenbank erzeugt ein Datenbankobjekt, das für die Klasse in der Schnittstelle bereitgestellt wird.
insertDataset()	Es soll ein Datensatz in der Datenbank gespeichert werden. Dazu wird über die Schnittstelle der gewünschte Datensatz an die Datenbank weitergeleitet, welche diesen nun in sich aufnimmt.
getCursor()	Es müssen Datensätze aus der Datenbank ausgelesen werden. Die Datenbank liest die gewünschten Datensätze aus und speichert diese in einem Cursor. Dieser wird dann der Schnittstelle übergeben und an die Backend-Schicht weitergeleitet. Diese Methode gibt es in verschiedenen Ausführungen, die noch nicht explizit feststehen. Der einzige Unterschied ist dabei der Inhalt des Cursors, der je nach Einsatz des Cursors anders ist.
closeDatabase()	Wird von einer Klasse des Clients aufgerufen. Die Klasse benötigt den Zugriff auf die Datenbank und muss explizit diese für sich schließen lassen.

Schnittstelle $\langle I20 \rangle$: IStrengthListener

handleStrengthChanged()	GoogleMapsActivity, XYChartActivity sowie DBTableActivity sind Listener von der Singleton Back-end Klasse StrengthListener. Dafür implementieren sie das Interface IStrengthListener mit der Methode handleStrengthChanged(). Sobald neue Nutzerdaten gemessen und in der Datenbank gespeichert werden, wird im Back-end über alle, in einer Liste gesammelten, Listeners iteriert und handleStrengthChanged() aufgerufen. In HandleStrengthChanged() wird das Verhalten der einzelnen Visualisierungstypen bei neuen Daten implementiert. Auf diese Weise kann das Back-end ohne direkte Abhängigkeit das Front-end (GUI) notifizieren.

Schnittstelle $\langle I30 \rangle$: ClientCommunication

getInstance()	Gibt ein übertragungsbereites Objekt der Singleton-Klasse Client zurück.
sendData()	Sendet ein Array aus DBSet Objekten an den Server.
registerUser()	Registriert einen Nutzer am WebService
registerDevice()	Registriert ein Gerät am WebSservice

Schnittstelle $\langle I40 \rangle$: SharedPreferences

Operation	Beschreibung
getDefaultSharedPreferences()	Wird von einer Klasse des Clients aufgerufen, die Lesezugriff auf die gespeicherten Einstellungen benötigt.
edit()	Wird von einer Klasse des Clients aufgerufen, die Schreibzugriff auf die gespeicherten Einstellungen benötigt.
commit()	Wird aufgerufen, um veränderte Wertepaare zu speichern.
onSharedPreferenceChanged()	Listener eines Wertepaares. Wird von einer Klasse des Clients implementiert, die bei Änderung des Wertepaares benachrichtigt werden muss.

Schnittstelle $\langle I50 \rangle$: IXMapService

Operation	Beschreibung
sendData()	Wird vom Mobilgerät aufgerufen, um Daten zur Speicherung an die Datenbank zu übertragen.
endSession()	Mobilgerät übergibt dem WebService seine SessionID, welche daraufhin durch das SessionManagement entfernt wird.
registerUser()	Mobilgerät überträgt E-Mail-Adresse, Passwort und Telefon-Spezifika an den WebService, sodass ein neuer User erstellt und das Gerät der Gerätedatenbank hinzugefügt werden kann.
login()	Mobilgerät überträgt Login-Daten des Nutzers zur Authentifizierung an den WebService, woraufhin eine SessionID generiert und zurückgegeben wird.

Hinweis: Die Interfaces ClientCommunication $\langle I30 \rangle$ und IX-MapService $\langle I50 \rangle$ sind zwei Sichtweisen auf die gleiche Verbindung. $\langle I30 \rangle$ ist hierbei die Sicht des Clients, $\langle I50 \rangle$ die des Servers.

Schnittstelle $\langle I60 \rangle$: manageSessionID's

Operation	Beschreibung
isValidID()	Sucht unter den SessionIDs nach der angefragten SessionID und überprüft gegebenenfalls den zugehörigen Zeitstempel. Gibt einen Boolean-Wert mit der Antwort zurück.
blockingDeleteID() und nonBlockingDeleteID()	Beide Methoden löschen die übergebene SessionID. Die blockierende Methode kehrt hierbei erst zurück, wenn die Löschung erfolgt ist. Die nicht-blockierende kehrt sofort zurück, ungeachtet ob das Löschen bereits erfolgt ist.
generateID()	Generiert eine neue SessionID, trägt sie in die entsprechende Datenstruktur ein und gibt sie daraufhin zurück.

Schnittstelle $\langle I70 \rangle$: DBAccess

Operation	Beschreibung
insertData()	Erhält einen Messdatensatz und speichert diesen in der Datenbank. Gibt zurück, ob das Speichern geklappt hat oder nicht.
userInDB()	Überprüft, ob sich die übergebenen Benutzerdaten, insbesondere die E-Mail-Adresse, schon in der Userdatenbank befindet.
storeNewUser()	Speichert die übergebenen Benutzerdaten mit einer User-ID in der Datenbank.
getData()	Liest die gewünschten Daten aus der Datenbank aus

Schnittstelle $\langle I80 \rangle$: DBAccessServer

Es werden parametisierte Anfragen verwendet, um mit SQL-Statements auf die Daten, die in der Datenbank gespeichert sind, zuzugreifen.

Schnittstelle $\langle I90 \rangle$: getDataServer

Die Methoden zur Datenabfrage für die Web-Applikation (im Diagramm „ResearcherGUI“) konnten noch nicht entworfen und implementiert werden. Angegeben sind daher nur die aktuell wahrscheinlichen Methoden:

Operation	Beschreibung
registerResearcher()	Methode zur Registrierung eines neuen Benutzers mit Forschungszugriff.
loginResearcher()	Methode zum Login eines bestehenden Benutzers mit Forschungszugriff.
getFilters()	Gibt die aktuell zur Verfügung stehenden Möglichkeiten zum Filtern von Messergebnissen zurück.
getOverlayData() und getTableData()	Liefert jeweils angepasste Datensätze für die Ausgabe über ein Kartenoverlay bzw. über eine Tabellendarstellung.

3.3 Protokolle für die Benutzung der Komponenten

In diesem Abschnitt wird die korrekte Verwendung der einzelnen Komponenten anhand von Statecharts dargestellt. Die Komponenten die wiederverwendet werden, werden in den dazugehörigen Teilabschnitt beschrieben.

3.3.1 Komponente $\langle C10 \rangle$: Client

In diesem Teilabschnitt wird die korrekte Verwendung des Clients beschrieben.

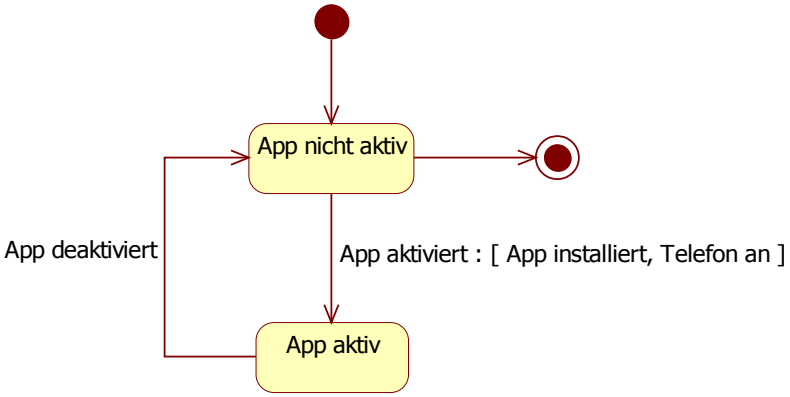

Abbildung 3.2: StateChart zur Komponente Client $\langle C10 \rangle$

Die Komponente Client besitzt, abgesehen vom Start- und Finalzustand, nur zwei Zustände. Solange die App nicht aktiv ist, passiert nichts in der Komponente. Um die App aktivieren zu können, muss sie auf dem Mobilgerät installiert und das Mobilgerät eingeschaltet sein.
Aktiv bedeutet, dass entweder die App im Vordergrund läuft, also das man sich in einem Zustand der UserGUI $\langle C11 \rangle$ befindet, oder die App Messungen und Übertragungen im Hintergrund durchführt, sich also kein Zustand der UserGUI $\langle C10 \rangle$ zuordnen lässt. Wärend letzterem Zustand kann sich das Smartphone selbst auch im Standby-Modus befinden.
Im aktiven Zustand agieren die Komponenten $\langle C12 \rangle$, $\langle C13 \rangle$ und ggf. $\langle C10 \rangle$ im Zusammenspiel miteinander und arbeiten ihre jeweiligen Abläufe ab.
Wenn man die App deaktiviert, kann sie manuell reaktiviert werden, oder aber sie aktiviert sich selbstständig, um Messungen und Übertragungen durchzuführen.

3.3.2 Komponente $\langle C11 \rangle$: UserGUI

Dieser Teilabschnitt beschreibt die korrekte Verwendung der UserGUI.

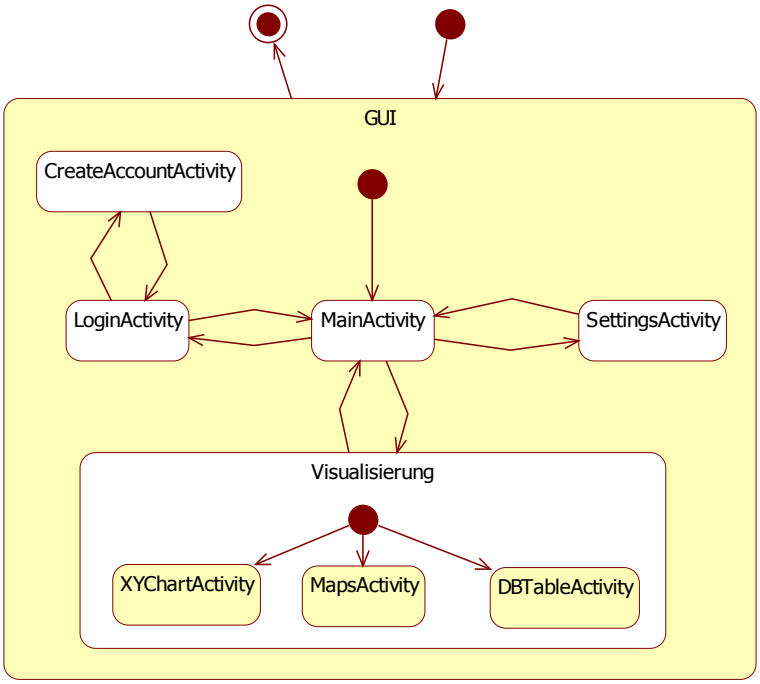

Abbildung 3.3: StateChart zur Komponente UserGUI $\langle C11 \rangle$

Die Komponente UserGUI dient zur Kommunikation mit dem Nutzer. Die MainActivitiy ist der Startbildschirm beim Öffnen der X-Map App. Durch Auswahloptionen kann zum einen eine Anmeldung (LoginActivity) erfolgen. Falls der Nutzer noch kein Konto eingerichtet hat, kann er sich registrieren (CreatAccountActivity). Zum anderen können Einstellungen bzgl. der zu übertragenden Messwerte, der Häufigkeit der Übertragung, eine maximale Speichergröße der Datenbank sowie Datenschutzoptionen ausgewählt werden (SettingsActivity). Über einen Visualisierungsbutton kann der Nutzer seine Messdaten durch verschiedene Visualisierungstypen auswerten: Eine Tabelle (DBTableAcitivy), eine Google Maps (MapsActivity) oder einen XY-Graphen (XYChartActivity). Der Nutzer kann jederzeit zum Startbildschirm zurückkehren. Von jeder Activity aus kann der Benutzer die App und somit die GUI beenden.

Für eine Applikation mit einer ähnlichen Schichten-Architektur ist die UserGUI für eine Wiederverwendung geeignet.

3.3.3 Komponente $\langle C12 \rangle$: Back-end

Im Backend werden Messungen durchgeführt und Daten verwaltet. Das beinhaltet deren Speicherung und Übertragung. Das StateChart geht auch nur auf diese beiden Funktionen ein. Es gibt zwar noch weitere Funktionen und Berechnungen, die in dieser Komponente durchgeführt werden, jedoch sind diese zustandslos und werden deshalb nicht weiter betrachtet.

Backend Übersicht

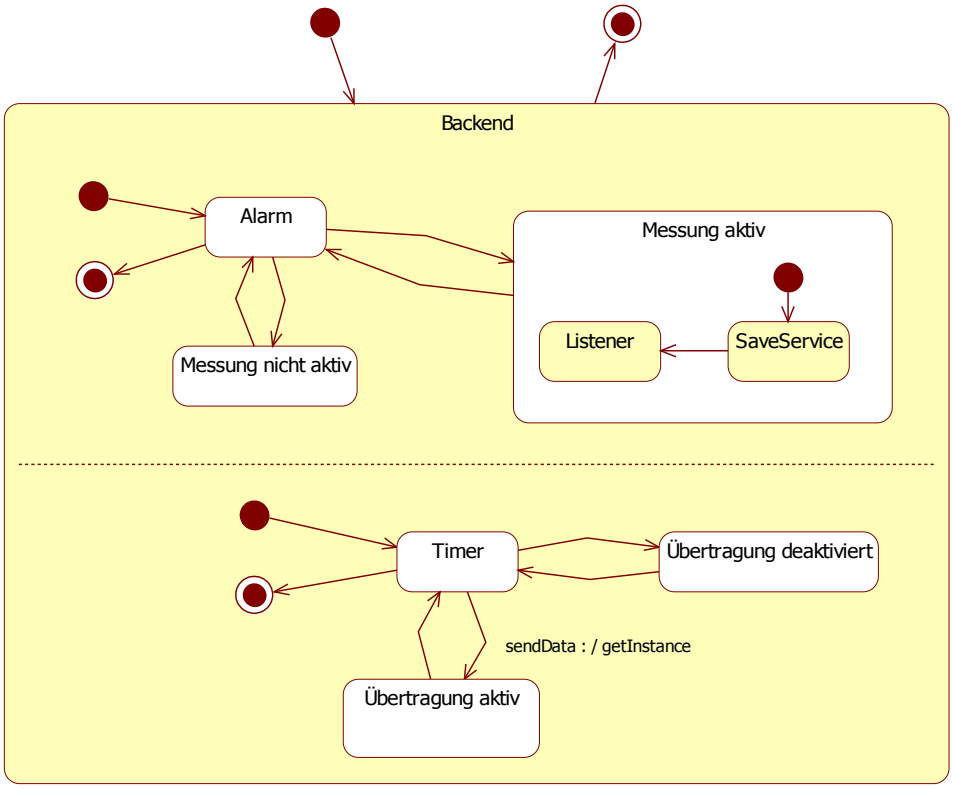

Abbildung 3.4: StateChart zur Komponente Back-end $\langle C12 \rangle$

Es gibt zwei Timer („Alarm“ und „Timer“), die jeweils einmal während des vom User vorgegeben Zeitintervalls eine Aktion ausführen. Diese beiden Timer laufen dabei parallel, was durch die gestrichelte Linie gekennzeichnet wird. „Alarm“ initiert eine Messung und deren anschließende Speicherung mithilfe der Persistenzschicht. „Timer“ startet die Übertragung an den Server durch Übergabe der Datensätze an die Client-Subkomponente. Dabei wird im Zustand “Übertragung aktiv“ das nachfolgende StateChart zum Client (Abbildung 3.5) gestartet.

Client

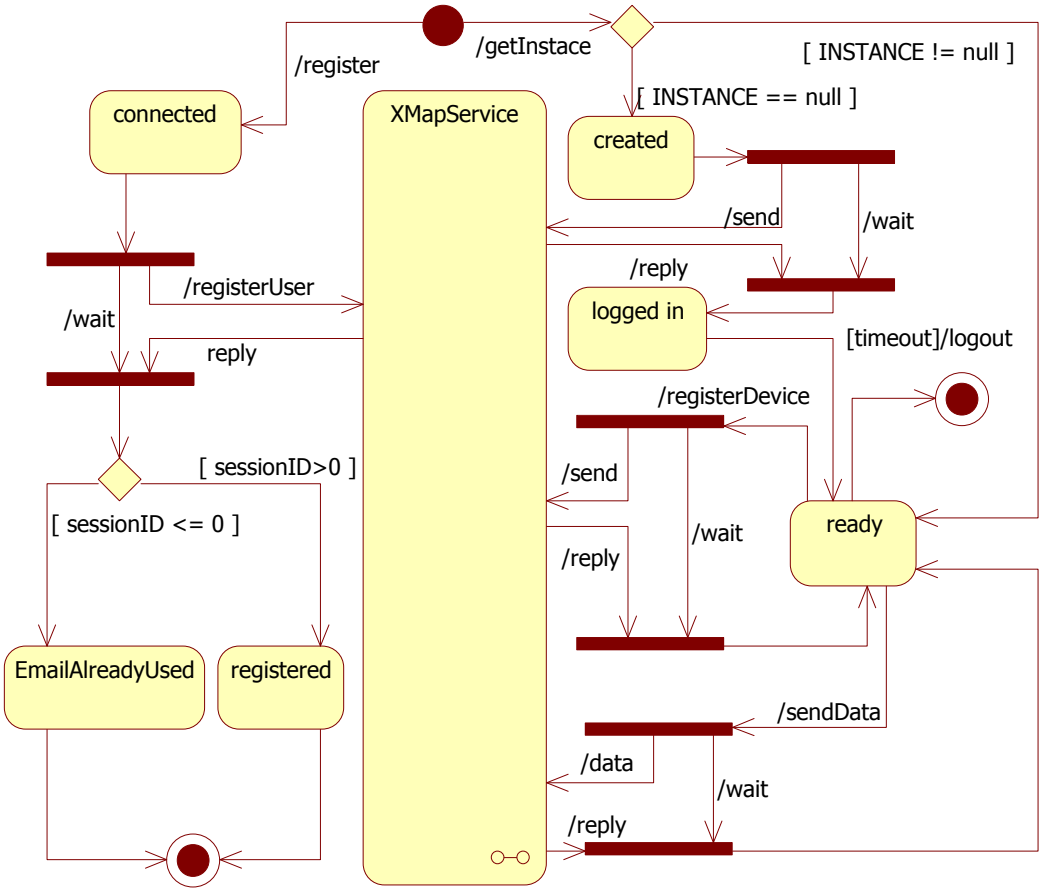

Abbildung 3.5: StateChart zur Kommunikation

Der Client kann entweder zur Registrierung oder zur regulären Kommunikation instanziiert werden. Nach der Registrierung wird die Verbindung sofort wieder beendet. Beim regulären Aufruf bleibt der Client aktiv, bis eine Timeoutzeit erreicht ist oder ein manuelles Logout ausgeführt wird. Nach der erfolgreichen Anmeldung kann das Gerät registriert oder Daten übertragen werden. Treten Fehler auf, werden diese zur Behandlung an die aufrufenden Entitäten weitergeleitet, welche die Fehler dann in einem entsprechenden Fehlerlog vermerken.

3.3.4 Komponente $\langle C13 \rangle$: DatabaseClient

Dieser Teilabschnitt beschreibt die Datenbank der Applikation, die für die Verwaltung der lokalen Daten vorhanden ist. Dabei wird diese Komponente mehrfach wiederverwendet, da die Applikation in periodischen Zeitabständen Daten abspeichert, welche auch mehrfach für die Visualisierung und Übertragung an den WebService ausgelesen werden. Dabei muss jede Methode und Klasse, die die Datenbank geöffnet hat, diese auch explizit wieder schließen.

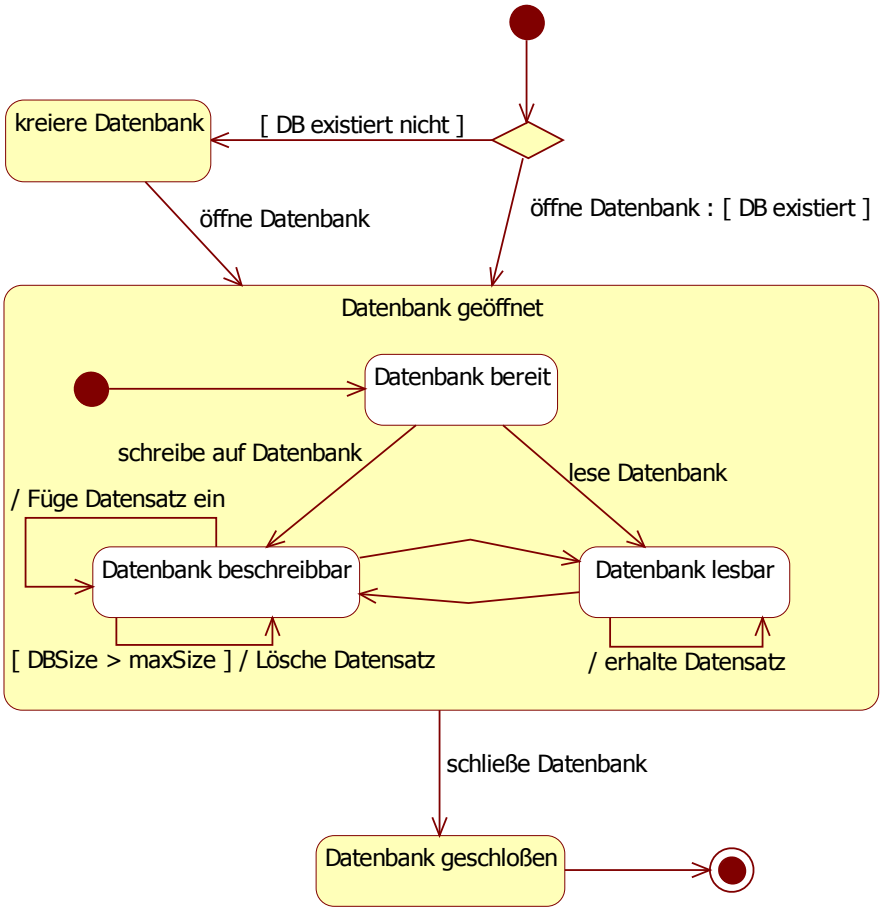

Abbildung 3.6: StateChart zur Komponente DatabaseClient $\langle C13 \rangle$

Die Komponente kreiert die Datenbank, falls diese noch nicht existiert, und öffnet sie. Wenn die Datenbank bereits existiert, wird stattdessen jene direkt geöffnet.
Während die Datenbank geöffnet ist, kann man in ihr schreiben und lesen. Schreibvorgänge sind das Einfügen und Löschen von Datensätze. Dabei werden Daten nur gelöscht, wenn die Datenbankgröße die maximal eingestellten Speichergröße überschreitet. Für das Lesen der Datenbank werden die jeweils benötigten Datensätze ausgelesen. Solange die Datenbank geöffnet ist, kann sie jederzeit geschlossen werden.
Nachdem die Datenbank einmal geöffnet wurde, muss sie zuerst wieder geschlossen werden, bevor sie beendet werden kann.

3.3.5 Server

Die Komponenten des Servers sind zustandslos, abgesehen vielleicht von „Busy“-„Idle“-Zuständen. Daher werden sie an dieser Stelle nicht weiter beschrieben.

4 Verteilungsentwurf

Deses Kapitel beschreibt die Verteilung der Komponenten mit einem Verteilungsdiagramm. Da das Produkt eine Client-Server-Architektur besitzt, wird die Verteilung auch dementsprechend sein. Dabei befindet sich die genaue Aufteilung noch sehr früh in der Planung, weshalb das Verteilungsdiagramm noch unvollständig ist. Es beschreibt nur die Verteilung auf die Knoten, die Komponenten der einzelnen Knoten fehlen.

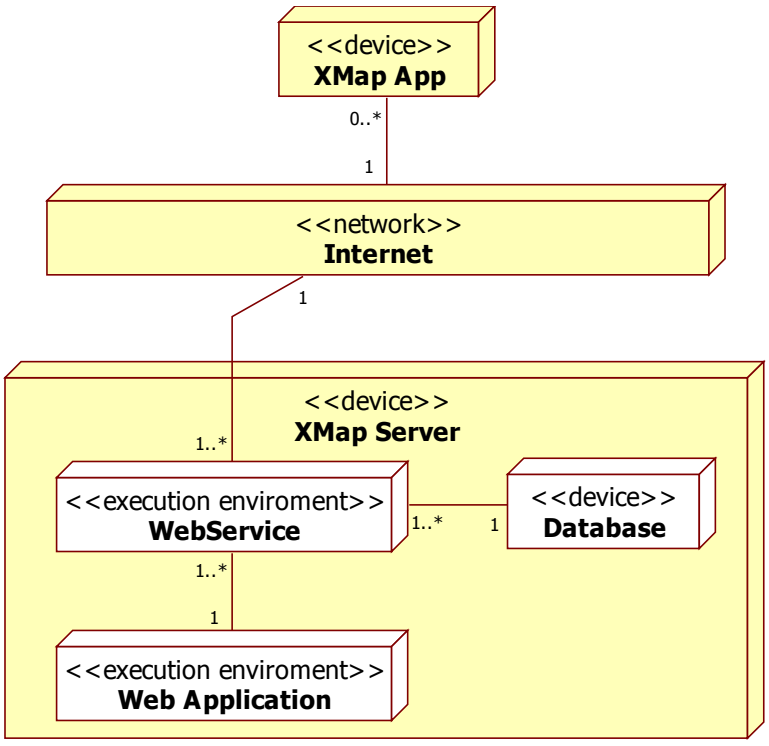

Abbildung 4.1: Verteilungsdiagramm

Die Abbildung 4.1 zeigt ein einfaches Verteilungsdiagramm. Auf dem X-Map Server laufen mehrere multithreaded WebServices, eine Web Applikation und eine Datenbank. Dabei ist die Datenbank ein Hardware Element (device) und die WebServices und die Web Applikation Laufzeitumgebungen (execution enviroment). Sowohl Datenbank als auch Web Applikation kommunizieren mit den WebServices.
Die WebServices des X-Map Servers sind mittels der physikalischen Verbindung, dem Internet, mit beliebig viele X-Map Apps verbunden.
Eine X-Map App beschreibt dabei ein Mobilgerät und somit auch ein Client.

5 Erfüllung der Kriterien

Folgendes Kapitel beschreibt den aktuellen Stand der Erfüllung der Kriterien aus dem Pflichtenheft.

5.1 Musskriterien

Es folgt die Beschreibung zu den Musskriterien des Pflichtenheftes.

5.1.1 App

Kriterien der Android-App:

- $\langle RM1 \rangle$ Aufzeichnung ortsabhängiger Daten des Mobilfunknetzes
  Dieses Kriterium wird erfüllt. Es werden dazu Methoden zum Auslesen der Telefon- und Verbindungsinformationen (aus den Klassen SignalStrength, CellLocation und Telephony-Manager), sowie zum Bestimmen der aktuellen Position (aus der Klasse Location), verwendet, welche bereits in Android enthalten sind.
- $\langle RM2 \rangle$ Einfache Graphendarstellung der vom eigenen Gerät gesammelten Daten
  Dieses Kriterium wird erfüllt. Es wird einfacher Graph mit x- und y-Achse erstellt, der die Signalstärke auf die y-Achse aufträgt und die Zeit auf die x-Achse. Dazu wird ein externes Paket "`AndroidPlot`" verwendet, da in Android keine Graphendarstellung bereitgestellt wird. Durch eine Vererbung der in "`AndroidPlot`" bereitgestellten Klasse "`XYPlot`" wurden weitere Methoden hinzugefügt, mit denen zusätzliches Zoomen und Skalieren des Graphen möglich wird.
- $\langle RM3 \rangle$ Möglichkeit, die gesammelten Daten an den zugehörigen WebService zu übermitteln
  Dieses Kriterium wird erfüllt. Um die gesammelten Daten zu übermitteln, werden in gewissen Zeitabständen mittels des externen Paketes "`kSOAP 2`" die aktuellsten Messdaten an den WebService übertragen.

- $\langle RM4 \rangle$ Option, die Übermittlungsfrequenz und den maximalen Speicherbedarf zu variieren

  Dieses Kriterium wird erfüllt. Dem Benutzer wird die Wahl gelassen, in welchen Abständen die Applikation Messdaten an den WebService übertragen soll. Dazu kann er bis jetzt zwischen 2 Stunde, 5 Stunden und 10 Stunden wählen. Optional kann er auch eine Übertragung sofort durchführen lassen, wenn der Benutzer es wünscht.
  Die Speichergröße ist veränderbar. Dabei wird die Maximalgröße der Datenbank eingestellt. Einstellbare Speichermöglichkeiten sind 10 MB, 50 MB und 100 MB. Hierdurch kann der Speicherbedarf an die Wünsche des Nutzers angepasst werden.

- $\langle RM5 \rangle$ Lokales Speichern der aufgezeichneten Messdaten
  Dieses Kriterium wird durch eine auf dem Mobilgerät eingerichtete Datenbank erfüllt. Dazu wird das in Android bereitgestellte "`SQLite`"-Datenbankverwaltungssystem verwendet. Durch diese wird eine einfache Verwaltung der Daten ermöglicht.

- $\langle RM6 \rangle$ Auswahl der zu messenden und lokal zu speichernden Parameter
  Dieses Kriterium wird erfüllt. Die vom Nutzer ausgewählten Einstellungen werden mittels des (bereits in Android enthaltenen) PreferenceManagers in einer XML-Datei permanent gespeichert. Die gespeicherten Einstellungen sind mittels des Interfaces SharedPreferences innerhalb jeder Klasse der App sichtbar.

- $\langle RM7 \rangle$ Selektion verschiedener Datenschutzoptionen
  Aufgrund von Änderungen ist dieses Musskriterium nun mit $\langle RM6 \rangle$ verbunden. Dieses Kriterium gibt es nun nicht mehr alleinstehend, wird aber in $\langle RM6 \rangle$ dennoch erfüllt.

- $\langle RM8 \rangle$ Möglichkeit zur Erstellung eines Benutzerkontos für den WebService
  Dieses Kriterium ist noch nicht erfüllt. Es soll möglich sein, dass der Benutzer sich mittels E-Mail-Adresse und Passwort am WebService registrieren soll. Eine Übertragung zu dem WebService fehlt bis jetzt.

- $\langle RM9 \rangle$ Möglichkeit zum Login am WebService
  Dieses Kriterium ist noch nicht erfüllt. Es soll möglich sein, dass der Benutzer sich mittels seiner E-Mail-Adresse und seinem gewählten Passwort am WebService anmelden kann. Eine Übertragung zu dem WebService und eine Überprüfung, ob der Benutzer sich vorher registriert hat, fehlt bis jetzt.

5.1.2 WebService

Kriterien des WebService:

- $\langle RM10 \rangle$ Registrierung von Mobilgeräten und Verwaltung der Anmeldedaten
  Dieses Kriterium wird erfüllt. Bei der Registrierung nimmt der WebService E-Mail-Adresse und Passwort als Nutzerdaten entgegen und speichert diese in der Datenbank nach einer Prüfung, ob die E-Mail-Adresse schon existiert. Bei jeder Anmeldung werden die Anmeldedaten mit den Daten aus der Datenbank verglichen und der App wird eine SessionID zur Datenübertragung bzw. eine Fehlermeldung bei nicht erfolgreicher Anmeldung übergeben.
- $\langle RM11 \rangle$ Entgegennehmen der von den Mobilgeräten gesendeten Daten und Speicherung in einer MSSQL Datenbank
  Dieses Kriterium wird erfüllt. Der WebService nimmt ein Paket an Datensätzen von Mobilgeräten entgegen und speichert diese einzeln in der Datenbank.
- $\langle RM12 \rangle$ Grundlegende Möglichkeiten zur Auswertung und Visualisierung der vorhandenen Daten
  Dieses Kriterium ist bisher noch nicht erfüllt. Geplant ist, dass die Daten nach verschiedenen Kriterien gefiltert werden können, z.B. nach Provider oder Handymodell, und es dann Möglichkeiten gibt, die Messdaten als Tabelle oder als ein Overlay auf einer GoogleMaps-Karte (s. $\langle RC6 \rangle$) darzustellen.

5.2 Sollkriterien

In der Folge Kommentare zur Erfüllung der angestrebten aber nicht zwingend nötigen Kriterien:

5.2.1 App

Kriterien der Android-App:

- $\langle RS1 \rangle$ Energiesparende Ressourcennutzung
  Dieses Kriterium wird erfüllt. Der größte Ressourcenverbrauch der Anwendung fällt auf die Verwendung des GPS-Moduls zum Bestimmen der aktuellen Position. Der Nutzer hat die Möglichkeit, den Messintervall zwischen zwei Positionsbestimmungen in den Einstellungen selbst zu bestimmen. Zwischen den Messungen wird das GPS-Modul deaktiviert.
- $\langle RS2 \rangle$ Anonymisierung der Messdaten
  Der Benutzer kann einstellen, ob seine Messdaten mit seinem Mobilgerät verbunden werden oder nicht. Ein Bezug zu den Benutzerdaten erfolgt nicht.
- $\langle RS3 \rangle$ Englische und deutsche Lokalisierung
  Dieses Kriterium wird durch die von Android bereitgestellte Lokalisierungsmethode erfüllt. Dabei wird die Lokalisierung an die eingestellte Systemsprache von Android abhängen. Englisch ist dazu die Standardspracheinstellung.
  Wenn das Mobilgerät auf deutsche Sprache eingestellt ist, wird die X-Map App ebenfalls auf Deutsch dargestellt. Bei allen anderen Spracheinstellungen wird die X-Map App auf Englisch angezeigt.
- $\langle RS4 \rangle$ Bei Überschreitung einer maximalen Antwortzeit ist der Nutzer darauf hinzuweisen
  Der Benutzer wird bei einer zu langen Antwortzeit durch einen Ladebildschirm darauf hingewiesen, dass die X-Map App zum Arbeiten noch Zeit benötigt und noch nicht antworten kann.

- $\langle RS5 \rangle$ Bewegungsmuster dürfen nicht nachvollzogen werden können
  Dieses Kriterium wird erfüllt. Die Messdaten der App werden dazu ohne die dazugehörigen Zeitpunkte der Messungen an den WebService übertragen. Die Daten werden auf dem WebService nur mit dem Zeitpunkt der Übertragung markiert. Dadurch wird ein Bewegungsmuster nicht ersichtlich.
  Innerhalb der App kann der Benutzer sein Bewegungsmuster aber nachvollziehen. Der Zeitpunkt der jeweiligen Messung wird lokal gespeichert. Diese Daten bleiben aber lokal auf dem Mobilgerät und werden nicht weitergegeben.

  Dieses Soll-Krierium wurde zu Beginn falsch interpretiert und bezieht sich darauf, dass das Bewegungsmuster auf dem WebService und damit durch Dritte nicht nachvollziehbar sein darf. Der Benutzer selbst kennt sein Bewegungsmuster und kann dieses somit auf seinem Mobilgerät nachvollziehen.

- $\langle RS6 \rangle$ Verhalten im Fehlerfall soll vom Nutzer konfigurierbar sein
  Dieses Kriterium ist noch nicht erfüllt. Eine Spezifizierung erfolgt im Laufe des Entwicklungsprozesses.

5.2.2 WebService

Kriterien des WebService:

- $\langle RS7 \rangle$ Erweiterbarkeit der Kapazität ohne Softwareaktualisierung
  Dieses Kriterium wird weitgehend umgesetzt. Wird auf eine Datenbank mit höherer Kapazität gewechselt, müsste aktuell gegebenenfalls die Adresse der Datenbank im Quellcode des WebService aktualisiert werden.

5.3 Kannkriterien

Es folgen Kommentare zu den wünschenswerten aber weniger wichtigen Kriterien:

5.3.1 App

Kriterien der Android-App:

- $\langle RC1 \rangle$ Fehlerprotokoll / Statusbericht
  Dieses Kriterium ist noch nicht erfüllt.
- $\langle RC2 \rangle$ Fortschrittsanzeigen bei Übertragungsverzögerungen
  Dieses Kriterium ist noch nicht erfüllt.
- $\langle RC3 \rangle$ Nachtmodus (mit einstellbaren Zeiten) / Sparmodus / Zugmodus
  Der Sparmodus wird durch das Soll-Kriterium $\langle RS1 \rangle$ erfüllt. Der Benutzer kann die Dauer zwischen den Messungen einstellen.
  Für den Nachtmodus und Zugmodus gibt es bis jetzt keine Planung.
- $\langle RC4 \rangle$ Overlay für eine visuelle Auswertung der Empfangsqualität entlang der eigenen Spur mittels eines Kartendienstes
  Dieses Kriterium wird erfüllt und ist in die Planung des Projektes eingebunden. Der Benutzer kann sich auf dem Kartendienst "`Google Maps`" seine Empfangsqualität, durch colorierte, variable, geometrische Strukturen, anzeigen lassen.
- $\langle RC5 \rangle$ Keine sofortige Interaktion des Nutzers bei Fehlern in Hintergrundfunktionen nötig
  Dieses Kriterium ist bis jetzt teilweise erfüllt. Wenn Hintergrundfunktionen ausfallen, wird der Benutzer nicht benachrichtigt und muss nicht eingreifen. Bei anderen Funktionen kann dies aber zurzeit noch der Fall sein.

5.3.2 WebService

Kriterien des WebService:

- $\langle RC6 \rangle$ Overlay für eine visuelle Auswertung räumlicher Versorgungseigenschaften mittels eines Kartendienstes
  Dieses Kriterium ist bisher noch nicht erfüllt. Die gesammelten Messdaten könnten von der Web-Applikation als ein Overlay über einer Karte dargestellt werden.

- $\langle RC7 \rangle$ Zusätzliche Möglichkeit zur Verwendung einer MySQL Datenbank
  Dieses Kriterium kann erfüllt werden. Dazu verändert man den Connection-String in der config-Datei des WebService, der die benötigten Daten zur Verbindung mit der Datenbank enthält. Nach aktuellem Stand sind die genutzten SQL-Statements bei MySQL und MSSQL gleich.
- $\langle RC8 \rangle$ E-Mail-Adressen verifizieren
  Dieses Kriterium ist bisher noch nicht erfüllt. Dabei soll nach der Registrierung am WebService eine E-Mail vom Server verschickt werden, die einen Link o.ä. enthält, wodurch der Nutzer seine E-Mail-Adresse verifizieren kann.

5.4 Abgrenzungskriterien

Hernach Kommentare zu den explizit nicht zu erfüllenden Kriterien:

- $\langle RW1 \rangle$ Normalen Benutzern die Auswertung der Daten auf dem Server ermöglichen
  Geplant ist die Unterscheidung von Gruppen verschiedener Benutzer, z.B. „Forscher“ und „App-User“. Der Zugriff auf die Möglichkeiten der Web-Applikation kann so Gruppenweise geregelt werden.
- $\langle RW2 \rangle$ Veröffentlichung der App im Google Play Store
  Stattdessen soll die App zunächst nur direkt in Form ihrer Application Package-Datei weitergegeben werden.

6 Glossar

.NET Framework:
Bezeichnet eine von Microsoft entwickelte Software-Plattform zur Entwicklung und Ausführung von Anwendungsprogrammen. .NET besteht aus einer Laufzeitumgebung (Common Language Runtime), in der die Programme ausgeführt werden, sowie einer Sammlung von Klassenbibliotheken, Programmierschnittstellen und Dienstprogrammen (Services).

Android:
„Android“ ist der Name eines von Google entwickelten Betriebssystems für Mobilgeräte.

Android-App:
Kurzform für Android-Applikation

ANDROID_ID:
Die ANDROID_ID ist eine eindeutige 64-bit Folge, welcher bei der Erstaktivierung eins Android-Gerätes generiert wird und potentiell auch nach einem Reset erhalten bleibt.

AndroidPlot:
AndroidPlot ist ein von einer kleinen Gruppe von Mitarbeitern in und um Silicon Valley erstelltes externes Paket, dass eine Graphendarstellung in Android ermöglicht. Dabei ist AndroidPlot für jede öffentliche, private und gewerbliche Nutzung zugägnlich.

Android Application Package File (.apk)
Komprimiertes Java Archiv, welches speziell zum Packen von Android-Applikationen genutzt wird.

Android SDK:
Software Development Kit (SDK), die die Java-Programmiersprache erweitert, um eine Programmierung für Androidgeräte zu ermöglichen.

Eclipse:
Ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als integrierte Entwicklungsumgebung (IDE) für die Programmiersprache Java genutzt.

Overlay:
Overlays sind Objekte auf der Karte, die an Breitengrad/Längengradkoordinaten gebunden sind und mit der Karte zusammen bewegt werden, wenn Sie diese verschieben oder zoomen. Overlays

stehen für Objekte, die Sie zur Karte hinzufügen, um Punkte, Linien, Bereiche oder Sammlungen von Objekten anzugeben.

Google Play Store:
Ein E-Commerce-System zum Vertrieb von Android-Apps und deren Verwaltung, sowie zum Erwerb von Büchern, Musik und Filmen in digitaler Form.

GPS:
Das Global Positioning System (GPS) ist ein ursprünglich militärisches System zur weltweiten satellitengestützen Positionsbestimmung.

IMEI:
Die International Mobile Station Equipment Identity (IMEI) ist eine eindeutige 15-stellige Seriennummer, anhand derer jedes GSM- oder UMTS-Endgerät theoretisch eindeutig identifiziert werden kann.

Microsoft Visual Studio 2010:
Eine von dem Unternehmen Microsoft angebotene, integrierte Entwicklungsumgebung für verschiedene Programmiersprachen.

SOAP:
Das *Simple Object Access Protocol* ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf XML zur Repräsentation der Daten.

Windows Server 2008 R2:
Ein speziell für Server optimiertes Betriebssystem von Microsoft. Wurde parallel mit dem bekannten Betriebssystem Windows 7 entwickelt und kam im Oktober 2009 auf den Markt.