Technische Universität Braunschweig

Institut für Nachrichtentechnik

X-Map

Android-App zur qualitativen Erfassung von Mobilfunknetzen

Software-Entwicklungspraktikum (SEP)
Sommersemester 2013

Testspezifikation

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, 10. Juli 2013

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
0.1.0.0	19.04.2013	Timo Schwarz	abgenommen	Kapitel 1 entworfen
0.1.1.0	19.04.2013	Sofia Ananieva	abgenommen	Kapitel 1 erweitert
0.1.2.0	19.04.2013	Julian Troegel	abgenommen	Kapitel 2.2 & 2.3 entworfen
0.1.3.0	19.04.2013	Timo Schwarz	abgenommen	Kapitel 2.3 erweitert, 2.5 entworfen
0.1.4.0	20.04.2013	Julian Troegel	abgenommen	Kapitel 2.5.1 überarbeitet
0.1.4.1	21.04.2013	Julia Kreyßig	abgenommen	Kapitel 3.1
0.1.5.0	21.04.2013	Yurtseven	abgenommen	Kapitel 3.3
0.1.5.1	22.04.2013	Timo Schwarz	abgenommen	Kapitel 3.2, 3.3 erweitert
0.1.5.2	22.04.2013	Bauerfeld	abgenommen	Kapitel 3.3 erweitert
0.1.5.3	22.04.2013	Ferhat Cinar	abgenommen	Kapitel 3.3 überarbeitet
0.1.5.4	23.04.2013	Bauerfeld	abgenommen	Kapitel 3.3 überarbeitet
0.1.6.0	21.04.2013	Sofia Ananieva	abgenommen	Kapitel 2.1 überarbeitet
0.1.7.0	22.04.2013	Sofia Ananieva	abgenommen	Einleitungen Kapitel 2 und 3
0.1.8.0	22.04.2013	Julian Troegel	abgenommen	Kapitel 2 überarbeitet
0.1.9.0	23.04.2013	Sofia Ananieva	abgenommen	Kapitel 2.4 und 3.3 erweitert
0.2.0.0	23.04.2013	Andreas Hecker	abgenommen	Kapitel 3.3 erweitert
0.2.1.0	23.04.2013	Julian Troegel	abgenommen	Kapitel 2 und 3.3 erweitert
0.2.2.0	24.04.2013	Ferhat Cinar	abgenommen	Kapitel 3.3 überarbeitet
1.0.0.0	24.04.2013	Timo Schwarz	abgenommen	Abgabeverison
1.1.0.0	07.07.2013	Julian Troegel	abgenommen	Kapitel 2, 3 und 4 überarbeitet
1.2.0	10.07.2013	Timo Schwarz	abgenommen	Kapitel 5 überarbeitet
1.3.0	10.07.2013	Julian Troegel	abgenommen	Review gelesen
1.5.0	10.07.2013	Julian Troegel, Timo Schwarz	abgenommen	Abgabeversion

Inhaltsverzeichnis

1 Einleitung

Bei dem vorliegenden Dokument handelt es sich um die endgültige Testspezifikation für die Android App „X-Map“ im Rahmen des Softwareentwicklungspraktikums 2013.

Das X-Map-Projekt besteht aus verschiedenen Software-Komponenten innerhalb von zwei verschiedenen Umgebungen:

Zunächst wäre da der WebService, welcher gemeinsam mit einer Datenbank auf einem zentralen Server läuft. Dieser muss sowohl Datensätze von den Mobilgeräten entgegennehmen und in die Datenbank eintragen als auch ein Interface zur Auswertung der Messdaten zur Verfügung stellen. Er sollte in der Lage sein eine große Zahl an Übertragungen von den Mobilgeräten gleichzeitig verarbeiten zu können während er zusätzlich eine gute Benutzbarkeit der Auswertungsoberfläche zur Verfügung stellt.

Auf der anderen Seite wäre da die Android-Applikation, die auf den Mobilgeräten die Mobilfunkdaten sammelt und bis zur Löschung abspeichert. Neben der Korrektheit der Datensammlung, -sicherung und -übertragung ist für sie von Bedeutung, dass die angebotenen Datenschutzoptionen auch tatsächlich angewendet werden. Darüber hinaus ist sie als Anwendung für den Dauerbetrieb konzeptioniert, sodass auch ein zuverlässiger Betrieb über längere Zeit sichergestellt werden sollte.

Der Schwerpunkt der Testfälle ist in diesem Dokument der Abnahmetest. Im Testplan werden die zu testende Komponenten und Funktionen identifiziert und adäquate Testfälle festgelegt. Des weiteren wird eine allgemeine Vorgehensweise für die einzelnen Testobjekte beschrieben. Der Aufbau der Testspezifikation entspricht dem IEEE Standard 829 für Software-Testdokumentation.

Die Testspezifikation richtet sich an den Kunden und die Auftragsnehmer.

2 Testplan

Der Testplan konzentriert sich auf die Qualitätssicherung. In diesem Kapitel werden die zu testenden Komponenten und Funktionen identifiziert und adäquate Testfälle festgelegt. Des weiteren wird ausführlich eine allgemeine Vorgehensweise für die einzelnen Testobjekte beschrieben und welche Testumgebung genutzt wird.

2.1 Zu testende Komponenten

In diesem Dokument werden folgende Komponenten getestet und analysiert:

Client:

- Software
- Datenbestand
- Serververbindung
- GUI für Nutzer-Frontend

Server:

- Datenbankzugriff
- Benutzerverwaltung
- Messdaten
- GUI für Administrator-Frontend

2.2 Zu testende Funktionen/Merkmale

- $\langle F10 \rangle$ Daten messen
- $\langle F20 \rangle$ Visualisierung der Android-Applikation
- $\langle F30 \rangle$ Daten für WebService
- $\langle F40 \rangle$ Daten löschen
- $\langle F50 \rangle$ Daten lokal speichern
- $\langle F60 \rangle$ Einstellungen für lokal zu speichernden Daten
- $\langle F70 \rangle$ Datenschutz
- $\langle F80 \rangle$ Registrierung eines Benutzers
- $\langle F90 \rangle$ Anmelden
- $\langle F100 \rangle$ Registrierungen eines Mobilgerätes
- $\langle F110 \rangle$ Speicherung der Daten
- $\langle F120 \rangle$ Auslesen der Daten
- $\langle F130 \rangle$ Visualisierung auf WebService

Hier werden zu testende Kombinationen der Produktfunktionen aufgeführt:

- $\langle F20 \rangle + \langle F50 \rangle$ Die lokal gespeicherten Daten werden richtig visualisiert
- $\langle F50 \rangle + \langle F60 \rangle$ Nur die richtig eingestellten Daten werden lokal gespeichert
- $\langle F80 \rangle + \langle F100 \rangle$ Wenn ein Benutzer sich registriert, wird auch ein Mobilgerät auf dem WebService registriert
- $\langle F30 \rangle + \langle F110 \rangle$ Vom Mobilgerät gesendeten Daten werden auf dem WebService gespeichert
- $\langle F120 \rangle + \langle F130 \rangle$ Ausgelesene Daten werden richtig visualisiert
- $\langle F30 \rangle + \langle F40 \rangle + \langle F60 \rangle$ Zwischengespeicherte Daten werden auf dem Mobilgerät gelöscht, die nicht lokal gespeichert werden, wenn eine Übertragung zum WebService stattfindet. (entfernt)
- $\langle F40 \rangle + \langle F50 \rangle + \langle F60 \rangle$ Älteste Daten werden gelöscht, wenn neue Daten lokal gespeichert werden und die maximale Speichergröße überschritten wird

2.3 Nicht zu testende Funktionen

Grundsätzlich nicht getestet werden alle vorausgesetzten Betriebsmittel, was auch alle zugrundeliegenden Betriebssysteme mit einschließt. Darüber hinaus werden die verwendeten Tools von Drittanbietern nicht gesondert getestet. Bei all diesen wird davon ausgegangen, dass der Hersteller bzw. Urheber selbst hinreichende Tests getätigt hat. Insbesondere sind dies:

- AndroidPlot
- Google Android
- Google Android SDK (Eclipse)
- Google Play Services
- Google Maps
- JUnit
- kSOAP2
- Microsoft Internet Information Services
- Microsoft SQL Server 2010
- Microsoft Visual Studio 2010
- Microsoft Windows Server 2008 R2
- SoapUI
- TortoiseSVN
- Wireshark

2.4 Vorgehen

In dem X-Map Projekt können drei Bereiche separat voneinander getestet werden: Funktionalitäten des Clients, Funktionalitäten des Servers und schließlich das Zusammenspiel des Client-Server-Systems. In folgenden Schritten wird beim Testen vorgegangen:

a) Komponententest
Mit Ausnahme der GUI-Komponenten werden die Klassen des Clients mit JUnit-Testfällen geprüft und die Klassen des Servers mit Unit-Testfällen von Microsoft Visual Studio. Während der Implementierung werden bereits Blackbox-Testfälle vom jeweiligen Autor erstellt und im „Test-first"-Verfahren begleitend zur Implementierung durchgeführt. Nach Abschluss der Implementierung einer Klasse ist so bereits ihre (unabhängige) Funktionalität sichergestellt.

b) Integrations- und Funktionstest
Wurde eine Klasse oder Komponente fertiggestellt und erfolgreich getestet, folgt nach dem Bottom-Up-Prinzip die Integration in das bisher bestehende Programm und die Verknüpfung mit der GUI. In diesem Zuge erfolgt durch den Autor über die GUI direkt der Integrationstest und der Funktionstest für Anwendungsfälle.
Anfangs muss die Integration der Datenbankanbindung geprüft werden, da das Mapping der Datenbank die unterste Schicht des Projektes bildet.
In welcher Reihenfolge die Komponenten unter Berücksichtigung ihrer Abhängigkeit integriert werden, wird zu einem späteren Zeitpunkt konkretisiert.

c) Abnahmetest
Die Anwendungsfälle werden aus der Anforderungsspezifikation geprüft. Mindestanforderung hierfür ist es, jeden Fall einmal auf seine korrekte Funktionalität zu testen. Die Android-Applikation wird dabei sowohl vom Kunden, als auch von unabhängigen Personen, geprüft. Probleme oder Auffälligkeiten werden dokumentiert und gegebenenfalls behoben.
Der WebService wird nur durch den Kunden getestet, da der WebServer keinen öffentlichen Zugang hat.

2.5 Testumgebung

In diesem Kapitel werden die zum Testen verwendeten Umgebungen und Tools näher beschrieben.

2.5.1 Android-Applikation

Die Android-App wird in Java programmiert, daher wird für Unit-Tests die JUnit-Testsuit benutzt. Dabei handelt es sich um eine speziell an Android angepasste JUnit-Testsuit in Version 3. Diese Tests sind von der Maschine, auf der sie ausgeführt werden, unabhängig. JUnit in Version 4 (oder höher) kann nicht für die Android SDK verwendet werden. Für normale Java-Klassen, ohne Bezug auf die Android SDK, können aber durch JUnit 4 (oder höher) getestet werden.
Die JUnit-Tests werden auf einem Mobilgerät oder dem vom Android SDK bereitgestellten Android-Emulator durchgeführt und die Auswertung der Test erfolgt in Eclipse.
Zusätzlich wird das LogCat aus dem Android SDK, welches in Eclipse integriert wurde, benutzt. Dabei werden Log-Nachrichten in den Programmcode geschrieben, welche bei der Ausführung ausgeführt werden und die Nachricht in das LogCat schreiben. Hauptsächliche Anwendung findet das LogCat bei dem Test vom Testfall $\langle T1000 \rangle$ und bei White-Box-Tests.
Ein Endbenutzer-Test wird auf einer möglichst großen Zahl an privat genutzten Mobilgeräten verschiedener Leistungsfähigkeiten und Android-Versionen geschehen. An diesem Test nehmen in jedem Fall die Team-Mitglieder sowie nach Möglichkeit außenstehende Nutzer mit durchschnittlichen Kenntnissen teil.

2.5.2 WebService

Die Möglichkeit für Unit-Tests ist in Visual Studio 2010 in den Varianten „Premium“ und aufwärts integriert, weshalb diese Testsuit genutzt werden wird. Diese Tests werden auf beliebigen Maschinen erfolgen. Auf diesem Weg soll auch der Test der Datenbank erfolgen. Später wird der Service in der realen Umgebung auf dem zukünftigen Server installiert und mit Anwenderszenarien getestet werden.
Tests der Verbindung zwischen Mobilgeräten und WebService sollen mithilfe der Software SoapUI erfolgen.
Das Tool Wireshark wird verwendet, um alle im Server eingehenden und ausgehenden Nachrichten zu erkennen. Man kann also in Kombination mit dem LogCat für den Client überprüfen, ob der Datentransfer korrekt erfolgt.

3 Abnahmetest

Der Abnahmetest ist eine Überprüfung des Softwareprodukts durch den Auftraggeber und zielt auf eine Abnahme durch den Kunden. In diesem Kapitel werden die zu testenden Anforderungen sowie Testverfahren aufgeführt. Abschließend wird auf alle Testfälle einzeln eingegangen.

3.1 Zu testende Anforderungen

App: Registrieren, Anmelden, Daten sammeln/graphisch darstellen lassen, Einstellungen bearbeiten
WebService: Daten visualisieren

3.2 Testverfahren

Der Abnahmetest des Clients erfolgt auf verschiedenen Geräten durch Endnutzer, der des Servers über verschiedene Rechner, welche Zugriff zur Administratoren-Oberfläche zur Verfügung stellen können. Es werden Testfälle definiert, durch welche die Produktfunktionen systematisch getestet werden.

Da sich das Projekt aktuell noch in der Analysephase befindet, können zu skriptbasierten Tests noch keine Aussagen getroffen werden.

3.3 Testfälle

Testfall $\langle T100 \rangle$ - Registrierung am WebService

Ziel

Überprüfung des Normalfalls: Nutzer legt ein Konto mit allen benötigten Informationen auf dem Server an.

Objekte/Methoden/Funktionen

In diesem Testfall werden die Komponenten Applikation und Server und die Funktionen $\langle F80 \rangle$ und $\langle F100 \rangle$ ausgeführt.
Zu testende Methoden:

- Client.register()
- Client.registerSend()
- Client.login()

Zu testende Klassen:

- CreateAccountActivity
- Client

Pass/Fail Kriterien

Pass: Das System enthält die Informationen des Nutzers und speichert diese in der Datenbank ab.

Fail: Das System ist in dem Zustand, in welchem es sich vor dem Anwendungsfall befand.

Vorbedingung

Die App ist auf einem Android Smartphone installiert. WebService ist erreichbar.

Einzelschritte

Eingabe:
a) E-Mail-Adresse eingeben
b) Passwort eingeben
c) Auf „Registrieren“ klicken

Daraufhin werden die Registrierungsdaten zum Server übertragen und mit der Benutzerdatenbank abgeglichen.

Ausgabe:
Der Nutzer erhält auf der Anwendung eine Meldung, dass die Registrierung erfolgreich war.

Alternative Ausgabe:
a) Der Nutzer erhält eine Meldung, dass die von ihm eingegebene E-Mail-Adresse schon verwendet wird und wird aufgefordert, eine andere Adresse zu nutzen.
b) Der Nutzer erhält eine Meldung, dass die Registrierung fehlgeschlagen ist. Eine Verbindung zum WebService ist nach dem Klicken auf „Registrieren“ mangels Signalstärke nicht möglich. Der Nutzer soll die Informationen für die Registrierung erneut abschicken.

Beobachtungen / Log / Umgebung

Eintrag von E-Mail-Adresse und Passwort in der Datenbank des WebServices.

Testfall $\langle T200 \rangle$ - Anmeldung am WebService

Ziel

Überprüfung des Normalfalls: Zum Zweck der Datenübertragung meldet sich ein bereits registrierter bzw. ein noch nicht registrierter Nutzer am WebService an.

Objekte/Methoden/Funktionen

In diesem Testfall werden die Komponenten Applikation und Server und die Funktion $\langle F90 \rangle$ ausgeführt.
Zu testende Methoden:

- Client.login()
- Client.retrieveLoginData()
- Client.register()
  Zu testende Klassen:
  - LoginActivity
  - Client

Pass/Fail Kriterien

Pass [Benutzer registriert]: Anmeldung erfolgreich, Datenübertragung und Meldung wird starten.

Pass [Benutzer nicht registriert]: Ausgabe einer Fehlermeldung. Eine Neu-Registrierung wird angeboten.

Fail [Benutzer registriert]: Anmeldung schlägt fehl. Ausgabe einer Fehlermeldung.

Vorbedingung

1) App ist installiert
2) Server ist vom Client aus erreichbar
3) Benutzer ist registriert

Einzelschritte

Eingabe:

a) E-Mail-Adresse eingeben

b) Passwort eingeben

c) Auf „Anmelden“ klicken

Ausgabe:

Der Nutzer erhält eine Meldung, dass der Anmeldevorgang erfolgreich war.

Alternative Ausgabe:

a) Anmeldeinformationen falsch. Der Vorgang muss wiederholt werden.

b) Server ist überlastet. Später erneut versuchen.

Beobachtungen / Log / Umgebung

Auf dem Mobilgerät wird der Zustand eingeloggt gespeichert.

Abhängigkeiten

Der Testfall $\langle T200 \rangle$ ist abhängig vom vorherigen Testfall $\langle T100 \rangle$. (Siehe Vorbedingung)

Testfall $\langle T300 \rangle$ - Daten sammeln und Ergebnisse graphisch darstellen

Ziel

Überprüfung des Normalfalls: Graphische Ausgabe der ermittelten Daten.

Objekte/Methoden/Funktionen

Fall [WebServer]: Test für Funktionskombination $\langle F120 \rangle$ + $\langle F130 \rangle$ ausgeführt.

Fall [Anwendung]: Test für Funktionskombination $\langle F20 \rangle$ + $\langle F50 \rangle$ ausgeführt.
Zu testende Methoden:

- Database.createCursorForXYChart()
- Database.createCursorForGoogleMaps()
- Database.createCursorForTable()
- DBForPresentation.getCursor()

Zu testende Klassen:

- MapsActivity
- DBTableActivity
- XYChartActivity
- DBForPresentation
- Database

Pass/Fail Kriterien

Überprüfung, ob Daten graphisch dargestellt werden können. Wenn nicht wird eine leere Visualisierung gezeigt.

Pass [Messdaten vorhanden]: Graphendarstellung der Messdaten.
Pass [Keine Messdaten vorhanden]: leere Visualisierung.

Fail: Keine Darstellung oder Fehler.

Vorbedingung

Fallunterscheidung: Lokal [Messdaten vorhanden] *oder* Lokal [Keine Messdaten vorhanden]

Einzelschritte

Eingabe:
Anwendung:
a) (optional) Warten, bis Messdaten gesammelt wurden.
b) Die graphische Darstellung in der App aufrufen.

WebService:
a) (optional) Messdaten filtern.
b) (optional) Warten, bis gesammelte Messdaten an WebService übermittelt wurden.
c) Die graphische Darstellung in der WebApplikation aufrufen.

Ausgabe:
Graph mit den gesammelten Werten (Mobilgerät), Karte mit Overlay (WebApplikation)

Alternative Ausgabe:
Eine leere Darstellung, wenn keine Messwerte vorhanden sind.

Testfall $\langle T400 \rangle$ - Einstellung lokal zu speichernder Daten

Ziel

Überprüfung des Normalfalls: Der Nutzer wählt lokal zu speichernde Daten aus. Überprüfung, ob lediglich vom Nutzer gewählte Daten gespeichert werden.

Objekte/Methoden/Funktionen

Anwendung: Test wird für Funktionskombination $\langle F50 \rangle + \langle F60 \rangle$ ausgeführt.
Zu testende Methoden:

- Database.insertDataSet()
- Database.createDataObject()
- SettingsHelper.get_pref_ber()
- SettingsHelper.get_pref_cellid()
- SettingsHelper.get_pref_lac()
- SettingsHelper.get_pref_phonetype()

Zu testende Klassen:

- Database
- SettingsHelper

Pass/Fail Kriterien

Pass: Ausschließlich vom Nutzer gewählte Daten werden in der lokalen Datenbank gespeichert.

Fail: Nicht vom Nutzer gewählte Daten werden gespeichert.

Fail: Es werden keine Daten gespeichert.

Vorbedingung

Die App ist auf einem Android Smartphone installiert.

Einzelschritte

Eingabe:

a) Einstellen der lokal zu speichernden Daten über die Seite Einstellungen

b) Aufzeichnung von Daten durch die Anwendung

c) Auslesen der Datenbank (lokal oder durch Übertragung an WebService)

Ausgabe:

Werte des gespeicherten Datensatzes

Beobachtungen / Log / Umgebung

Die Daten die nicht gespeichert werden bzw. gespeichert werden, werden in den Einstellungen gespeichert.

Testfall $\langle T500 \rangle$ - Vom Mobilgerät gesendeten Daten werden auf dem Web-Service gespeichert

Ziel

Überprüfung des Normalfalls: Vom Nutzer übertragene Messdaten wurden auf dem Webservice gespeichert.

Objekte/Methoden/Funktionen

In diesem Testfall werden die Komponenten Applikation und Server und die Funktionen $\langle F30 \rangle + \langle F110 \rangle$ ausgeführt.

Pass/Fail Kriterien

Pass: Der WebService hat die Daten vom Mobilgerät erhalten und in einer Datenbank gespeichert.

Pass: Datensatz existiert bereits in der Datenbank des WebServices und wird nicht gespeichert.

Fail: Datensatz wird nicht an den WebService gesendet.

Fail: Datensatz konnte aus Fehlergründen nicht in der Datenbank des WebServices gespeichert werden.

Vorbedingung

1) Die App ist auf einem Android Smartphone installiert.
2) Der User ist registriert und angemeldet.
3) Der User erlaubt Datenversendung.
4) Der Server ist vom Client aus erreichbar.

Einzelschritte

Eingabe:

a) Eine Messung wird im Mobilgerät eines registrierten und angemeldeten Nutzers durchgeführt.
b) Übermittlung des neuen Datensatzes an den WebService.

Ausgabe:

Auf dem Mobilgerät die Anzahl der übertragenden und gespeicherten Datensätze.

Alternative Ausgabe:

a) Daten werden aus Fehlergründen nicht an den WebService geschickt. Dem Nutzer wird keine zugehörige Information angezeigt.

b) Unveränderte Größe der Datenbank, da neue Daten redundant sind.

Beobachtungen / Log / Umgebung

Auf dem Mobilgerät werden alle Übertragungen im Log von Android eingetragen.
Auf dem WebService werden die Daten in der Datenbank gespeichert.

Abhängigkeiten

Der Testfall $\langle T500 \rangle$ ist abhängig vom Testfall $\langle T200 \rangle$.

Testfall $\langle T600 \rangle$ - Älteste Daten werden gelöscht, wenn Datenbank maximalen Speicher erreicht

Ziel

Überprüfung des Normalfalls: Der älteste Datensatz wird gelöscht, wenn die Datenbank den maximal eingestellten Speicherplatz erreicht und die neu gemessenen Daten gespeichert werden sollen oder der Benutzer die Datenbankgröße minimiert.

Objekte/Methoden/Funktionen

In diesem Testfall wird die Komponente Applikation und die Funktionen $\langle F40 \rangle$ + $\langle F50 \rangle$ + $\langle F60 \rangle$ ausgeführt.
Zu testende Methoden:

- Database.insertDataSet()
- Database.deleteOldestDataSet()
- SettingsHelper.get_pref_db_size()

Zu testende Klassen:

- Database
- SettingsHelper

Pass/Fail Kriterien

Pass: Der älteste Datensatz wird gelöscht und der aktuellste wird in der Datenbank gespeichert.

Fail: Der älteste Datensatz wird nicht gelöscht. Der maximale Speicherplatz wird überschritten.

Fail: Der aktuellste Datensatz wird nicht gespeichert.

Vorbedingung

1) Die App ist auf einem Android Smartphone installiert.
2) Der Benutzer hat eine maximale Speichergröße eingestellt.

Einzelschritte

Eingabe:

a) Eine maximale Speichergröße wird eingestellt.

b) Es werden so lange Messungen durchgeführt, bis der maximale Speicherplatz erreicht wird.

c) Es wird überprüft, ob der maximale Speicherplatz überschritten wird.

Alternative Eingabe:

a) Eine maximale Speichergröße wird eingestellt.

b) Es werden so lange Messungen durchgeführt und in der Datenbank gespeichert.

c) Die maximale Speichergröße wird unter der aktuellen Größe eingestellt.

Ausgabe:

Aktuelle Größe der Datenbank.

Abhängigkeiten

Der Testfall $\langle T600 \rangle$ ist abhängig vom Testfall $\langle T400 \rangle$.

Testfall $\langle T700 \rangle$ - Zwischengespeicherte Daten werden gelöscht, wenn diese übertragen wurden

Ziel

Überprüfung des Normalfalls: Zwischengespeicherte Daten, die nicht lokal gespeichert werden sollen, werden auf dem Mobilgerät gelöscht, wenn eine Übertragung zum WebService stattgefunden hat.

Dieser Testfall $\langle T700 \rangle$ benötigt keine Testdurchführung! Aufgrund des frühen Entwicklungsstands bei der ersten Testspezifikation, wurde der Test eingeführt. Im Laufe der Entwicklung wurde diese Funktion überarbeitet, so dass Daten auf dem Mobilgerät nur gelöscht werden, wenn die maximale Speichergröße überschritten wird oder der Benutzer manuell Daten löscht, aber nicht mehr, wenn eine Übertragung stattgefunden hat. Deshalb entfällt dieser Test.

Objekte/Methoden/Funktionen

In diesem Testfall wird die Komponente Applikation und die Funktionen $\langle F30 \rangle$ + $\langle F40 \rangle$ + $\langle F60 \rangle$ ausgeführt.

Pass/Fail Kriterien

Pass: Die zwischengespeicherten Daten werden gelöscht.

Fail: Die zwischengespeicherten Daten werden nicht gelöscht. Es wird eine geeignete Fehlermeldung ausgegeben.

Fail: Die Daten werden nicht übertragen. Eine Löschung wird nicht durchgeführt.

Vorbedingung

1) Die App ist auf einem Android Smartphone installiert.
2) Messungen finden satt, die Daten werden zwischengespeichert.
3) Die Daten können an den WebService übertragen werden.

Einzelschritte

Eingabe:
a) Messungen werden im Mobilgerät durchgeführt.
b) Die Daten werden an den WebService übertragen.
c) Es wird überprüft, ob die Daten erfolgreich übertragen wurden.

Ausgabe:

Letzter Datensatz wird dahingehend überprüft, ob nur noch lokal zu speichernde Daten vorhanden sind.

Abhängigkeiten

Der Testfall $\langle T700 \rangle$ ist abhängig vom Testfall $\langle T400 \rangle$.

Testfall $\langle T800 \rangle$ - Daten messen

Ziel

Überprüfung des Normalfalls: Wenn eine Messung auf dem Mobilgerät durchgeführt wird, sind danach Messdaten vorhanden.

Objekte/Methoden/Funktionen

In diesem Testfall wird die Komponente Applikation und die Funktion $\langle F10 \rangle$ ausgeführt. Zu testende Methoden:

- SignalStrengthListener.onSignalStrengthChanged()
- SaveServiceListener.onSignalStrengthChanged()

Zu testende Klassen:

- SignalStrengthListener
- SaveServiceListener
- MainActivity
- Alarm

Pass/Fail Kriterien

Pass: Die gemessenen Daten sind vorhanden.

Fail: Die Messdaten sind nicht vorhanden.

Vorbedingung

1) Die App ist auf einem Android Smartphone installiert.
2) Eine Messungen kann stattfinden.

Einzelschritte

Eingabe:
Eine Messung wird im Mobilgerät durchgeführt.

Ausgabe:
Die gemessenen Daten.

4 Integrationstest

In diesem Kapitel wird die Integration von mehreren Komponenten getestet. Dabei wird zwischen Client, WebService und Client+WebService unterschieden.
Bei dem Client fungieren die 3 Komponenten $\langle C11\rangle$, $\langle C12\rangle$ und $\langle C13\rangle$ nicht alleine. Die Komponente $\langle C12\rangle$ Backend dient als zentrale Schalteinheit der Applikation und kommuniziert mit den anderen beiden Komponenten $\langle C11\rangle$ GUI und $\langle C12\rangle$ ClientDatabase. Demnach wurden die Komponenten auch von Beginn der Entwicklung an gemeinsam erstellt und integriert.
Deshalb benötigen die Komponenten des Clients kein Integrationstest.

Bei der dem WebService wurden die Komponenten $\langle C21\rangle$, $\langle C22\rangle$, $\langle C23\rangle$, $\langle C24\rangle$, $\langle C25\rangle$ und $\langle C26\rangle$ ähnlich zum Client von Beginn an integriert. Die Komponenten sind untereinander von Beginn an abhängig und konnten deshalb nicht alleine konzipiert werden. Deshalb benötigen die Komponenten des WebServices keinen Integrationstest.

Die Komponenten $\langle C10\rangle$ und $\langle C20\rangle$ beschreiben die Integration von Client und WebService. Diese beiden Komponenten wurden einzeln erstellt und im Verlauf der Entwicklung miteinander integriert.
Dabei kommunizieren Client und Server über das SOAP-Protokoll. Das Mobilgerät hat dazu die Bibliothek kSOAP 2 verwendet und beim Server ist dies ein Bestandteil von Microsoft Visual Studio 2010. Deshalb braucht hier nicht getestet werden, ob das SOAP-Protokoll eingehalten wird.
Das Ziel bei der Integration von Client und WebService ist, dass die Daten, die von der einen Komponente verschickt werden korrekt bei der anderen Komponente eintreffen.

4.1 Zu testende Komponenten

- $\langle C10\rangle + \langle C20\rangle$

4.2 Testverfahren

Beim Test für die Integration der Komponenten $\langle C10 \rangle$ + $\langle C20 \rangle$ werden jeweils Daten vom Client und vom Server verschickt, welche auf der jeweils anderen Komponente eintreffen. Dabei werden dann die ausgehenden und eingehenden Daten miteinander verglichen. Dabei werden die Nachrichten aus dem LogCat von Eclipse auf der Clientseite und die Nachrichten von Wireshark auf der Serverseite eingetragen und miteinander verglichen.

4.2.1 Testskripte

Für den Test von $\langle T900 \rangle$ wird folgendes Testskript verwendet:

Nr	Testschritt	Erwartetes Resultat	Kommentar
1	Öffnen Sie auf einen Computer Eclipse	Eclipse startet	*Eclipse muss das Android SDK enthalten.*
2	Schließen Sie Ihr Mobilgerät an den Computer an und öffnen Sie in Eclipse das LogCat	Im LogCat werden alle vom Mobilgerät dargestellten Nachrichten aufgelistet	Auf dem Mobilgerät ist der Debug-Modus aktiv
3	Starten Sie die Filtereinstellungen des LogCat mit Klick auf das grüne Kreuz	Die Filtereinstellungen werden geöffnet	
4	Geben Sie ins Textfeld *by Application Name com.example.x_map* ein	Der Nachrichtenstrom im LogCat versiegt. Es werden nur noch Nachrichten zur Applikation angezeigt	
5	Öffnen Sie am Server Wireshark	Wireshark startet	
6	Starten Sie auf einem Android-Mobilgerät X-Map	X-Map wird gestartet	*Der Benutzer ist in der App nicht angemeldet*
7	Öffnen Sie das Anmeldeformular auf dem Mobilgerät	Das Anmeldeformular wird sichtbar	
8	Öffnen Sie das Registrierformular auf dem Mobilgerät	Das Registrierformular wird sichtbar	

9	Geben Sie eine E-Mail-Adresse und zweimal ein beliebiges Passwort ein und bestätigen Sie die Eingabe mit einem Klick auf *Registrieren*	Auf dem LogCat und im Wireshark werden Nachrichten sichtbar	*Der Benutzer hat sich bei Erfolg registriert*
10	Sie gelangen auf dem Mobilgerät zum Anmeldeformular mit ihren Registrierdaten. Melden sich mit einem Klick auf *Anmelden* an	Auf dem LogCat und im Wireshark werden weitere Nachrichten sichtbar	*Der Benutzer hat sich bei Erfolg angemeldet*
11	Warten Sie auf eine Datenübertragung oder starten Sie eine Übertragung indem Sie das App-Menü öffnen und auf *Jetzt synchronisieren* klicken	Auf dem LogCat und im Wireshark werden weitere Nachrichten, für jeden Datensatz eine, sichtbar	*Es wurden vor der Übertragung Daten gemessen und in der Datenbank der App gespeichert*
12	Vergleichen Sie alle im LogCat mit dem Tag *Client* markierten Nachrichten mit den Nachrichten im Wireshark	Die Nachrichten müssten gleiche Daten enthalten	*Es wird das SOAP-Protokoll verwendet. Die Nachrichten sind im XML-Format*

4.3 Testfälle

Testfall $\langle T900 \rangle$ **-** $\langle C10 \rangle + \langle C20 \rangle$

Ziel

Überprüfung des Normalfalls: Wenn eine Übertragung durchgeführt wird, wird diese korrekt durchgeführt und alle Daten werden während der Übertragung nicht verändert.

Objekte/Methoden/Funktionen

In diesem Integrations-Testfall werden die Komponenten Client und WebService und die Funktionen $\langle F30 \rangle + \langle F110 \rangle$, sowie $\langle F80 \rangle + \langle F100 \rangle$ und $\langle F90 \rangle$ getestet.
Serverseitig werden die zur Übertragung nötigen Methoden-Rümpfe vollautomatisch und transparent generiert, sodass für diese keine Tests möglich und nötig sind.
Zu testende Methoden:

- Client.register()
- Client.registerSend()
- Client.login()

- Client.sendData()

Zu testende Klassen:

- Client

Pass/Fail Kriterien

Pass [Anwendung zu Server]: Die Daten die von dem Mobilgerät gesendet werden, werden korrekt vom Server empfangen.
Fail [Anwendung zu Server]: Die Daten die von dem Mobilgerät gesendet werden, werden fehlerhaft oder gar nicht vom Server empfangen.

Pass [Server zu Anwendung]: Die Daten die von dem Server gesendet werden, werden korrekt vom Mobilgerät empfangen.
Fail [Server zu Anwendung]: Die Daten die von dem Server gesendet werden, werden fehlerhaft oder gar nicht vom Mobilgerät empfangen.

Vorbedingung

1) Die App ist auf einem Android Smartphone installiert.
2) Die Daten können an den WebService übertragen werden.
3) Die Daten können an das Smartphone übertragen werden.

Einzelschritte

Eingabe:
a) Der Benutzer registriert sich.
b) Daten werden übertragen.
c) Der Benutzer meldet sich an.
d) Daten werden übertragen.
e) Messdaten werden übertragen.

Ausgabe:
zu a) + b): Der Benutzer erhält eine Benachrichtigung über erfolgreiche Registrierung.
zu c) + d): Der Benutzer erhält eine Benachrichtigung über den Erfolg der Anmeldung.
zu e): Der Benutzer erhält eine Benachrichtigung über die erfolgreiche Datenübertragung.

Beobachtungen / Log / Umgebung

Im Log von Android wird der Datenausgangs- und Eingangsverkehr eingetragen.

5 Unit-Tests

Dieses Kapitel beschreibt die Unit-Tests der einzelnen Klassen. Dabei sind bei dem Client keine Unit-Tests möglich, da die Methoden aus Android-Methoden und -Aufrufen aufbauen. Ansonsten werden die Klassen durch die Bibliotheken “AndroidPlot“ und “kSOAP 2“ aufgebaut. Diese brauchen deshalb nicht getestet werden.
Das Entscheidende der Klassen und Methoden von der Applikation ist die Kommunikation der ganzen Klassen untereinander. Diese werden im Abnahmetest behandelt, da die Kommunikation von Beginn an vorhanden sein musste und damit der Integrationstest wegfällt.

Auf dem WebService gilt ein ähnliches Verhalten. Die Klassen besitzen überwiegend Bibliotheksaufrufe von Visual Studio oder sind nicht-testbare Datenbankzugriffe. Es müssen auf dem WebService die Kommunikation der Klassen untereinander getestet werden. Diese werden in dem Abnahmetest behandelt.
Außerdem wurden dennoch Unit-Tests für den WebService vorbereitet, diese konnten aber nicht ausgeführt werden, da das notwendige Wissen über die Entwicklerumgebung von Microsoft Visual Studio 2010 fehlt und nicht erkennbare und behebbare Fehler beim Testversuch entstanden sind, die Testdurchläufe nicht möglich machten.

Sowohl für den Client als auch für den WebService sind aber alle Klassen und Methoden im Gesamten schon seit längerem in einem Beta-Betrieb und wurden in realen Verhältnissen getestet. Dabei wurden zum Teil auch verschiedene Situationen betrachtet, um alle möglichen Fehler, die im Dauerbetrieb und im alltäglichen Betrieb auftreten können, zu erkennen und zu beheben.

Als einzige artverwandte Testart kann ein Black-Box-Test des Webservice via SoapUI durchgeführt werden, welcher im Nachfolgenden beschrieben wird.

5.1 Zu testende Komponenten

Der WebService bzw. die Schnittstelle IXMapService

5.2 Testverfahren

Black-Box-Test der bereitgestellten Methoden via SoapUI

5.3 Testfälle

Testfall $\langle T1000 \rangle$ - Black-Box-Test der Methoden des WebService

Ziel

Sicherstellung der korrekten Kommunikation des WebService über das SOAP-Protokoll

Methoden

a) registerUser()

b) login()

c) registerDevice()

d) sendData() in der Variante für einzelne Datensätze

e) endSession()

- getProvider()
- getDevices()
- getModels()

Die drei letzten Methoden können unabhängig von den vorausgehenden aufgerufen werden, da für sie keine SessionID notwendig und die Reihenfolge nicht von Bedeutung ist.

Pass/Fail Kriterien

Pass: Auf die jeweiligen Methodenaufrufe werden die korrekten Antworten gegeben, die Details werden in den Testprotokollen vermerkt.
Fail: Es werden teilweise oder vollständig inkorrekte Antworten gegeben.

Vorbedingung

Der momentane Inhalt der Datenbank muss im Voraus zumindest grob bekannt sein.

Einzelschritte

Oben angegebene Methoden werden ihrer Bestimmung und notwendigen Reihenfolge gemäß nacheinander ausgeführt.

Besonderheiten

Es muss insbesondere auf die Rückgabe mit inkorrekten Daten geachtet werden.