Technische Universität Braunschweig

Institut für Betriebssysteme und Rechnerverbund

Delay-Tolerant Jodel

Team 2

Software-Entwicklungspraktikum (SEP)
Sommersemester 2016

Testspezifikation

Auftraggeber
Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Prof. Dr.-Ing. Lars Wolf
Mühlenpfordtstraße 23
38106 Braunschweig

Betreuer: Björn Gernert und Dominik Schürmann

Auftragnehmer:

Name	E-Mail-Adresse
Melisa Camdzic	m.camdzic@tu-bs.de
Micha Horlboge	m.horlboge@tu-bs.de
Markus Jockusch	m.jockusch@tu-bs.de
Philipp Kuhn	ph.kuhn@tu-bs.de
Daniel Röhrs	d.roehrs@tu-bs.de
Edin Seferovic	e.seferovic@tu-bs.de
Tim Siebels	t.siebels@tu-bs.de
Sanjeeban Thevarasa	s.thevarasa@tu-bs.de

Braunschweig, 6. Juli 2016

Bearbeiterübersicht

Kapitel	Autoren	Kommentare
1	Daniel Röhrs, Markus Jockusch, Sanjeeban Thevarasa	
2	Daniel Röhrs, Markus Jockusch, Sanjeeban Thevarasa	Dieses Kapitel ist aus der Abnahmetestspezifikation übernommen.
2.1	Tim Siebels	Diese Sektion wurde ergänzt.
2.2	Daniel Röhrs, Markus Jockusch, Sanjeeban Thevarasa	
2.3	Daniel Röhrs, Markus Jockusch, Sanjeeban Thevarasa	
2.4	Daniel Röhrs, Markus Jockusch, Sanjeeban Thevarasa	
2.5	Daniel Röhrs, Markus Jockusch, Sanjeeban Thevarasa	
3	Melisa Camdzic, Edin Seferovic	Dieses Kapitel ist aus der Abnahmetestspezifikation übernommen.
3.1	Melisa Camdzic, Edin Seferovic	
3.2	Melisa Camdzic, Edin Seferovic	
3.3	Melisa Camdzic, Edin Seferovic	
4	Tim Siebels, Philipp Kuhn	
4.1	Tim Siebels, Philipp Kuhn	
4.2	Tim Siebels, Philipp Kuhn	

4.3	Tim Siebels, Philipp Kuhn	
5	Micha Horlboge, Philipp Kuhn	
5.1	Micha Horlboge, Philipp Kuhn	
5.2	Micha Horlboge, Philipp Kuhn	
5.3	Micha Horlboge, Philipp Kuhn	
6	Tim Siebels	

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Dieses Dokument soll sich mit der einwandfreien Qualität der Software durch Planung und Dokumentation von so genannten Softwaretests befassen. Die Softwaretests sollen dazu dienen die Funktionalität der Software zu überprüfen, um nach Abschluss der Entwicklung eine reibungslose Funktionalität gewährleisten zu können. Zudem sollen die Testverfahren kontinuierlich und parallel durch den gesamten Entwicklungsprozess durchgeführt werden, damit man den festgelegten Qualitätskriterien und Funktionen des Pflichtenhefts ständig nachkommt und diese Richtlinien somit einbehält.

Das im Rahmen des SEP 2016 zu entwickelnde Projekt *Delay-Tolerant Jodel* soll somit auf die wichtigsten Funktionalitäten und Qualitätsmerkmale geprüft werden. Unsere zum Projekt gehörige Applikation *DTNJodel* soll es ermöglichen, Studenten durch Veröffentlichung von Posts (Bilder, Nachrichten, Sprachaufnahmen) tagtäglich durch lustige Kommunikationen in Form von Kommentaren oder Votes zu den Posts in Verbindung zu halten, um das Studentenleben noch etwas mehr abzurunden. Die Verfügbarkeit der geposteten Daten wird durch die per GPS bestimmte Position eingeschränkt. So können nur Leute auf Beiträge einer Universität/Fachhochschule zugreifen, wenn sie sich in deren Nähe befinden.

Die Testverfahren sollen so durchgeführt werden, dass die App fehlerfrei genutzt werden kann und alle vorher festgelegten Qualitätsmerkmale erfüllt werden. Zu diesen einzuhaltenden Merkmalen zählen sowohl die Effizienz als auch die Zuverlässigkeit des gesamten Systems. Zudem sind für unser System die einzelnen Komponentengruppen sehr wichtig, wie zum Beispiel die Funkionen, Klassen und Module, welche als komplette Einheit getestet werden sollen. Im Allgemeinen spielen hauptsächlich die drei großen Arten von Tests, die im nachfolgenden Kapitel erwähnt werden, für das Testen während und nach der Entwicklung unserer *Jodel*-Anwendung die größte Rolle.

2 Testplan

In diesem Kapitel wird der Umfang und die Vorgehensweise der Qualitätssicherung beschrieben sowie zu testende Funktionen genannt.

2.1 Zu testende Komponenten

Aus dem Technischen Entwurf kann man folgende Komponenten entnehmen, welche alle getestet werden müssen:

- GUI <C10>
- LocationManager <C20>
- ContentManager <C30>
- DatabaseHandler <C40>
- Communicator <C50>

2.2 Zu testende Funktionen/Merkmale

Folgende Funktionen und Merkmale der Jodel-Applikation müssen getestet werden:

- Jodelliste anzeigen <F10>
- Jodel mit Kommentaren ansehen <F20>
- Jodel bewerten <F30>
- Kommentar bewerten <F35>
- Kommunikation via IBR-DTN <F40>
- Standort <F50>
- Jodel erstellen <F60>
- Kommentar erstellen <F70>
- Bild ansehen <F80>

- Meine Aktion <F90>

2.3 Nicht zu testende Funktionen

Alle Funktionen der Android-App sind in vollem Umfang zu testen. Ausgenommen sind selbstverständlich interne Funktionalitäten verwendeter Software von Drittanbietern. Hierzu zählen: Android Betriebssystem, Android Treiber, Android-APIs, Android Support AppCompat, Android Support Annotations, Android Support Design, Android Support CardView, Android Support Library, AssertJ, Butterknife, IBR-DTN, Google Guava, Google Play Services, Google Location Service, JaCoCo, JUnit, Leakcanary, Robolectric, Robolectric Shadows, SugarORM, SQLite.

2.4 Vorgehen

Der folgende Abschnitt beschreibt die Vorgehensweisen im Hinblick auf die Funktionen der Software. Dazu werden die jeweiligen Techniken, Werkzeuge und Aktivitäten näher erläutert. Für das Testvorgehen wird das V-Modell von Barry Boehm genutzt. Zu Beginn muss die Funktionalität der einzelnen Komponenten getestet werden. Daraufhin folgt ein Integrationstest, welcher die Zusammenarbeit der einzelnen Komponenten überprüft. Um den Abnahmetest mit dem Kunden vorzubereiten, durchläuft das ganze System einen Systemtest, wodurch eine Überprüfung der funktionalen und nichtfunktionalen Anforderungen an dem Gesamtsystem stattfindet.
Sind alle unten aufgeführten Tests durchgeführt und bestanden worden, so ist das Softwareprodukt erfolgreich entwickelt worden.

a) **Komponententest**
Der Komponententest besteht aus dem Black-Box-Test und dem White-Box-Test. Der Black-Box-Test untersucht Komponenten auf korrekte Funktionsweise, ohne hierbei die konkrete Implementation zu kennen. Der White-Box-Test wird dagegen auf Grundlage des Quellcodes durchgeführt und kann so den Code an sich besser testen, sprich die Funktionsweise innerhalb der Komponente. Im Komponententest werden die einzelnen Bestandteile der Software unabhängig voneinander getestet. Dies ermöglicht ein gezieltes Filtern von individuellen Fehlern, welche direkt behoben werden können. Sind die einzelnen Komponenten fehlerfrei, so ist dies auch für die spätere Entwicklung ein Vorteil, da andere auftauchende Fehler besser eingegrenzt werden können. Das Testziel besteht somit darin, dass wir als Entwickler die Fehlerfreiheit der Komponenten aufweisen können.

b) **Integrationstest**
Sind die einzelnen Komponenten fehlerfrei, so ist der Komponententest abgeschlossen und es findet der Integrationstest statt, bei welchem das Bottom-up Verfahren genutzt wird. Bei dem Integrationstest wird mit dem Ziel getestet, Fehler bei der Zusammenarbeit der Komponenten aufzuzeigen. Der Schwerpunkt liegt darin die korrekte Reihenfolge der Einzeltests zu definieren. Bei dem Bottom-up Verfahren werden beispielsweise die Funktionen, Klassen und Module festgelegt und zusammengeführt. Das Zusammenspiel der einzelnen Komponentengruppen wird dann getestet.

c) **Systemtest**
Bei dieser Stufe wird das gesamte System anhand der funktionalen und nichtfunktionalen Anforderungen überprüft, wobei besonders auf die bereits definierten Kriterien im Pflichtenheft, Zuverlässigkeit und auf die Benutzbarkeit eingegangen werden soll. Ziel ist dementsprechend, dass das System die Anforderungen leistet und für eine fehlerfreie Übergabe des Produkts an den Kunden bereit ist. Aus diesem Grund wird eine Simulation durchgeführt, welche auch auf noch eventuell bestehende Fehler hinweisen kann, sodass diese rechtzeitig beseitigt werden können.

d) **Abnahmetest**
Der Abnahmetest beschreibt das Testen der ausgelieferten Software direkt durch den Kunden selbst. Vor allem dient es dazu, dass man überprüft, ob die zwischen Auftraggeber und Auftragnehmer abgesprochenen Leistungen umgesetzt worden sind. Damit die Zufriedenheit des Kunden sichergestellt wird, werden die im Pflichtenheft genannten Abnahmekriterien überprüft. Ist der Abnahmetest erfolgreich bestanden, so kann der Vertrag rechtens abgeschlossen werden.

2.5 Testumgebung

Während der Entwicklung werden zahlreiche Unit Tests erstellt. Verwendet wird dazu *Robolectric*[1]. Dadurch wird kein Smartphone zum Ausführen dieser Tests benötigt. Die Tests können auf jedem Rechner mit Java und der Android SDK ausgeführt werden.

Um die grafische Oberfläche zu testen wird *Espresso*[2] genutzt. Dabei werden wiederholbare Tests erstellt, die auf dem Smartphone ausgeführt werden können.

Zu diesen Tests wird eine Statistik der Testabdeckung durch *JaCoCo*[3] erstellt.

[1] http://robolectric.org/
[2] http://developer.android.com/training/testing/ui-testing/espresso-testing.html
[3] http://eclemma.org/jacoco/

3 Abnahmetest

Ziel dieser Tests ist es, das Produkt durch den Kunden abzunehmen.

3.1 Zu testende Anforderungen

Die zu testenden Anforderungen sind:

Nr	Anforderung	Testfälle	Kommentar
1	Ist die Anwendung lauffähig?	<T100>	
2	<F10>, <F50> Jodelliste wird korrekt angezeigt	<T200>, <T1100>	
3	<F20> Kommentare zu einzelnen Jodeln werden korrekt angezeigt	<T300>	
4	<F30>, <F35> Jodel und Kommentare können bewertet werden. Die Votes werden korrekt abgespeichert.	<T600>, <T700>	
5	<F40> Daten werden korrekt über das IBR-DTN übertragen	<T800>	
6	<F70> Kommentare zu Jodeln können erstellt werden und werden korrekt zugeordnet.	<T500>, <T550>	
7	<F60>, <F50> Jodel können erstellt und mit Lokationsdaten versehen werden.	<T400>, <T550>	
8	<F80> Das Anzeigen von Bildern funktioniert gemäß Funktionsdefinition.	<T250>	
9	<F90> Jodel und Kommentare des Nutzers können abgerufen werden und werden korrekt dargestellt.	<T900>	
10	Mehrsprachiges Layout	<T1000>	

3.2 Testverfahren

Die korrekte Zusammenarbeit der einzelnen Komponenten werden durch Black-Box-Tests geprüft. Dazu werden verschiedene Betriebsumgebungen genutzt.

Die Verteilung der Daten auf anderen Geräten wird durch verschiedene Aktionen auf einem Gerät getestet. Nach kurzer Zeit müssen die Aktionen auf mehreren anderen Geräten zu sehen sein. So muss zum Beispiel ein Jodel, der auf einem Gerät erstellt wird, kurze Zeit später auf anderen Geräten zu finden sein. Analog gilt dies für Votes und Kommentare.

Diese Tests werden in verschiedenen Netzwerken durchgeführt.

Die Benutzeroberfläche wird durch verschiedene Eingaben und korrekter Ergebnisse getestet.

Es wird das Verhalten der Applikation während der einzelnen Testfälle beobachtet und anhand der funktionalen Anforderung auf Richtigkeit geprüft.

3.2.1 Testskripte

Zu Beginn der Entwicklung können Jodel durch ein Skript erstellt werden. So kann das Auslesen und die Darstellung vorhandener Jodel getestet werden, ohne dass die Funktion zur Erstellung von Jodel bereits implementiert worden sein muss.

Zur Abnahme werden *Monkeytests*[1] ausgeführt, die zufällige Eingaben tätigen um die Applikation einem Stress-Test zu unterziehen.

3.3 Testfälle

[1] http://developer.android.com/tools/help/monkey.html

3.3.1 Testfall $\langle T100 \rangle$ - App starten

Ziel

Überprüfen der Lauffähigkeit der Applikation

Objekte/Methoden/Funktionen

Die App wird gestartet.

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn die App startet und nicht sofort mit einer Fehlermeldung beendet wird.

Vorbedingung

keine

Einzelschritte

Eingabe:

1. Die App wird gestartet, z.B. durch tippen auf das App-Symbol im Launcher

3.3.2 Testfall $\langle T200 \rangle$ - Lokale Jodel anzeigen

Ziel

Überprüfen der Anzeige von Jodel und deren Gesamtvoting ausschließlich aus der Umgebung

Objekte/Methoden/Funktionen

Funktionen: <F10>, <F50>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn in der Liste nur Jodel angezeigt werden, die in einem Umkreis von 10km verfasst wurden.

Vorbedingung

Der Standortdienst des Gerätes ist aktiviert und es besteht GPS-Empfang.

Einzelschritte

Eingabe:

1. Zum Startbildschirm navigieren.

Ausgabe:

1. Jodelliste mit entprechenden Ortseinträgen.

Beobachtungen / Log / Umgebung

Beobachtet werden die Ortangaben der angezeigten Jodel um deren Nähe zu bestätigen.

3.3.3 Testfall $\langle T250 \rangle$ - Bild anzeigen

Ziel

Überprüfen der Anzeige von Bildern

Objekte/Methoden/Funktionen

Funktionen: <F80>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn ein Bild beim Gedrückthalten in voller Größe angezeigt und beim Loslassen wieder minimiert wird.

Vorbedingung

Ein Jodel oder Kommentar mit Bild muss vorhanden sein

Einzelschritte

Eingabe:

1. Zur Jodelliste navigieren
2. Einen Bild-Jodel heraussuchen
3. Jodel gedrückt halten
4. Bild ansehen
5. Jodel loslassen

3.3.4 Testfall $\langle T300 \rangle$ - Kommentare anzeigen

Ziel

Überprüfen des Anzeigens von Jodel mit zugehörigen Kommentaren

Objekte/Methoden/Funktionen

Funktionen: <F20>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn in einer seperaten Ansicht der Jodel mit seinen Kommentaren angezeigt wird.

Vorbedingung

Es existiert ein Jodel, der in der Liste angezeigt wird und Kommentare enthält.

Einzelschritte

Eingabe:

1. Zur Jodelliste navigieren
2. Einen Jodel auswählen (durch antippen)

Abhängigkeiten

Abhängig von <T200>, da sonst kein Jodel mit Kommentaren ausgewählt werden kann.

3.3.5 Testfall $\langle T400 \rangle$ - Jodel verfassen

Ziel

Es können Jodel als Text verfasst werden

Objekte/Methoden/Funktionen

Funktionen: <F60>, <F50>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn der Jodel verfasst werden kann und anschließend in der Liste angezeigt wird.

Vorbedingung

Der Standortdienst des Gerätes ist aktiviert und es besteht GPS-Empfang.

Einzelschritte

Eingabe:

1. Zum Startbildschirm navigieren
2. Schaltfläche zum Erstellen von Jodel antippen
3. Jodel in Form von Text verfassen
4. Auf „Senden" tippen

Abhängigkeiten

Abhängig von <T200> zur Feststellung des Testresultates.

3.3.6 Testfall $\langle T500 \rangle$ - Kommentar verfassen

Ziel

Jodel können kommentiert werden

Objekte/Methoden/Funktionen

Funktionen: <F70>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn ein Kommentar zu einem Jodel verfasst werden kann und dieser anschließend in der seperaten Kommentaransicht des Jodels angezeigt wird.

Vorbedingung

Es muss ein Jodel vorhanden sein, der kommentiert werden kann.

Einzelschritte

Eingabe:

1. Jodel antippen, um zur Kommentaransicht zu gelangen
2. Schaltfläche zum Hinzufügen eines Kommentares antippen
3. Kommentar verfassen
4. Schaltfläche zum Senden drücken

Abhängigkeiten

Abhängig von <T200> sowie <T300> zur Feststellung des Testresultates.

3.3.7 Testfall $\langle T550 \rangle$ - Jodel/Kommentar mit Bild verfassen

Ziel

Es können Jodel und Kommentare auch als Bild verfasst werden.

Objekte/Methoden/Funktionen

Funktionen: <F60>, <F50>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn der Jodel als Bild verfasst werden kann und anschließend in der Liste angezeigt wird.

Vorbedingung

Der Standortdienst des Gerätes ist aktiviert und es besteht GPS-Empfang. Des Weiteren muss das Gerät über eine Kamera verfügen.

Einzelschritte

Eingabe:

1. Zum Startbildschirm navigieren
2. Schaltfläche zum Erstellen von Jodel antippen
3. Auf die Kamera-Schaltfläche tippen
4. Bild aufnehmen
5. Auf „Senden" tippen

Abhängigkeiten

Abhängig von <T200>, <T250> zur Feststellung des Testresultates.

3.3.8 Testfall $\langle T600 \rangle$ - Voten

Ziel

Jodel/Kommentar hoch- oder runterstufen

Objekte/Methoden/Funktionen

Funktionen: <F30>, <F35>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn sich das Voting beim Hochstufen des Jodels oder Kommentares um 1 erhöht bzw. beim Runterstufen um 1 verringert.

Vorbedingung

Es muss ein Jodel oder Kommentar zum Bewerten vorhanden sein.

Einzelschritte

Eingabe:

1. Zur Jodelliste navigieren
2. Eine entsprechende Schaltfläche zum Bewerten drücken

Abhängigkeiten

Abhängig von <T200> und eventuell auch von <T300> zur Feststellung des Testresultates.

3.3.9 Testfall $\langle T700 \rangle$ - Löschen aufgrund schlechter Bewertung

Ziel

Jodel/Kommentare mit einem Voting kleiner als -4 werden gelöscht

Objekte/Methoden/Funktionen

Funktionen: <F30>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn kein Jodel/Kommentar eine Bewertung kleiner als -4 hat und bei einer fünften negativen Bewertung automatisch gelöscht wird.

Vorbedingung

Es existiert ein Jodel/Kommentar der bewertet werden kann und ein Voting von -4 hat.

Einzelschritte

Eingabe:

1. Zur Jodelliste navigieren
2. Einen Jodel oder Kommentar mit einer Bewertung von -4 suchen
3. Diesen Jodel/Kommentar negativ bewerten

Abhängigkeiten

Abhängig von <T200> und eventuell auch von <T300> zur Feststellung des Testresultates sowei von <T600>.

3.3.10 Testfall ⟨T800⟩ - Jodel über IBR-DTN verteilen

Ziel

Überprüfung der Kommunikation der Geräte via IBR-DTN

Objekte/Methoden/Funktionen

Funktionen: <F40>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn ein Jodel/Kommentar/Voting an einem Gerät abgeschickt wird und an allen Geräten im selben Netzwerk angezeigt wird. Ebenso muss für ein erfolgreiches abschließen dieses Test gewährleistet sein, dass ein Jodel, Kommentar oder Vote auch von jedem Geräten, das ihn erhalten hat, weitergegeben wird.

Vorbedingung

Die IBR-DTN App muss installiert sein und der Dienst dieser muss ausgeführt werden.

Einzelschritte

Eingabe:

1. Jodel/Kommentar/Voting erstellen, wie im dazugehörigen Testfall beschrieben

Beobachtungen / Log / Umgebung

Es werden weitere Geräten mit DTN-Jodel im selben Netzwerk beobachtet und auf das erscheinen des gesendeten Jodels gewartet.

Besonderheiten

Es werden mehrere Geräte und ein Netzwerk (WLAN) für diesen Test benötigt

Abhängigkeiten

Abhängig von <T400>, <T500> und <T600> zum Erstellen des zu sendenden Inhaltes.

3.3.11 Testfall $\langle T900 \rangle$ - Eigene Aktionen anzeigen

Ziel

Anzeigen von selbst erstellten Posts und Kommentaren sowie abgegebenen Votings

Objekte/Methoden/Funktionen

Funktionen: <F90>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn unter dem Menupunkt "Meine Aktionen" alle Jodel, Kommentare und Bewertungen, die der Nutzer selbst erstellt hat, angezeigt werden. Ausnahmen bilden hier nur die aufgrund von Alter oder geringem Speicherplatz bereits gelöschten Aktionen.

Vorbedingung

Der Nutzer hat einen Jodel, Kommentar verfasst oder eine Bewertung abgegeben.

Einzelschritte

Eingabe:

1. Navigiere zur Ansicht für eigene Aktionen

3.3.12 Testfall $\langle T1000 \rangle$ - Sprache ändern

Ziel

Sprache in der App passt sich dem System an, wenn eine Übersetzung vorhanden ist.

Objekte/Methoden/Funktionen

Kriterien: <RS3>, <RC1>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn die App sich der Systemsprache anpasst, sofern eine entsprechende Übersetzung vorhanden ist.

Vorbedingung

keine

Einzelschritte

Eingabe:

1. Systemsprache ändern
2. DTN-Jodel starten
3. Durch die verschiedenen Ansichten navigieren

Besonderheiten

Dieser Test ist von den Systemeinstellungen abhängig. Wie genau die Systemsprache geändert wird, ist Gerätespezifisch und kann somit hier nicht allgemein wiedergegeben werden.

3.3.13 Testfall $\langle T1100 \rangle$ - Layout drehen

Ziel

Das Layout passt sich horizontaler sowie vertikaler Haltung des Gerätes an.

Objekte/Methoden/Funktionen

Kriterien: <RS1>

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn sich das Layout der App bei Rotation des Gerätes anpasst und weiterhin funktionsfähig ist.

Vorbedingung

Das Gerät erkennt Rotationen und die Bildschirmausrichtung ist nicht fixiert.

Einzelschritte

Eingabe:

1. App starten
2. Zu einer beliebigen Ansicht navigieren
3. Das Gerät nach belieben rotieren

4 Integrationstest

In diesem Kapitel werden die Integrationstests spezifiziert. Es werden Komponenten verknüpft und getestet, ob diese korrekt miteinander funktionieren.

Das Testziel ist alle Schnittstellen der Komponenten auf korrekte Funktionsweise zu überprüfen.

4.1 Zu testende Komponenten

Nr	Komponenten	Testfälle	Kommentar
1	<C10> GUI	T2100, T2200, T2300, T2400	
2	<C20> LocationManager	T2100	
3	<C30> ContentManager	T2200, T2300, T2400, T2500, T2600	
4	<C40> DatabaseHandler	T2500, T2600, T2700, T2800, T2900, T3000	
5	<C50> Communicator	T2700, T2800, T2900, T3000	

4.2 Testverfahren

Es werden wiederholbare Integrationstests geschrieben, die während der Entwicklung und letztendlich zum Abnahmetests nochmal ausgeführt werden. Es wird angestrebt, dass zu jedem Commit alle Tests bestanden werden.

4.2.1 Testskripte

Es wird Robolectric[1] verwendet, um die Integrationstests wiederholt auszuführen. Dazu werden vorher Tests geschrieben, die Knopfdrücke simulieren und somit die einzelnen Komponenten im Zusammenspiel testen.

4.3 Testfälle

[1] http://robolectric.org/

4.3.1 Testfall $\langle T2100 \rangle$ - Komponente <C10> + <C20>

Ziel

Korrekte Darstellung des aktuellen Standortes

Objekte/Methoden/Funktionen

getGPSLocation()

Pass/Fail Kriterien

Aktueller Standort wird in der Toolbar angezeigt.

Vorbedingung

GPS aktiviert und gefunden.

Einzelschritte

Eingaben:

- App starten
- Warten, bis Standort gefunden wurde

4.3.2 Testfall $\langle T2200 \rangle$ - Komponente <C10> + <C30>

Ziel

Korrekte Darstellung mehrere Jodel

Objekte/Methoden/Funktionen

displayJodel(), displayComments(), displayCloseJodels()

Pass/Fail Kriterien

Mehrere Jodel aus der Umgebung sind auf dem Startbildschirm zu sehen.

Vorbedingung

Jodel sind in Datenbank vorhanden

Einzelschritte

App starten, Jodel auf dem Startbildschirm sehen

4.3.3 Testfall $\langle T2300 \rangle$ - Komponente <C10> + <C30>

Ziel

Erstellen von Jodel

Objekte/Methoden/Funktionen

assembleJodel()

Pass/Fail Kriterien

Jodelobjekt wird erstellt.

Vorbedingung

Zur „Jodel erstellen"-Ansicht navigiert

Einzelschritte

Jodel erstellen, Jodel Objekt auslesen.

Besonderheiten

Diese Funktionen erzeugen keine Ausgabe auf dem Bildschirm

4.3.4 Testfall $\langle T2400 \rangle$ - Komponente <C10> + <C30>

Ziel

Erstellen von Votes

Objekte/Methoden/Funktionen

assembleVote()

Pass/Fail Kriterien

Voteobjekt wird erstellt.

Vorbedingung

Jodel in Datenbank vorhanden

Einzelschritte

Bei einem Jodel den Upvote Button drücken, Vote Objekt auslesen.

Besonderheiten

Diese Funktionen erzeugen keine Ausgabe auf dem Bildschirm

4.3.5 Testfall $\langle T2500 \rangle$ - Komponente <C30> + <C40>

Ziel

Speichern und Auslesen von Jodel

Objekte/Methoden/Funktionen

saveJodel(), getJodel()

Pass/Fail Kriterien

Jodel wird in Datenbank gespeichert und von dort ausgelesen

Vorbedingung

Zur „Jodel erstellen"-Ansicht navigieren

Einzelschritte

Jodel erstellen, Jodel aus Datenbank auslesen

4.3.6 Testfall $\langle T2600 \rangle$ - Komponente <C30> + <C40>

Ziel

Speichern und Auslesen von Votes

Objekte/Methoden/Funktionen

saveVote(), getVote()

Pass/Fail Kriterien

Vote wird in Datenbank gespeichert und von dort ausgelesen

Vorbedingung

Vote in Datenbank vorhanden

Einzelschritte

Vote erstellen, Vote aus Datenbank auslesen

4.3.7 Testfall $\langle T2700 \rangle$ - Komponente <C40> + <C50>

Ziel

Jodel aus Datenbank an anderes Gerät schicken

Objekte/Methoden/Funktionen

sendJodel()

Pass/Fail Kriterien

Jodel wird an andere versendet

Vorbedingung

Jodel in Datenbank vorhanden

Einzelschritte

Jodel erstellen

Abhängigkeiten

<T2800>

4.3.8 Testfall $\langle T2800 \rangle$ - Komponente <C40> + <C50>

Ziel

Jodel von anderen Geräten empfangen und in Datenbank abspeichern

Objekte/Methoden/Funktionen

saveJodel()

Pass/Fail Kriterien

Jodel wird in Datenbank gespeichert

Vorbedingung

Jodel von anderem Gerät versendet

Einzelschritte

Jodel empfangen

Abhängigkeiten

<T2700>

4.3.9 Testfall $\langle T2900 \rangle$ - Komponente <C40> + <C50>

Ziel

Vote aus Datenbank an anderes Gerät schicken

Objekte/Methoden/Funktionen

sendVote()

Pass/Fail Kriterien

Vote wird an andere versendet

Vorbedingung

Vote in Datenbank vorhanden

Einzelschritte

Vote erstellen

Abhängigkeiten

<T3000>

4.3.10 Testfall $\langle T3000 \rangle$ - Komponente <C40> + <C50>

Ziel

Vote von anderen Geräten empfangen und in Datenbank abspeichern

Objekte/Methoden/Funktionen

saveVote()

Pass/Fail Kriterien

Vote wird in Datenbank gespeichert

Vorbedingung

Vote von anderem Gerät versendet

Einzelschritte

Vote empfangen

Abhängigkeiten

<T2900>

5 Unit-Tests

In diesem Kapitel werden die Unit-Tests spezifiziert. Es werden Unit-Tests zu Komponenten sowie Klassen und Funktionen zu Unit-Tests zugeordnet.

Das Testziel ist alle zu testenden Methoden auf korrekte Funktionsweise zu überprüfen.

5.1 Zu testende Komponenten

Nr	Komponenten	Testfälle	Kommentar
1	<C1> GUI	T13000, T13600	
2	<C2> LocationManager	T13700	
3	<C3> ContentManager	T13200, T13300, T13500	
4	<C4> DatabaseHandler	T13100, T13400	
5	<C5> Communicator	T13900, T13800	

5.2 Testverfahren

Während der Entwicklung werden zu jeder Änderung Unit-Tests geschrieben, die einzelne Funktionen testen. Es wird angestrebt, dass zu jedem Commit alle Tests bestanden werden.

5.2.1 Testskripte

Zum Ausführen der Unit-Tests wird Robolectric[1] und Android Studio verwendet.

5.3 Testfälle

[1] http://robolectric.org

5.3.1 Testfall $\langle T13000 \rangle$ - Jodel

Ziel

Korrekte Anzeige der Vote Werte.

Objekte/Methoden/Funktionen

Methoden: upVote(), downVote(), vote()

Pass/Fail Kriterien

Ist Erfolgreich, wenn der Jodel den korrekten Vote-Wert besitzt.

Vorbedingung

Ein Jodel muss vorhanden sein.

Einzelschritte Eingaben:

- upvote() ausführen
- Vote-Wert kontrollieren
- downvote() ausführen
- Vote-Wert kontrollieren

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.2 Testfall $\langle T13100 \rangle$ - Jodel

Ziel

Korrektes Abspeichern der Jodel.

Objekte/Methoden/Funktionen

Methoden: save()

Pass/Fail Kriterien

Der Test ist erfolgreich, wenn der Jodel in der Datenbank abgespeichert wurde.

Vorbedingung

Eine Jodel-Instanz muss vorhanden sein.

Einzelschritte

save() ausführen

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.3 Testfall $\langle T13200 \rangle$ - Jodel

Ziel

Korrekte Ausgabe, ob der Jodel ein Kommentar zu einem Jodel ist.

Objekte/Methoden/Funktionen

Methoden: isComment()

Pass/Fail Kriterien

Ist Erfolgreich, wenn ein Kommentar als Kommentar erkannt wird.

Vorbedingung

Ein Jodel und ein Kommentar müssen vorhanden sein.

Einzelschritte

Eingaben:

- Jodel mit isComment() prüfen
- Kommentar mit isComment() prüfen

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.4 Testfall $\langle T13400 \rangle$ - Jodel

Ziel

Korrekte Ausgabe, ob der Jodel von einem selber stammt

Objekte/Methoden/Funktionen

Methoden: isOwnContent()

Pass/Fail Kriterien

Ist Erfolgreich, wenn ein Jodel korrekt als eigener oder fremder Jodel erkannt wird.

Vorbedingung

Ein Jodel ist vorhanden.

Einzelschritte

isOwnContent() wird ausgeführt

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.5 Testfall $\langle T13500 \rangle$ - Vote

Ziel

Korrektes Abspeichern der Votes

Objekte/Methoden/Funktionen

Methoden: save()

Pass/Fail Kriterien

Ist Erfolgreich, wenn der Vote in der Datenbank gespeichert wurde

Vorbedingung

Es ist ein Vote vorhanden.

Einzelschritte

save() wird ausgeführt.

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.6 Testfall $\langle T13600 \rangle$ - Vote

Ziel

Eigenen Vote zu Jodel laden

Objekte/Methoden/Funktionen

Methoden: myVote()

Pass/Fail Kriterien

Erfolgreich, wenn erkannt wurde, ob man einen eigenen Vote für den Jodel hat

Vorbedingung

Vote und Jodel sind vorhanden.

Einzelschritte

myVote() mit entsprechendem Jodel als Parameter aufrufen.

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.7 Testfall $\langle T13700 \rangle$ - JodelPresenter

Ziel

Korrektes Anzeigen der einzelnen Jodel

Objekte/Methoden/Funktionen

Methoden: display()

Pass/Fail Kriterien

Ist Erfolgreich, wenn korrektes Layout mit korrekten Werten ausgegeben wird.

Vorbedingung

Jodel ist vorhanden

Einzelschritte

display() ausführen

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

Besonderheiten

Die Ausgabe des Layouts wird nicht über ein Display kontrolliert, da es auf keinem angezeigt wird. Robolectric simuliert einen Bildschirm und es werden die Texte in diesem anschließend kontrolliert

5.3.8 Testfall $\langle T13800 \rangle$ - JodelPresenter

Ziel

Die Stadt, in der der Jodel erstellt wurde, wird korrekt ermittelt.

Objekte/Methoden/Funktionen

Methoden: getCityFromJodel()

Pass/Fail Kriterien

Erfolgreich, wenn die Stadt mit den im Jodel gespeicherten Koordinaten übereinstimmt.

Vorbedingung

Jodel mit korrekt gespeicherten Koordinaten ist vorhanden.

Einzelschritte

getCityFromJodel() wird mit einem Jodel als Parameter ausgeführt

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.9 Testfall $\langle T13900 \rangle$ - BundleTransformer

Ziel

Jodel wird gemäß des Protokolls transformiert.

Objekte/Methoden/Funktionen

Methoden: transformJodel(), transformInputStream()

Pass/Fail Kriterien

Erfolgreich, wenn Jodel, mithilfe von transformJodel(), gemäß des Protokolls korrekt codiert wurde und mithilfe von transformInputStream() wieder korrekt decodiert wurde.

Vorbedingung

Jodel, der codiert werden soll, ist vorhanden.

Einzelschritte

Eingaben:

- Jodel wird erstellt
- Jodel wird codiert
- Jodel wird decodiert

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

5.3.10 Testfall $\langle T14000 \rangle$ - BundleTransformer

Ziel

Vote wird gemäß des Protokolls transformiert.

Objekte/Methoden/Funktionen

Methoden: transformVote(), transformInputStream()

Pass/Fail Kriterien

Erfolgreich, wenn Vote, mithilfe von transformVote(), gemäß des Protokolls korrekt codiert wurde und mithilfe von transformInputStream() wieder korrekt decodiert wurde.

Vorbedingung

Vote, der codiert werden soll, ist vorhanden.

Einzelschritte

Eingaben:

- Vote wird erstellt
- Vote wird codiert
- Vote wird decodiert

Beobachtungen / Log / Umgebung

Es wird die Ausgabe auf der Konsole beobachtet und darauf geachtet, ob eine Fehlermeldung ausgegeben wird.

6 Glossar

Android Betriebssystem für Mobiltelefone.

App Kurzform für Applikation.

Applikation Wird vorrangig im Kontext von Mobiltelefonen als Begriff für Anwendungen genutzt.

Betriebssystem Zusammenstellung von Programmen, die die Systemressourcen eines Computers verwalten.

Bibliothek Zusammenstellung von Programmstücken zur Weiterverwendung.

Code Für Menschen lesbarer, in einer Programmiersprache verfasster, Text eines Computerprogramms.

Commit Zusammenhängende Änderung an Dateien und Ordnern. Enthalten ist der Autor, die Änderung, der Zeitpunkt der Änderung, eine Nachricht sowie eine zeitliche Einsortierung mit anderen Commits.

Downvote Herunterstufen eines Jodels oder Kommentars

Gradle Programm, welches das Kompilieren eines Java Programms automatisiert.

IBR-DTN Software zur Erstellung einer verzögerungstoleranten Peer-To-Peer Verbindung

Java Eine Programmiersprache.

Jodel Eine Nachricht, die in der geographischen Nähe versendet wird

Javadoc Dokumentationsstandart für Java Code mit Möglichkeit zur Generierung einer Internetseite mit der vollständigen Dokumentation.

Klasse Entität in Objektorientierter Programmierung

Kompilieren Übersetzen von einem Format in ein anderes Format. Üblicherweise wird aus ein Programm von der Programmiersprache in ein ausführbares Format übersetzt.

Lokation Position des Geräts auf der Erde.

MAC Adresse Media-Access-Control-Adresse. Dient zur eindeutigen Identifikation eines Gerätes in einem Rechnernetz.

Methode Funktion innerhalb einer Klasse.

Protokoll (techn.) Eine standardisierte Form von Nachrichten, die Anwendungen untereinander austauschen.

Upvote Hochstufen eines Jodels oder Kommentars

Voten Hoch- bzw. Herunterstufen eines Jodels oder Kommentars