Fachhochschule Köln
Cologne University of Applied Sciences

Campus Gummersbach

Datenanalyse mit Apache Cassandra und mobiler Statistik

Bachelorarbeit

Ausgearbeitet von	Benjamin Werker
	Matrikel-Nr.: 11059843
Vorgelegt an der	Fachhochschule Köln
	Campus Gummersbach
	Fakultät für Informatik und Ingenieruswissenschaften (Fakultät 10)
Im Studiengang	Allgemeine Informatik
Erstprüfer:	Prof. Dr. Heide Faeskorn-Woyke (Fachhochschule Köln)
Zweitprüfer	Prof. Dr. Frank Victor (Fachhochschule Köln)

Kurzzusammenfassung

Die Bachelorarbeit befasst sich mit der Verwendung der NoSQL Datenbank Apache Cassandra. Dabei werden auf der einen Seite die Unterschiede bei Verwendung und Betrieb von Apache Cassandra im Vergleich mit relationalen SLQ Datenbanken und auf der anderen Seite die Aspekte Geschwindigkeit, Ausfallsicherheit und Wiederverwendbarkeit untersucht. Die Verwendung und der Betrieb wird dabei durch die Umsetzung eines Datenimports, damit verbunden ist die Erstellung von entsprechenden Datenmodellen, und der Bereitstellung der Daten für die Darstellung von mobilen Statistiken in Form einer Android App untersucht. Für die Untersuchung der Geschwindigkeit, Ausfallsicherheit und Wiederverwendbarkeit werden zusätzlich zu den durch bereits durch die Umsetzung erhaltenen Ergebnissen noch an den jeweiligen Aspekt angepasste Belastungstest durchgeführt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs/Synonymverzeichnis

Abkürzung/Synonym	Bedeutung
Cluster	Ein Verbund aus mehreren Servern
CQL	Cassandra Query Language
GeoDaten	Geographisch bezogene Daten
INVEST	Independent Negotiable Valuable Estimable Small Testable
JSON	JavaScript Object Notation
Node	Ein Server innerhalb eines Clusters
NoSQL	„No SQL“ oder „Not Only SQL“
SQL	Structured Query Language

1 Einleitung

Die Idee zur Bachelorarbeit „Datenanalyse mit Apache Cassandra und mobiler Statistik“, im Weiteren als Projekt benannt, ist aus dem Umgang mit verschiedenen Apps und Server Konstrukten entstanden und soll einen Vorteil für Anwender ermöglichen, indem mit der Nachlässigkeit der Entwickler gearbeitet wird. Bei der Summe an Daten, welche verschiedene Apps während der Verwendung an den Server senden, handelt es sich nicht immer um triviale Daten, so verwenden einige Apps geographisch bezogene Daten (GeoDaten) um bestimmte Interaktionsmöglichkeiten wie zum Beispiel Werbung, Gutscheine, Eintrittskarten oder Spiele an die Position des Benutzers zu fixieren. Diese gesammelten GeoDaten können dann wiederum verwendet werden um ein Profil vom Verhalten des Benutzers anzulegen. Ein solches Profil kann auf Basis wahrscheinlich gesammelter Daten zum Beispiel der Tagesablauf inklusive des Arbeitsweges und eventueller Verweildauer im abendlich besuchten Restaurant sein.

Meist ist die Bereitstellung dieser Daten an den Entwickler der App vom Benutzer bei der Installation oder beim ersten Start der App durch die Zustimmung zu den Allgemeinen Geschäftsbedingungen erlaubt worden. Viele Nutzer wissen aber größtenteils nicht wie unsicher viele Entwickler mit diesen Daten umgehen. So kann durch Nachlässigkeit oder einfaches Unwissen des Entwicklers mit wenig Aufwand der komplette Datenverkehr eines Benutzers abgefangen und gesammelt werden.

Genau an diesen Punkt setzt das Projekt an und versucht die Menge der gesammelten Daten sortiert zu speichern, auszuwerten und für den Benutzer aufbereitet wieder zur Verfügung zu stellen. Bei der für dieses Projekt genutzten App handelt es sich um die von Niantic Incorporated[1]. entwickelte App Ingress[2] (im weiteren Verlauf nur App genannt)in der eine Unmenge an geographisch bezogene Serverkommunikation anfällt, welche dem Anwender nicht direkt zur Verfügung steht.

Dabei wird zum Speichern der Daten ein Apache Cassandra Cluster verwendet, welcher neben dem Einsatz zur Speicherung und Abfrage von Daten auch unter den Aspekten Geschwindigkeit, Ausfallsicherheit und Wiederverwendbarkeit betrachtet wird.

[1] Niantic Labs sind als ein internes Startup von Google gestartet. (Niantic Labs, 2015)
[2] Webpräsenz von Ingress (Niantic Labs, 2015)

1.1 Ziel

Das Projekt befasst sich mit der Datenverarbeitung (aus Fremdquellen[3]) und soll die Verwendbarkeit von Apache Cassandra aufweisen. Bei der Verarbeitung der Daten werden zwei wesentliche Punkte behandelt, das Analysieren, Einlesen und Speichern der Daten in einem Apache Cassandra Cluster und die Abfrage der hinterlegten Daten zur weiteren Verwendung. Dabei wird neben der Umsetzung ein zentraler Punkt die Auswertung der von Apache Cassandra in den Vordergrund gestellten Aspekte Geschwindigkeit und Ausfallsicherheit sein. Die Wiederverwendbarkeit von Apache Cassandra sollte im Hinblick auf den Einsatz für andere Projekte oder Erweiterungen auch untersucht werden. Für die grafische Darstellung der verarbeiteten Daten wird abschließend eine Android App entwickelt.

1.1.1 Produktperspektive

Das Projekt soll die zur Verfügung gestellten Daten aufbereiten, für die weitere Verwendung bereitstellen und benutzerfreundlich darstellen. Dabei sind alle Teilprozesse unabhängig voneinander zu entwickeln um einen Austausch zu einem späteren Zeitpunkt zu gewährleisten. Die Verwendbarkeit des abschließend entwickelten Produktes durch Dritte und über den Umfang dieses Projektes hinaus stellt dabei einen weiteren Aspekt der Zielsetzung dar.

1.1.2 Einsatzkontext

Für den Betrieb des Projektes sind neben einem Apache Cassandra Cluster, ein Datenverarbeitungsserver und ein Webserver zur Bereitstellung der verarbeiteten Daten notwendig. Der Aufbau und die Anzahl der Server innerhalb des Apache Cassandra Clusters kann entsprechend der zu verarbeitenden Daten skaliert werden.

Im Kontext des Projektes kann ein separater Datenverarbeitungsserver aufgrund einer beschränkten Datengrundlage vernachlässigt werden und durch einmaligen Import der Daten ersetzt werden. Die grundlegenden Funktionen für den Datenimport sollten dabei dennoch für den Einsatz in einem später zu verwendenden Server vorbereitet werden.

Für die Darstellung der verarbeiteten Daten und Nutzung der Android App wird ein Gerät mit Android Betriebssystem ab Version 4.4 (KitKat) benötigt. Für die

[3] Die Daten wurden nicht im Rahmen des Projektes erstellt oder gesammelt sondern nur für die Durchführung des Projektes von der Internetcommunity „Enlightened Köln“ bereitgestellt. (Ingress Enlightened Köln, 2015)

Bereitstellung der Android App soll um den Nutzerkreis einzuschränken die Verteilung als Alpha/Beta Version über den Google Play Store verwendet werden.[4]

1.2 Funktionale Anforderungen

Der mögliche Funktionsumfang wird in Rücksprache mit den Benutzern und im Rahmen des Projektes reduziert und kann nach Abschluss beliebig erweitert werden. Die für das Projekt damit relevanten Funktionalitäten, Daten, Schnittstellen und eventuelle Anwendungsprofile werden im Weiteren beschrieben.

1.2.1 Funktionalitäten

Bei der Definition der Funktionen kann zu Gunsten eines besseren Verständnisses eine Unterscheidung zwischen den Punkten Datenerfassung, Datenverarbeitung und Datendarstellung gemacht werden.

1.2.1.1 Datenerfassung

Auf Grundlage der zur Verfügung gestellten Datenpakete soll ein strukturierter Import ermöglicht werden. Dabei werden die unveränderten Datensätze als Ausgangspunkt angesehen und müssen in ein für die weiteren Funktionen verwendbares Format gebracht werden. Dies soll durch das genaue Erfassen und Bestimmen von einzelnen Identifikations- und Informationsmerkmale für die jeweiligen Datenpakete ermöglicht werden.

1.2.1.2 Datenverarbeitung

Die für die Verarbeitung vorbereiteten Datensätze müssen für die späteren Anwendungsfälle vorgesehenen Strukturen übernommen werden. Dazu müssen die verschiedenen Verwendungsarten bestimmt und anhand der Daten definiert werden. Hierzu sind die späteren Darstellungsarten zu beachten. Aus diesen geht hervor, dass eine Einteilung der jeweiligen Daten auf Basis von verschiedenen Ereignissen und der Gruppierung nach Zeit und Ort notwendig sind.

1.2.1.3 Datendarstellung

Die für die Statistiken notwendigen Daten müssen für die Darstellung auf einem Android Gerät über einen Server bereitgestellt werden. Bei der Abfrage der Daten soll

[4] Eine Alpha/Beta Version taucht nicht für alle Nutzer des Google Play Stores auf sondern nur für eingeladene Teilnehmer des jeweiligen Tests. (Google, 2015)

im Rahmen des Projektes auf eine ausführliche Authentifikation zu Gunsten einer vereinfachten Datenabfrage für die einzelnen Anwender verzichtet werden.

1.2.2 Daten

Die notwendigen Daten wurden unter Zustimmung der Testkandidaten innerhalb eines Zeitraums von 2 Wochen gesammelt und liegen im JSON Format vor. Dabei handelt es sich um genau jenes Format welches zur Kommunikation zwischen der App und dem Server verwendet wurde. Eine weitere Anpassung bzw. Anonymisierung ist aufgrund der von der App gewählten Benutzerkennungsverschlüsselung nicht mehr notwendig.

1.2.3 Schnittstellen

Für die Kommunikation zwischen den einzelnen Servern, Apache Cassandra Cluster und Anwendungslogikservern, muss eine Netzwerkverbindung bestehen.

Für die Darstellung der Statistiken muss eine Netzwerkverbindung zwischen dem Server mit den Daten und dem Android Gerät bestehen.

1.2.4 Anwendungsprofile

Bei der Unterscheidung der möglichen Anwender muss man zwei generelle Fälle in Betracht ziehen. Der erste Fall wäre der Nutzer welcher die für die Statistik relevanten Daten bereitgestellt hat und diese nun für seine eigene Nutzung wieder abruft. Ein Nutzer der eine Statistik über den gesamten Datenbestand abruft würde als zweiter Fall definiert werden müssen, da dieser nicht unbedingt Daten bereitgestellt hat, aber auf die Ergebnisse zugreifen kann.

1.3 Nicht funktionale Anforderungen

Das Projekt definiert neben den üblichen Anforderungen der Parallelnutzung, Zuverlässigkeit und Ausfallsicherheit noch zusätzliche Anforderungen die im Folgenden erläutert werden.

1.3.1 Effizienz

Sowohl bei der Speicherung der Daten als auch beim Abruf muss darauf geachtet werden, dass die Transfermengen optimiert werden. Dies geschieht auf der einen Seite durch die Verwendung eines für Apache Cassandra geschriebenen Treiber der eine

automatische Optimierung der Übertragung gewährleistet und auf der anderen Seite durch die für die Kommunikation im Internet üblichen GZIP Kompression[5].

1.3.2 Erweiterbarkeit

Nach dem Ende des Projektes sollen verschiedenen Komponenten über die für das Projekt definierten Vorgaben hinaus erweitert werden können. Hierfür werden sämtliche Komponenten in einzelnen logischen Teilblöcken entwickelt und ermöglichen somit ohne Anpassung anderer Komponenten einen einfachen Austausch, dieses Vorgehen entspricht dem Modularen Programmierparadigma und wird in verschiedenen Fachbüchern öfters aufgegriffen.

1.3.3 Benutzerfreundlichkeit

Die Android App muss eine einfache Bedienung für den Anwender garantieren. Dies wird unter anderem durch ein vereinfachtes Authentifizierungsverfahren garantiert, welche nur die Eingabe des zu verwendenden Benutzernamen vorsieht und führt automatisch zu den hinterlegten Daten. Der Wechsel zwischen den einzelnen verfügbaren Statistiken wird durch ein für Android übliches Navigationsmodel, dem Drawer[6], umgesetzt.

1.4 Technische Anforderungen

Man muss zwischen den datenverarbeitenden und datendarstellenden Funktionen Unterscheiden. Die datenverarbeitenden Funktionen werden plattformunabhängig für die Java Runtime[7] entwickelt und können somit auf unterschiedlichsten Plattformen laufen solange eine voll funktionsfähige Java Runtime vorhanden ist.

Für die darstellenden Funktionen ist eine Android App vorgesehen. Diese wird für Android 4.4, alias KitKat, entwickelt und sollte trotz der Fragmentierung der Android Versionen zum Zeitpunkt des Projektes eine Verteilung auf ca. 60%[8] aller Android fähigen Geräten ermöglichen. Die Einschränkung in der Kompatibilität wurde durch den Mehraufwand für die Entwicklung für früher Versionen festgelegt. Eine Erweiterung der Kompatibilität auf ältere Versionen ist im Anschluss des Projektes durch aus denkbar.

[5] RFC zur GZIP-Kompression. (Internet Engineering Task Force (IETF), 2012)
[6] Drawer Konzept von Google. (Google, 2015)
[7] Java Runtime. (Oracle, 2015)
[8] Android Versionsfragmentierung. (Google, 2015)

1.5 Projektabnahmekriterien

Für eine erfolgreiche Abnahme sind die Erstellung einer Dateninfrastruktur, bestehend aus einem Speicherungs- und Abfragesystem, und eine funktionsfähige auslieferungsfertige Android App notwendig. Das abgeschlossene Projekt wird entsprechend der jeweiligen Module mit einer Anleitung für den Betrieb und die Ausbaumöglichkeiten in Form eines Git Repository abgeliefert. Die Abnahme wird durch einen Testlauf der App durch die Anwender, welche die Daten bereitgestellt haben, durchgeführt.

2 Organisatorisches

Um eine bessere zeitliche Einteilung für den Ablauf des Projektes zu ermöglichen wurde anhand der abzuschließenden Aufgaben ein Projektplan erstellt. Dieser wurde bereits unter Berücksichtigung eventueller Verzögerungen der einzelnen Teilaufgaben kalkuliert und sollte eine genaue zeitliche Übersicht über die einzelnen zu erreichenden Meilensteine liefern.

2.1 Ablaufdiagramm

In Abbildung 1 - Ablaufdiagramm sind die verschiedenen Abschnitte des Projektes mit den jeweiligen zeitlich intensivsten Aufgaben dargestellt. Die jeweiligen Abschnitte sind dabei immer abhängig von dem vorhergehenden und können erst nach Abschluss dessen bearbeitet werden.

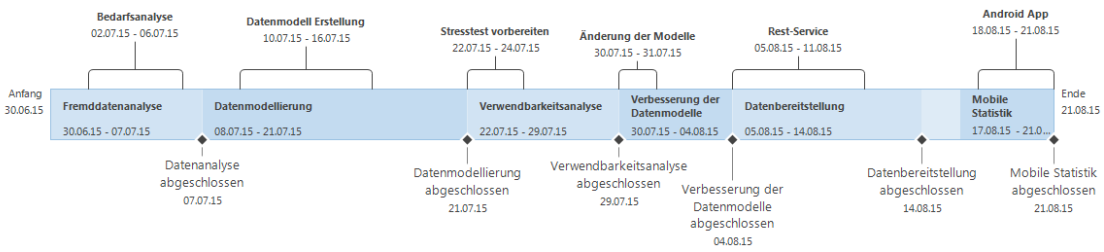

Abbildung 1 - Ablaufdiagramm

Bei Betrachtung der tabellarischen Auflistung aller Aufgaben [Tabelle 1] ist diese Abhängigkeit der einzelnen Abschnitte noch wesentlich deutlicher zu erkennen. Daraus ergibt sich, dass bei nur minimaler Verzögerung einzelner Abschnitte diese Auswirkungen auf den Verlauf des kompletten Projektes haben könnten.

Aufgaben	geschätze Dauer	Anfang	Ende
Fremddatenanalyse	**6 Tage**	**30.06.2015**	**07.07.2015**
Strukturierung	2 Tage	30.06.2015	01.07.2015
Bedarfsanalyse	3 Tage	02.07.2015	06.07.2015
Vorb. Datenimport	1 Tag	07.07.2015	07.07.2015
Datenanalyse abgeschlossen			07.07.2015
Datenmodellierung	**10 Tage**	**08.07.2015**	**21.07.2015**
Datentyp analyse	2 Tage	08.07.2015	09.07.2015
Datenmodell Erstellung	5 Tage	10.07.2015	16.07.2015
Datenimport	3 Tage	17.07.2015	21.07.2015
Datenmodellierung abgeschlossen			21.07.2015
Verwendbarkeitsanalyse	**6 Tage**	**22.07.2015**	**29.07.2015**
Stresstest vorbereiten	3 Tage	22.07.2015	24.07.2015
Stresstest	1 Tag	27.07.2015	27.07.2015
Auswertung	2 Tage	28.07.2015	29.07.2015
Verwendbarkeitsanalyse abgeschlossen			29.07.2015
Verbesserung der Datenmodelle	**4 Tage**	**30.07.2015**	**04.08.2015**
Änderung der Modelle	2 Tage	30.07.2015	31.07.2015
Änderung der Importe	2 Tage	03.08.2015	04.08.2015
Verbesserung der Datenmodelle abgeschlossen			04.08.2015
Datenbereitstellung	**8 Tage**	**05.08.2015**	**14.08.2015**
Rest-Service	5 Tage	05.08.2015	11.08.2015
Datenabfragen	2 Tage	12.08.2015	13.08.2015
Optimierung	1 Tag	14.08.2015	14.08.2015
Datenbereitstellung abgeschlossen			14.08.2015
Mobile Statistik	**5 Tage**	**17.08.2015**	**21.08.2015**
API Client	1 Tag	17.08.2015	17.08.2015
Android App	4 Tage	18.08.2015	21.08.2015
Mobile Statistik abgeschlossen			21.08.2015

Tabelle 1 - Auflistung der einzelaufgaben des Projektablaufs

2.2 Projektumfeld

Das Projekt wird technisch von der Widgetlabs GmbH[9] (im Weiteren nur Widgetlabs genannt) betreut und als Open Source Projekt für die weitere Verwendung an die Bereitsteller der Quelldaten übergeben. Die technische Betreuung durch Widgetlabs umfasst die Bereitstellung der notwendigen Infrastruktur und der fachlichen Kompetenzen in den Bereichen Apache Cassandra und mobiler Client Lösungen.

[9] (Widgetlabs GmbH, 2015)

3 User Cases

Es wurden in Zusammenarbeit mit den Benutzern unterschiedliche Anwendungsfälle entwickelt. Dabei wurde darauf geachtet, dass diese in Anlehnung an das von dem Softwarearchitekten Bill Wake entworfene INVEST[10] Prinzip erstellt wurden, um sowohl für die Entwickler als auch für die Benutzer einen Mehrwert aus den erarbeiteten Anwendungsfällen zu ziehen. Hierzu wird für den Umfang des Projektes auf die zeitliche Einschätzung für die Umsetzung der einzelnen Fälle verzichtet, da dies entsprechende des Projektplans, siehe dazu Abschnitt 2 Organisatorisches, bereits eingeplant ist.

Eine enge Kopplung zwischen den Anforderungen an die Daten und die Datenbankmodellierung ist aufgrund der relationslosen Nutzung von Apache Cassandra dringend notwendig und wird schon mit den Informationen aus den Anwendungsfällen festgelegt. Dementsprechend werden aus den entworfenen Anwendungsfalldiagrammen jeweils Datenmodelle für die spätere Verwendung im Projekt entwickelt.

3.1 INVEST Prinzip

Bei der Entwicklung von Softwareprodukten ist eine gute Abstimmung auf die Anforderungen des Benutzers im Vorfeld schon äußerst wichtig, da eine Änderung der Anforderungen zu einem späteren Zeitpunkt im immer einen Mehraufwand erzeugt. INVEST bietet hierfür einen sowohl für die Entwickler als auch für die Benutzer einfach verwendbares Prinzip.

Durch die Festlegung, dass Anwendungsfälle unabhängig voneinander gestaltet werden ist eine Umsetzung des jeweiligen Falles unabhängig von den Anderen möglich. Dabei sollte wiederum darauf geachtet werden, dass definierte Anwendungsfälle nicht jedes Detail bezüglich der Umsetzung beinhalten sondern einen abstrahierten Überblick bieten und die jeweiligen Details, wie zum Beispiel eventuelle Umsetzungs- oder Testabläufe, erst im Verlauf der Entwicklung in Zusammenarbeit mit dem Benutzer ausgeprägt werden. Es ist dabei unbedingt notwendig, dass ein so erarbeiteter Anwendungsfall einen Nutzen für den Anwender hat. Der Entwickler kann diesbezüglich Bedenken äußern, aber nur um den Benutzer dahingehend den Eindruck zu vermitteln, dass diese Bedenken auch relevant für den jeweiligen Fall sind. So wird es keinen Sinn machen, dem Benutzer eine Aufteilung der Anwendungsfälle im

[10] Ursprüngliche Erwähnung des INVEST Prinzip (Wake, 2003)

Hinblick auf eine spätere Entwicklungsstruktur, wie zum Beispiel die Trennung von Netzwerk und Persistenz Ebenen, da dies für den Anwender keinen direkten Nutzen erzeugt.

Um den Anwender einen weiteren Nutzen aus den Anwendungsfällen zu ermöglichen sollten die jeweiligen Fälle entsprechend ihres Aufwandes eingeschätzt werden. Dabei ist es nicht wichtig eine genau Schätzung abgeben zu können, sondern lediglich eine Einschätzung die es ermöglicht den Aufwand und Zeitpunkt für die spätere Umsetzung zu planen.

Gut definierte Anwendungsfälle sollten klein gehalten werden, dabei sollte der Aufwand zur Umsetzung meist nicht mehr als ein paar Wochen benötigen. Größer definierte Fälle tendieren öfters dazu, dass es schwer wird zu erkennen worauf der Focus in dem entsprechend Fall liegt. Dies würde unter anderem eine Verschlechterung der Aufwandsschätzung verursachen und eventuell beim Benutzer den Eindruck hinterlassen, dass die Aufgaben nicht verstanden worden sind.

Der Benutzer sollte die Anwendungsfälle dabei so gut verstehen, dass es ihm möglich ist für den jeweiligen Fall eine entsprechende Testbedingung definieren zu können. Dies ermöglicht dem Entwickler während der Umsetzung durchgehend zu prüfen ob die entsprechenden Testbedingungen erfüllt sind. Das INVEST Prinzip setzt auf eine wiederholte Bearbeitung der Anwendungsfälle um Unklarheiten und eventuell in früheren Phasen übersehene Aspekte frühzeitig zu erkennen und zu beseitigen.

3.2 Abfrage von zeitlich bezogenen Daten

Das Diagramm, dargestellt in Abbildung 2 - Abfrage von zeitlich bezogenen Daten, zeigt die allgemeine Abfrage eines Benutzers nach seinen Daten auf Basis einer zeitlichen Zusammenfassung. Dabei war es den Benutzer wichtig, dass eventuelle Zeitintervalle selbst bestimmt werden können. Zum besseren Verständnis wurde mit dem Benutzer ein beschreibender Satz für das Anwendungsfalldiagramm erstellt, welcher als Referenz für die spätere Umsetzung dient. Der Satz lautet „Als Benutzer möchte ich meine Statistiken nach Tag/Monat auswählen können um eine Tendenz für jeden Zeitraum zu erkennen“.

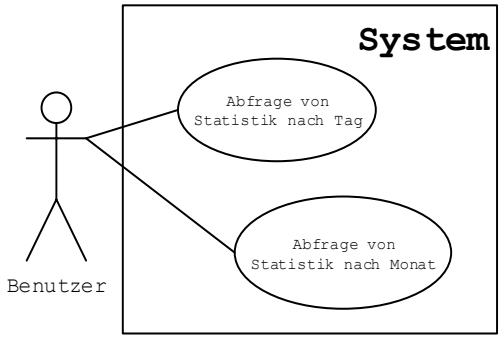

Abbildung 2 - Abfrage von zeitlich bezogenen Daten

3.3 Abfrage von Benutzer bezogenen Daten

Die Benutzer sind über die reine Statistik hinaus weiterhin an einem zusätzlichen Anwendungsfall interessiert, der für den jeweiligen Benutzer spezifisch und Zeit unabhängig Daten bereitstellt. Dies ist im Anwendungsfall Diagramm Abbildung 3 - Abfrage von Benutzer bezogenen Daten dargestellt.

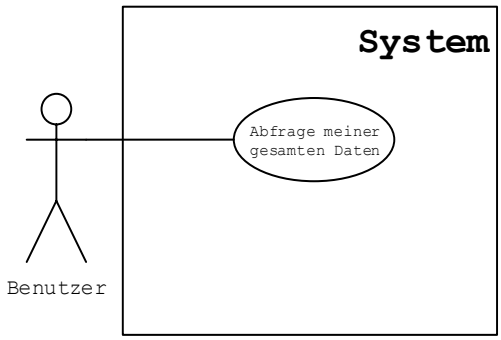

Abbildung 3 - Abfrage von Benutzer bezogenen Daten

Das Anwendungsfalldiagramm wird am ehesten mit dem Satz „Als Benutzer möchte ich alle meine Daten auf einmal erhalten um mir einen genauen Überblick verschaffen zu können“ erklärt, und verdeutlicht noch einmal welchen Mehrwert die Benutzer erwarten.

4 Technische Umsetzung

Bei der Umsetzung muss sich an einige technische Rahmenbedingungen gehalten werden, da, wie im Abschnitt Funktionale Anforderungen bereits definiert, unter anderem die Wartbarkeit und Erweiterbarkeit des Projekts über den Verlauf dieser Arbeit hinaus gegeben sein muss. Dabei ist ebenso die Verwendung bestehender Infrastrukturen zu berücksichtigen. Bei der Analyse und Aufbereitung der Daten wird sich entsprechend der aus den Anwendungsfällen hervorgehenden Anforderungen auf vordefinierte Bereiche konzentriert, so werden Statistiken für wöchentliches, monatliches und jährliches Verhalten benötigt, aber auch Darstellungen über den gesamten Zeitraum.

4.1 Rahmenbedingungen

Bei dem Verwendeten Datenbanksystem handelt es sich um die NoSQL Datenbank Apache Cassandra. Diese hat eine belebte Vergangenheit[11] mit ihrem Wurzeln bei Facebook, Amazon Dynamo und Google BigTable und biete eine Lösung für das Managen von großen Datenmengen über eine Verteilte Infrastruktur. NoSQL Datenbanken verwenden für die Interaktion meist eine eigene definierte Sprache und Syntax. Apache Cassandra ist dahingehend keine Ausnahme und verwendet für die Interaktion CQL, welche aber viele Parallelen zur SQL aufweist.

Die vorhandene Infrastruktur beherbergt augenblicklich einen Cluster aus 6 Apache Cassandra Nodes. Bei der Umsetzung des Projektes sollten diese zur Speicherung der Daten verwendet werden. Die Bereitstellung der aufbereiteten Daten wird über einen REST-Server geschehen, da nicht nur die Daten für die Statistiken der Benutzer, sondern auch die Daten für die Einbindung in Google Maps auf diese Art ausgeliefert werden können.

4.1.1 Architektur

Apache Cassandra setzt beim Aufbau der Architektur im Gegensatz zu vielen anderen Datenbanksystemen auf eine sogenannte „Masterless“ oder dezentralisierten Architektur[12], dies bedeutet, dass alle Server innerhalb eines Apache Cassandra Clusters gleichwertig zu behandeln sind. Abfragen können von jedem Server im Cluster angenommen und zu den jeweils zuständigen Servern delegiert werden.

[11] Vergangenheit von Apache Cassandra (Planet Cassandra, 2015)
[12] Dezentralisierte Architektur. (Apache Cassandra, 2015)

4.1.2 Gemeinsamkeiten und Unterschiede von SQL und CQL

Einer der wesentlichen Vorteile bei der Migration von einem relationalen Datenbanksystem, welches SQL zur Abfrage, Manipulation und Definition verwendet, ist die von Apache Cassandra verwendete Cassandra Query Language (CQL[13]). Diese ist vom Aufbau und der Struktur stark an SQL angelehnt und in vielen Fällen identisch

```
USE myDatabase;

/* Erstellen von Tabellen */
CREATE TABLE IF NOT EXISTS table1 (id INT PRIMARY KEY);

/* Verändern von Tabellen /*
ALTER TABLE table1 ADD col1 INT;

/* Erstellen von Index */
CREATE INDEX iCol1 ON table1 (col1);

/* Erstellen von Datensätzen */
INSERT INTO table1 (id, col1) VALUES (1, 7);

/* Abfrage von Datensätzen */
SELECT * FROM table1 WHERE id = 1;

/* Zählen von Datensätzen */
SELECT COUNT(*) FROM table1;

/* Löschen von Datensätzen */
DELETE FROM table1 WHERE id = 1;
```

Ein markanter Unterschied ist bereits in der Definition von Datenbanken, bei Apache Cassandra als Keyspace definiert, zu erkennen. Apache Cassandra benötigt für die Erstellung einige zusätzliche Angaben, wie die zu verwendende Replikationsstrategie und den Replikationsfaktor, dies wird im Abschnitt 7.1 Apache Cassandra Cluster genauer beschrieben.

```
/* Erstellen eines neuen Keyspace in CQL */
CREATE KEYSPACE database1 WITH replication =
   {'class': 'SimpleStrategy', 'replication_factor': 1};

/* Erstellen einer neuen Datenbank SQL */
CREATE DATABASE database1;
```

Einer der wohl am ehesten bei der Verwendung von CQL auffallenden Unterschiede ist das eigentliche Fehlen einer Unterscheidung von Insert und Update. Die CQL bildet

[13] CQL Dokumentation. (Apache Cassandra, 2011)

beide aus der SQL bekannten Funktionen syntaktisch ab, jedoch wird sowohl bei einem Insert als auch bei einem Update ein nicht vorhandener Datensatz erstellt oder ein vorhandener Datensatz mit den neuen Werten überschrieben.

Der jedoch größte Unterschied zwischen CQL und SQL ist wohl das komplette fehlen von relationalen Bezugsmöglichkeiten wie JOIN, GROUP BY oder FOREIGN KEY. Generell wird im Zusammenhang mit Apache Cassandra auch vermehrt der Merksatz „Writes are cheap, so write everything the way you want to read it.“[14], welcher darauf Anspielt, dass in Apache Cassandra Schreibzugriffe wesentlich weniger Zeit- und Rechenintensiv sind als Lesezugriffe. Dementsprechend sind viele Lesezugriffe um Daten aus unterschiedlichen Tabellen zusammenzufügen aufwendiger als wenn alle Daten für diesen Lesezugriff in einer Tabelle geschrieben wurden. Dies wird zu einer Denormalisierung und doppelter Datenhaltung führen da je nach Lesezugriff unterschiedliche Tabellen öfters den identischen Datensatz enthalten können.

4.2 Datenbereitstellung

Bei der Kommunikation zwischen Clients und Server wird das sich für Webserver etablierte „Representational State Transfer“ (REST[15]) Programmierparadigma verwendet. Dies bietet unter anderem die Möglichkeit sowohl auf Client als auch auf Server Seite auf bereits bestehende Bibliotheken für den Aufbau und Nutzung von REST-Architekturen zu setzen.

4.3 Datenmodelle

Dem Verhalten von Apache Cassandra geschuldet wird bei der Speicherung der Daten entgegen der in relationalen Datenbaken üblichen Normalisierung auf Duplikate und somit doppelter Datenhaltung gesetzt. Folglich werden die Datenmodelle anhand des späteren Verwendungszwecks erstellt bzw. der für die Analyse optimierten Datenabfragen.

Eine fehlerhafte Definition der Datenmodelle würde sich später unter anderem negativ in der Auslastung der Apache Cassandra Nodes zeigen, siehe dazu auch den Abschnitt Anlegen von Schlüsselspalten.

[14] (DataStax, 2015)
[15] Ursprünglich in der Disertation von Dr. Roy Fieldings beschrieben. (Fielding, 2000)

4.3.1 Datenmodell für Statistiken

Die statistisch auswertbaren Daten müssen beim Anlegen der Datenmodelle entsprechend nach Zeitintervallen, im ersten Ansatz nach Tag bzw. Monat, gruppiert werden. Dies ermöglicht eine bessere Verteilung der Daten innerhalb des Apache Cassandra Clusters und führt zu einer gleichbleibenden Auslastung. Außerdem können Datensätze somit entsprechend der vordefinierten Zeitintervalle abgefragt werden.

4.3.2 Datenmodell für Google Maps

Die spätere Darstellung der Daten auf Google Maps legt zwingend fest, dass die Datensätze entsprechend die Informationen bezüglich Längen- und Breitengrade enthalten. Zusätzlich dazu müssen die Datenmodelle die Unterscheidung zwischen den verschiedenen Interaktionsmöglichkeiten definieren.

5 Datenanalyse

Beim Aufbau der späteren Infrastruktur und den notwendigen Datenmodellen ist eine gründliche Analyse der zugrundeliegenden Daten notwendig. Hierzu wurden die bereitgestellten Datensätze im Hinblick auf die spätere Verwendung auf Identifikationsmerkmale, mögliche Gruppierungen und verwertbaren Informationen untersucht[16]. Für die Analyse wurden insgesamt 4 unterschiedliche Ereignisse festgelegt.

5.1 Ereignis 1 und 2

Für die Zusammensetzung der Datensätze des ersten und zweiten Ereignisses sind insgesamt 11 unterschiedliche Datenpakete zu verarbeiten. Aufgrund der zugrundeliegenden Datenstruktur ist für den überwiegenden Datenbestand ein festgesetzter Wert für 10 Datenpakete feststellbar, jedoch sind für einen kleinen Zeitraum andere Werte erfasst. Diese Anomalie im Datenbestand lässt sich am ehesten mit einer durch den Hersteller gestarteten Werbeaktion oder ähnlich vergleichbaren zeitlich begrenzten Änderungen erklären.

Um eine Verfälschung der Statistiken nicht durch eine unvorhersehbare Schwankung der Werte zu riskieren wird der genau übermittelte Wert gespeichert. Die jeweiligen Werte für das erste oder zweite Ereignis werden entsprechend des jeweiligen Datenpakets für einen Import abstrahiert um eventuelle neue Datenpakete einfacher erkennen und klassifizieren zu können.

5.2 Ereignis 3 und 4

Bei der Extraktion der Daten für das dritte und vierte Ereignis sind wesentlich weniger Datenpakete notwendig, da diese entsprechend der Datensätze nur durch ein paar wenige Interaktionen in der ursprünglichen App ausgelöst werden. Die Ereignisse werden später nicht zeitlich gruppiert dargestellt weshalb bei der Analyse ein größerer Focus auf die richtige Verarbeitung der GeoDaten gesetzten werden konnte.

[16] Die Untersuchung ist an die von Usama Fayyad definierte Methodik „From Data Mining to Knowledge Discovery in Databases“ (Fayyad, Piatetsky-Shapiro, & Smyth, 1996) zur Datenanalyse angelehnt. Dabei werden die Vorgehensweisen für die Clusteranalyse, Ausreißerkennung und Klassifikation angewandt.

6 Datenmodelle

Die Erstellung der Datenmodelle für Apache Cassandra gestaltet sich entgegen der bisher für relationale Datenbanken auf Beachtung der Normalformen fokussierten Datenmodelle etwas anders und benötigt dadurch eine andere Vorgehensweise. Jedem Datenmodell liegt später genau eine Anfrage zugrunde und kann dadurch zur mehrfachen Verwendung nahezu identischer Datenmodelle führen.

6.1 Anlegen von Schlüsselspalten

Bei der Anfertigung der Datenmodelle ist auf die von Apache Cassandra für die verteilte Speicherung[17] und die spätere Abfrage von Datensätzen vorgesehene Verwendung der Primär-/Partitionsschlüssel und der „Clustering Columns“ zu achten. Der Partitionsschlüssel, welcher in einer Tabelle mit nur einer Schlüsselspalte identisch mit dem Primärschlüssel wäre. ist dabei zuständig für die Verteilung innerhalb des Clusters. Die so genannten „Clustering Columns“ sind Spalten die zur Sortierung und Eingrenzung von Abfragen innerhalb einer Partition verwendet werden können.

Bei der Abfrage von Daten ist immer der komplette Partitionsschlüssel anzugeben, im Falle eines aus mehreren Spalten bestehender Partitionsschlüssel wären auch alle definierte Partition Spalten notwendig. Für eine weitere Einschränkung der Abfrage können die definierten „Clustering Columns“ verwendet werden, dabei ist darauf zu achten, dass diese nur in der durch die Tabellendefinition angegebenen Reihenfolge verwendet werden können.

6.2 Chebotko Diagramm

Für die Darstellung von logischen Datenmodellen wird im Umfeld von Apache Cassandra vermehrt das von dem „Datastax Solution Architect“ Artem Chebotko[18] entwickelte Chebotko Diagramm verwendet. Tabellen werden als Rechtecke definiert und Spalten werden optional mit **K** für Partition Key oder **C** für Clustering Column gekennzeichnet. Das Zugriffsmuster auf diese Datenmodelle wird durch Verbindungen zwischen den einzelnen Tabellen und der Zuordnung einzelner Abfragen gekennzeichnet.

[17] Ausführliche Dokumentation bezüglich der Tabellenerstellung. (DataStax, 2015)
[18] Onlineprofile von Artem Chebotko (Chebotko, 2014)

Aufgrund der Datenanalyse und der sich dadurch ergeben Abfragen wurde das Abbildung 4 - Chebotko Diagramm erstellt. Dieses zeigt die jeweiligen Abfragen und deutet eine Reihenfolge bei den Abfragen an, gekennzeichnet durch die gerichteten Verbindungspfeile. Dabei sei angemerkt, dass die Reihenfolge keinerlei Bezugsmöglichkeit innerhalb von Apache Cassandra hat und immer einzelne Abfragen darstellt. Um eine Verbindung zwischen den jeweiligen Anfragen herzustellen muss dies innerhalb der Anwendungslogik geschehen. In diesem Fall müsste das Ergebnis aus **Q1** für die weiteren Anfragen gespeichert werden.

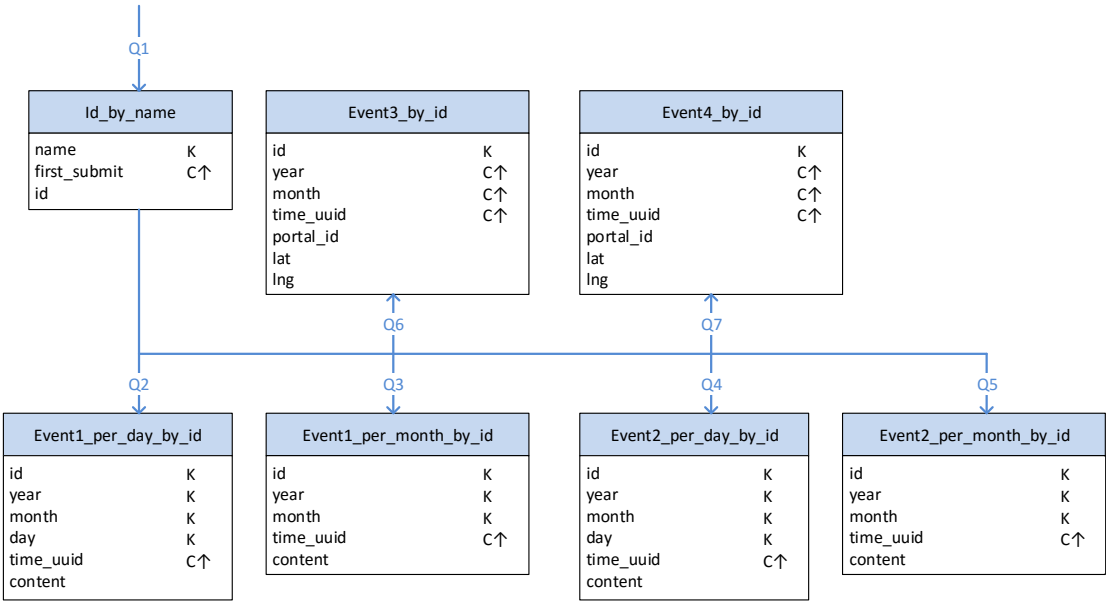

Abbildung 4 - Chebotko Diagramm

6.2.1 Q1: Abfrage der ID unter Angabe des Namens

Für eine vereinfachte Zuordnung der Datensätze können die notwendigen IDs für die Abfrage mit einem Namen abgefragt werden. Dies kann man im späteren Verlauf durch ein beliebiges Authentifikationsverfahren ersetzen um die Daten sichern.

6.2.2 Q2/Q4: Abfrage von Ereignis 1/2 unter Angabe von ID auf Tagesbasis

Die Abfrage auf Tagesbasis ermöglicht die einfache Erstellung von Tages- und Wochenstatistiken für die Ereignisse.

6.2.3 Q3/Q5: Abfrage von Ereignis 1/2 unter Angabe der ID auf Monatsbasis

Die Abfrage auf Monatsbasis ermöglicht die einfache Erstellung von Monats- oder Jahresstatistiken für die Ereignisse.

6.2.4 Q6/Q7: Abfrage von Ereignis 3/4 unter Angabe der ID

Die Abfrage liefert alle für die jeweilige ID hinterlegten Datensätze für das entsprechende Ereignis zurück.

6.3 Erstellung der Tabellen

Das Chebotko Diagramm dient als Grundlage für die Erstellung der Apache Cassandra Datenbanktabellen. Hierzu wird anhand der für die Anfrage Q2 definierten Tabelle dargestellt welche Besonderheiten im Vergleich zur Erstellung von Tabellen für relationale Datenbanken beachtet werden müssen.

```
CREATE TABLE IF NOT EXISTS q1 (
  id         INT
  year       INT
  month      INT
  day        INT
  time_uuid TIME_UUID
  content    VARCHAR
  PRIMARY KEY ((id, year, month, day), time_uuid)
);
```

Es ist zu erkennen, dass die Definition des Primärschlüssels etwas von der für relationale Datenbank üblichen Schema abweicht. Dies ist damit zu erklären, dass bei Apache Cassandra bei der Definition des Primärschlüssels der erste Wert immer der Partitionsschlüssel ist. Um nun einen aus mehreren Spalten zusammengesetzten Partitionsschlüssel nutzen zu können müssen diese durch eine Klammer zusammengefasst als erstes Element der Primärschlüsseldefinition gesetzt werden. Die weiteren Felder innerhalb der Primärschlüsseldefinition ermöglichen das Suchen und Sortieren innerhalb der Tabelle.

Da der Partitionsschlüssel für die Verteilung innerhalb des Apache Cassandra Clusters notwendig ist, kann man bereits anhand der Tabellendefinition erkennen welche Datenmengen wahrscheinlich innerhalb einer Partition gespeichert werden.

Würde man den oben angegebenen Primärschlüssel anders definieren, dann müsste man schon von der Definition davon ausgehen, dass wesentlich mehr Datensätze für eine einzige Partition verwendet werden. In dem Beispiel würden statt den Daten für einen Tag auf einmal die Daten für einen gesamten Monat in einer Partition gespeichert.

```
PRIMARY KEY ((id, year, month), day, time_uuid)
```

7 Server Architektur

Durch die Rahmenbedingungen, siehe Abschnitt 4.1, definierten Vorgaben ergibt sich schon eine relativ feste Strukturierung der zu verwendenden Server. Neben dem Apache Cassandra Cluster wird ein Webserver für die Bereitstellung der Daten verwendet.

Dieser greift über einen für Apache Cassandra bereitgestellten Treiber[19] auf den Cluster zu. Aufgrund der „Masterless“ Struktur von Apache Cassandra kann dem Webserver keine feste Verbindung zu einem bestimmten Server des Apache Cassandra Clusters nachgewiesen werden und wird in Abbildung 5 - Abstraktion der Server Architektur nur abstrahiert zum gesamten Cluster dargestellt.

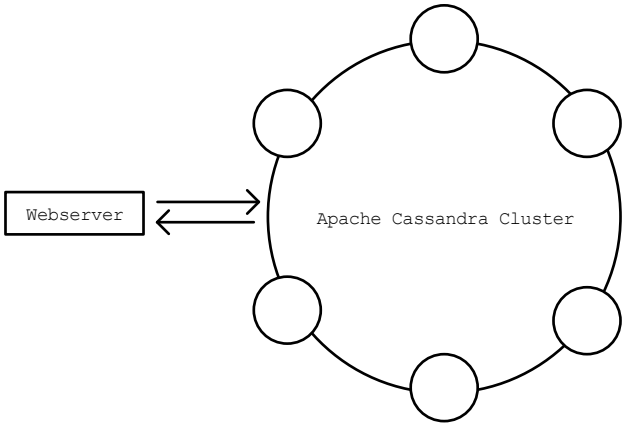

Abbildung 5 - Abstraktion der Server Architektur

7.1 Apache Cassandra Cluster

Für die Verwendung von Apache Cassandra ist im Prinzip keine Verteilung auf mehrere Server notwendig. Somit ist für den Test oder Entwicklungsbetrieb auch das Starten eines Apache Cassandra Nodes auf dem eigenen Rechner möglich. Um die von Apache Cassandra beschriebenen Vorteile der Zuverlässigkeit, Ausfallsicherheit und Geschwindigkeit zu profitieren ist eine Verteilung auf mehrere Rechner unausweichlich.

Dabei ist ein anfänglicher Aufbau mit mindestens 3 Apache Cassandra Nodes üblich um zumindest im Ansatz die bereits erwähnten Vorzüge nutzen zu können. Hierbei ist die Einstellung der zu verwendenden Replikation ebenso wichtig wie die Entscheidung ob bei Abfragen eine höhere Gewichtung auf die Geschwindigkeit oder auf die zu erreichende Datenkonsistenz gelegt werden soll.

[19] Liste aller Verfügbaren Treiber. (Planet Cassandra, 2015)

Eine Erweiterung des Apache Cassandra Clusters ist zu jeder Zeit möglich, aber sollte aufgrund der ansteigenden Datenübertragungsraten gut geplant werden um den normalen Betrieb nicht negativ zu beeinflussen.

7.1.1 Replikation

Bei der erstellen der Datenbanken wird bereits festgelegt welche Strategie von Replikation und welcher Faktor genutzt werden soll. Der Faktor sollte entsprechend der im Cluster vorhandenen Nodes definiert werden und entspricht der Anzahl an Nodes die den gleichen Datensatz vorhalten werden. Bei den Replikationsstrategien[20] wird grundsätzlich zwischen zwei allgemeinen Strategien unterschieden, der „SimpleStrategy“ und der „NetworkTopologyStrategy“.

Die „SimpleStrategy“ platziert entsprechend der Partitionierung den ersten Datensatz und entsprechend des Replikationsfaktors weitere Kopien auf die weiteren Nodes, die Replikationsnodes, im Uhrzeigersinn. Diese Strategie wird meist für die Verwendung von Apache Cassandra innerhalb eines Datenzentrums empfohlen.

Die „NetworkTopologyStrategy“ kann bei der Verteilung der Replikationen die Aufteilung der Nodes auf unterschiedliche Serverregale und Datenzentren berücksichtigen. Grundsätzlich werden Nodes auf unterschiedlichen Serverregalen bevorzugt für die Replikation verwendet, da der Ausfall einer Node unter Umständen auf Problemen des kompletten Serverregals zurückzuführen ist.

Während des Projektes wird die „SimpleStrategy“ verwendet, da nicht auf weitere Datenzentren ausgelagert wird. Außerdem ist bei Bedarf eine Änderung auf eine andere Strategie im Nachhinein immer noch möglich.

7.1.2 Abfragen

Eine Abfrage von Datensätzen bei Apache Cassandra wird von außen immer an eine „Coordinator“ Node gesendet. Diese Node wird meist zugunsten der Verbindung des anfragenden Servers durch den Cluster bestimmt und dient in erster Linie zunächst zur Weiterleitung der Anfrage. Die „Coordinator“ Node muss die angefragten Daten dazu nicht besitzen und kann bei einer erneuten Anfrage eine gänzlich andere Node sein.

[20] Ausführliche Erklärung der Strategien. (DataStax, 2015)

Die „Coordinator“ Node delegiert die Anfrage an die entsprechende Node innerhalb des Clusters weiter und handhabt auch im Falle eines Fehlers mögliche erneute Anfragen. Bei der Anfrage können entsprechen der gewünschten Datenkonsistenz unterschiedliche Konsistenzlevel[21] („Consistency Levels“) mit der Anfrage definiert werden. Dabei werden entsprechend des jeweiligen Konsistenzlevel und des Replikationsfaktors der Datenbank unterschiedlich viele Nodes für die Abfrage der Daten kontaktiert. Dies wird beispielhaft an den drei am Häufigsten verwendeten Konsistenzlevel demonstriert.

7.1.2.1 Konsistenzlevel „All“

Bietet das höchste Maß an Datenkonsistenz aber auch das niedrigste Maß an Verfügbarkeit, da für eine erfolgreiche Abfrage alle Replikationsnodes mit genau dem gleichen Datensatz antworten müssen. Im Falle einer einzigen nicht antwortenden Node ist die Abfrage bereits fehlerhaft.

7.1.2.2 Konsistenzlevel „One“

Es wird nur die Antwort von einer Replikationsnode benötigt für eine erfolgreiche Anfrage. Damit hat das Konsistenzlevel „One“ das höchste Maß an Verfügbarkeit, aber auch eine hohe Wahrscheinlichkeit nicht immer auf dem aktuellen Stand zu sein.

7.1.2.3 Konsistenzlevel „Quorum”

Verspricht eine hohe Datenkonsistenz und Datenaktualität. Dies geschieht unter Abfrage der absoluten Mehrheit der Nodes entsprechend des Replikationsfaktors[22]. Das „Quorum“ Level bietet unter normalen Umständen das ideale Mittelmaß aus Konsistenz und Aktualität und ist wenn nicht anders definiert der Standard für alle Abfragen.

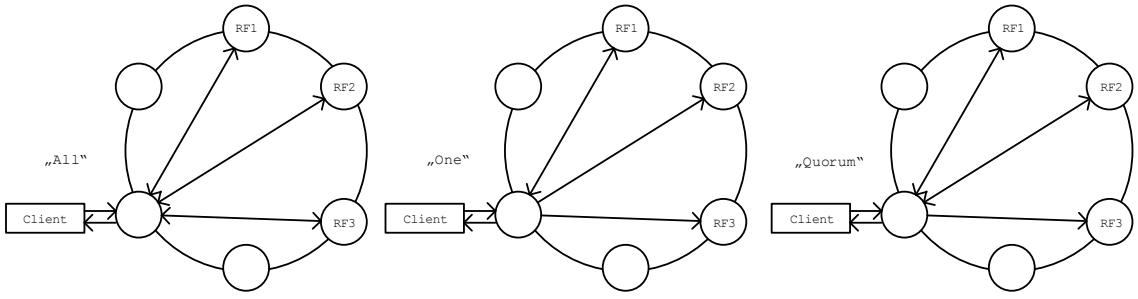

Abbildung 6 - Kommunikationsbedarf der Konsistenzlevel

[21] Auflistung der Konsistenzlevel. (DataStax, 2015)
[22] In ganzen Zahlen berechnet sich die Mehrheit wie folgt: (Faktor/2)+1

7.1.3 Verteilung von Daten

Die Verteilung der Daten und somit die Auslastung des Clusters wird durch die Partitionierung der Datensätze bestimmt. Diese Partitionierung erfolgt unter Berücksichtigung der durch die jeweiligen Tabellen definierten Partitionsschlüssel. Bei einer überdurchschnittlichen Auslastung einzelner Nodes ist meist ein unzureichende Partitionierung und somit ein schlecht definierter Partitionsschlüssel die Ursache.

7.2 Webserver

Der Webserver besteht aus dem Modul zur Abfrage des Apache Cassandra Clusters und dem Webinterface Module für REST Funktionalität. Die lose Verbindung der einzelnen Module ermöglicht ein einfaches Austauschen der Komponenten zu einem späteren Zeitpunkt. Entsprechend der nicht funktionalen Anforderungen, siehe Abschnitt 1.3, kann der Webserver parallel genutzt werden und ist im Falle einer Erweiterung der Auslastbarkeit durch das starten weitere Instanzen zu skalieren.

7.2.1 Modulkopplung

Die einzelnen Module sind in sich abgeschlossene Softwareelemente, welche durch entsprechend definierten Schnittstellen in das Hauptprojekt eingebunden werden. Dabei wird für die Konfiguration auf die durch Spring[23] bereitgestellten Frameworks gesetzt. Dies ermöglicht eine leichtgewichtige Einbindung von verschiedenen Modulen und Funktionalitäten.

7.2.2 REST Modul

Das REST Modul setzt auf die von Spring bereitgestellten Webserver Komponenten und liefert entsprechend der vorgenommenen Konfiguration eine JSON Antwort auf Anfragen gegen die definierten Schnittstellen.

7.2.2.1 Authentifikation

Damit eine erfolgreiche Abfrage gegen den Webserver vollführt werden kann, muss bei jeder Anfrage ein Authentifikationsmerkmal mitgesendet werden. Dies ist ein vom Server generierter und verschlüsselter Token[24]. Der Token enthält die für die Abfragen der Daten bei Apache Cassandra notwendigen Parameter wie zum Beispiel die ID eines Benutzers und ein Ablaufdatum.

[23] Spring. (Spring, 2015)
[24] Das Verfahren ist angelehnt an die für „JavaScript Object Signing and Encryption“ (JOSE) definierten Vorgehensweisen. (JOSE Working Group, 2015)

```
/* unverschlüsselt */
{
  "id":"123456asd324",
  "valid": 1234245
}

/* verschlüsselt */
lDLCvBWUigzeoAdoOlIfFzpFySSaZTFZLGDapgFehQJoWuH1vf3rPvH4mGEavQ
```

Damit vom Webserver ein solcher Token ausgegeben wird, muss zunächst eine Authentifizierung stattfinden. Diese wird im Verlauf des Projektes durch die vereinfachte Angabe eines Namens ermöglicht und kann im Anschluss an das Projekt durch beliebige andere Authentifikationsverfahren ausgetauscht werden.

```
SERVER_ADRESSE/auth/name/BENUTZER_NAME
```

7.2.2.2 Datenabfrage

Die Abfrage der Daten wird unter Verwendung des durch die Authentifikation bereitgestellten Token und eventuell relevanter Parameter durchgeführt. Dabei können je nach Schnittstelle andere Parameter übergeben werden. Bei der Abfrage von Tagesstatistiken kann somit eine Startdatum und eine Anzahl an Tagen übergeben werden.

```
SERVER_ADRESSE/stat/token/TOKEN/day/from/FROM_DATE/days/AMOUNT_OF_DAYS
```

7.2.3 Apache Cassandra Modul

Die Abfragen gegen den Apache Cassandra Cluster werden unter Verwendung des Persistenz-Manager „Achilles“[25] getätigt. Dies ermöglicht unter anderen eine einfache Diagnostik der Abfragen durch transparente Darstellung aller an den Cluster gesendeten Statements, automatisches zuordnen zu Java Objekten entsprechend der Annotationen und eine Unterstützung des JUnit Testframeworks mit einem eingebettet Cassandra Server für Testumgebungen.

[25] Dokumentation des Achilles Projektes (Doan, 2015)

8 Android App

Bei der Programmierung der Android App werden ähnlich der Modularisierung des Webservers auch die einzelnen Funktionen der App in separaten Modulen entwickelt. Somit werden die komplette Serverkommunikation und die Darstellung von Statistiken unabhängig voneinander programmiert. Dies ermöglicht später einen einfachen Austausch der Komponenten und verringert den Aufwand bei Erweiterungen.

8.1 Serverkommunikation

Um eine einheitliche Serverkommunikation bei unterschiedlichen Projekten zu gewährleisten wird ein komplett unabhängiger Serverclient entwickelt. Dieser kann mit allen vom Server gelieferten Datentypen und Rückgabewerten umgehen. Das Modul wird unabhängig von Android entwickelt und bildet eine reine Java Bibliothek, somit ist eine Verwendung unabhängig von Android möglich, dazu wird die OkHttp[26] Java Bibliothek als Netzwerkbibliothek verwendet.

8.2 Statistiken

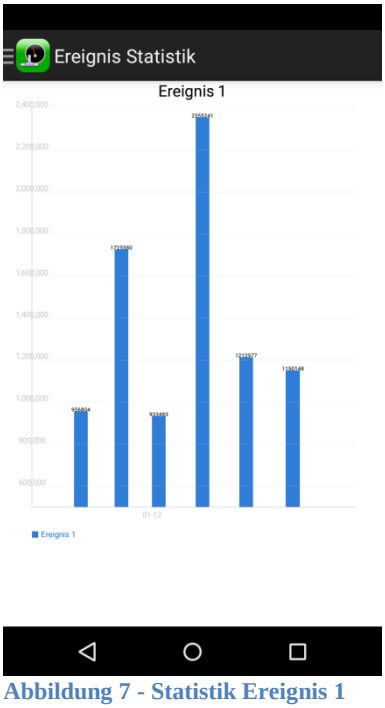

Abbildung 7 - Statistik Ereignis 1

Für die Darstellung wird eine Android Archive Bibliothek[27] erstellt, da dieses auf Programmfunktionen von Android zurückgreift welche sich nicht in einer Java Bibliothek abbilden lassen. Bei den Statistiken bietet sich aufgrund der zeitlichen Gruppierung ein Balken- oder Liniendiagramm an. Diese Diagrammarten werden als vorkonfigurierte Programmteile in der Bibliothek für die weitere Verwendung zu finden sein. Eine Einbindung in die Android App wird wie für Userinterface Elemente üblich über eine Deklaration innerhalb einer Layout Datei möglich sein. Dabei sind auch die grundlegenden Eigenschaften zur Darstellung mit konfigurierbar.

[26] Dokumentation zu Okhttp. (Square, 2015)
[27] Dokumentation zum Android Archive Library (AAR) Format. (Google, 2015)

8.3 Navigation

Der Hauptbestandteil der Android App wird die Darstellung der einzelnen Statistiken sein, aber um diese komfortabel anzeigen zu können muss eine einfache Navigation ermöglicht werden. Im Hinblick auf die Ausbaufähigkeit der App im Anschluss des Projektes um weitere Komponenten sollte die Navigation einfach erweiterbar sein.

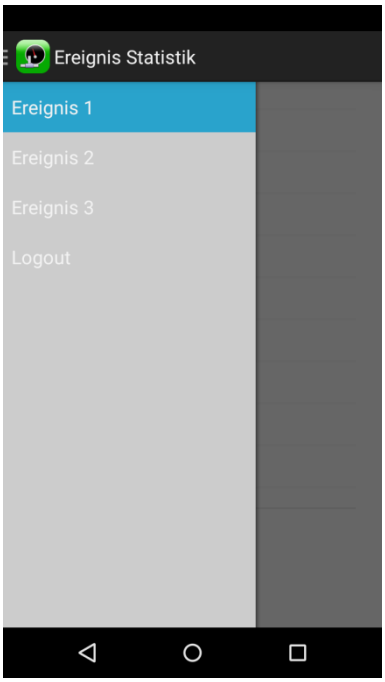

Abbildung 8 - Navigation Drawer

Hier wurde das „Navigation Drawer“ Konzept verwendet, dieses bei Android übliche Navigationsverhalten zeigt Navigationselemente in einem Seitenmenü an. Somit lassen sich nach der Anmeldung mit einem Benutzernamen alle Statistiken über das Seitenmenü direkt aufrufen.

9 Verwendbarkeit von Apache Cassandra

Neben der Umsetzung des Projektes waren neben den von Apache Cassandra beworbenen Aspekte Geschwindigkeit und Ausfallsicherheit auch die Wiederverwendbarkeit für weitere Projekte zu analysieren. Dazu werden für jeden Aspekt unterschiedliche Tests[28] durchgeführt um eine aussagekräftige Bewertung abgeben zu können. Dabei werden die von Datastax geschrieben Hinweise[29] für die Umsetzung der Analyse beachtet.

9.1 Testumgebung

Um für die Auswertung relevante Ergebnisse zu erzielen wird eine der Produktivumgebung ähnlich konfigurierte Architektur verwendet. Dazu wird ausschließlich für die Dauer der Analyse ein weiterer Apache Cassandra Cluster gestartet der unabhängig vom Produktivsystem ist. Dies garantiert auf der einen Seite, dass die Testergebnisse nicht durch etwaige andere Abfragen belastet werden und dass Nutzer der Produktivumgebung auf der anderen Seite nicht durch eventuelle negative Auswirkungen der Tests betroffen werden.

Die für das Projekt definierte Server Architektur sieht für das Lesen und Schreiben von Daten nur jeweils einen Webserver und einen Datenverarbeitungsserver vor. Um eine Auslastung von Apache Cassandra im Bezug auf die zu untersuchenden Aspekte zu erreichen wird für den jeweiligen Test das Lesen und Schreiben von mehreren Servern simuliert.

9.2 Geschwindigkeit

Durch das systematische Ausführen verschiedener Anfragen gegen den Apache Cassandra Cluster wird durch die Anzahl der durchgeführten Operationen pro Sekunde und den Mittelwert der Verzögerung von Antworten eine ungefähre Geschwindigkeit beim Lesen und Schreiben festgestellt. Der Ablauf des Geschwindigkeitstest sieht dabei wie folgt aus:

- 50.000 Schreibzugriffe
- 50.000 Lesezugriffe
- 50.000 Schreib/Lesezugriffe

[28] Angelehnt an das Agile Testverfahren. (Crispin & Gregory, 2008)
[29] Hinweise für Benchmarks. (DataStax, 2014)

Dies wird zunächst von einem Server ausgeführt um einen für das Projekt relevanten Wert zu erreichen und im Anschluss daran auf fünf Server gleichzeitig um eine Auslastung des Clusters zu erreichen.

Die Auswertung, eine genau Auflistung der Ergebnisse befindet sich im Anhang, zeigt, dass die maximale Anzahl an Operationen pro Sekunde bei Verwendung eines Servers weit unter den Werten, welche bei der Verwendung von mehreren abfragenden Servern erreicht werden, liegt. Dies deutet somit auf eine Einschränkung der Geschwindigkeit durch den abfragenden Server hin. Beim Vergleich der jeweiligen Maximalwerte, siehe Abbildung 9 - Operationen pro Sekunde, erkennt man das eine Gesamtsteigerung um fast das Vierfache bei der Verwendung von mehreren abfragenden Servern ermöglicht wurde. Auf Anfragen pro Server runtergerechnet lässt sich eine Verschlechterung der jeweiligen Werte erkennen, was wiederum auf eine Auslastung des Apache Cassandra Clusters zurück zu führen wäre.

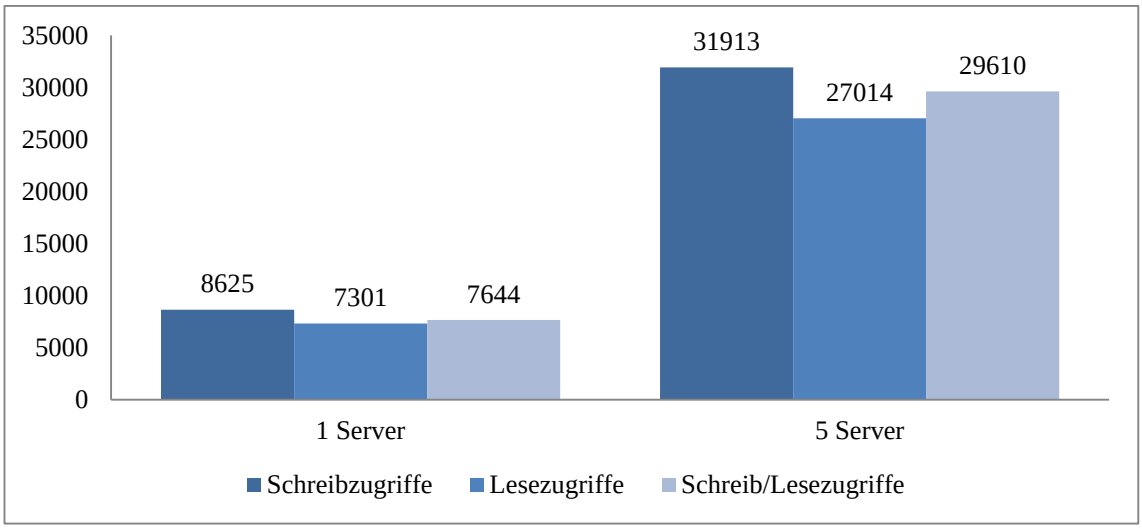

Abbildung 9 - Operationen pro Sekunde

Bei der Auswertung der Geschwindigkeit, siehe Abbildung 10 - Mittelwert der Latenzen in Millisekunden, ist neben der Anzahl von ausgeführten Operationen pro Sekunde auch die Antwortlatenz ein wichtiger Indikator. Hierbei ist zu erkennen, dass Schreibzugriffe eine geringere Verzögerung aufweisen als Lesezugriffe. Außerdem lässt sich bei der Verwendung von mehreren Servern einen Anstieg der Antwortzeiten erkenne, dies würde erneut auf eine Auslastung des Apache Cassandra Clusters hindeuten.

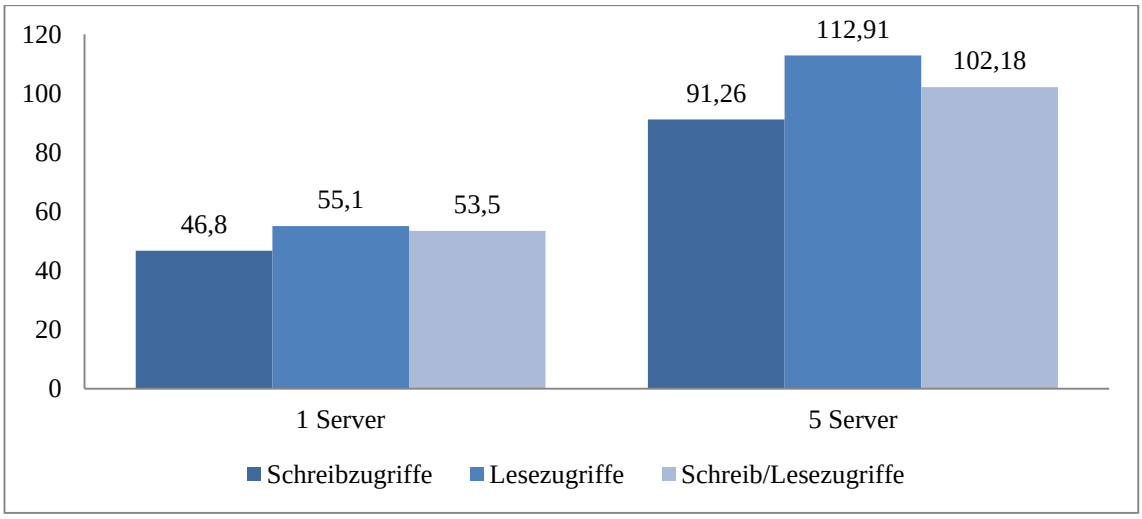

Abbildung 10 - Mittelwert der Latenzen in Millisekunden

Zusammenfassen kann aus den erreichten Werten dabei geschlossen werden, dass bei ausreichend Kapazitäten des Apache Cassandra Clusters eine gute Geschwindigkeit erreicht werden kann. Zu beachten ist dabei immer, dass Schreibzugriffe schneller durchgeführt werden als Lesezugriffe und die jeweiligen Prozesse dahingehend optimieren werden sollten.

Wenn es bei der Verwendung von Apache Cassandra zu Problemen mit der Geschwindigkeit kommt sind neben der Konfiguration des Clusters auch die Clientseite, zum Beispiel die Leistungsfähigkeit des Webservers und der Netzwerkverbindung zum Apache Cassandra Cluster, zu betrachten.

9.3 Ausfallsicherheit

Zum Feststellen wie gut Apache Cassandra bei einem Ausfall von einzelnen Server reagiert wird der im Geschwindigkeitstest bereits verwendete Testablauf von einem abfragenden Server wiederholt und zusätzlich einzelne Apache Cassandra Nodes vom Cluster isoliert. Um eine Abhängigkeit der Ausfallsicherheit zu dem verwendeten Konsistenzlevel aufzuzeigen wird der Test an dieser Stelle die Abfrage mit den Konsistenzlevel „One“ und „Quorum“ durchführen. Die Anzahl der fehlgeschlagenen Abfragen im Verhältnis zu den erfolgreich durchgeführten Abfragen wird dabei als Faktor für die Ausfallsicherheit verwendet.

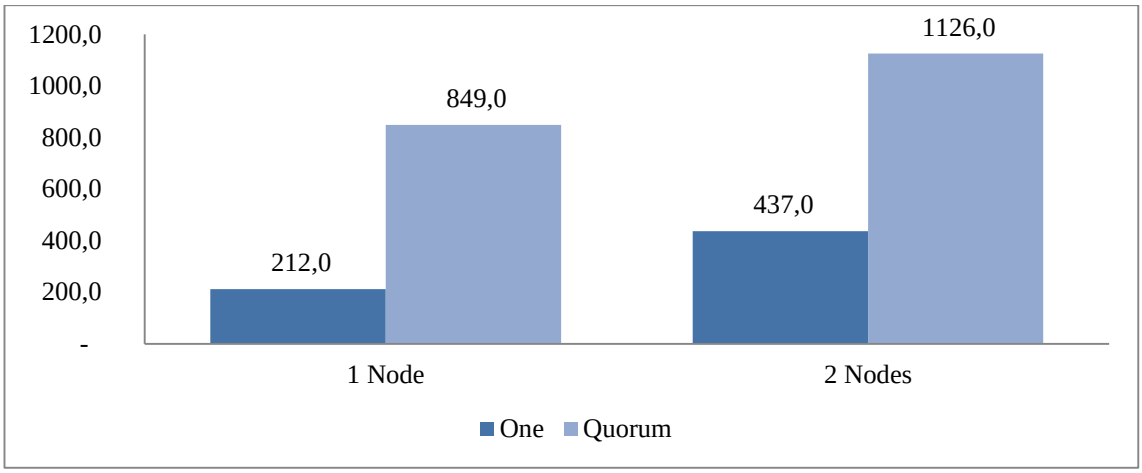

Abbildung 11 - Fehlgeschlagene Anfragen

Den Testergebnissen kann man generell entnehmen, dass zu keinem Moment mehr als 0.023% Abfragen fehlgeschlagen sind. Diese Quote ist eigentlich so gering, dass man dies auf Anwendungsebene immer noch durch eine erneute Abfrage der Daten und eine Anpassung des Konsistenzlevels kompensieren könnte.

Der Vergleich einzelnen Konsistenzlevel zeigt dabei deutlich, dass es zu nicht unerheblich weniger Abfragefehler beim Level „One“ im Vergleich zum Level „Quorum“ gekommen ist. Dies könnte man darauf zurückführen, dass für eine erfolgreiche „Quorum“-Abfrage mehr Server des Apache Cassandra Clusters abgefragt werden müssen.

Für die Verwendung sollte dementsprechend darauf geachtet werden, welche Vor- und Nachteile der jeweilige Konsistenzlevel für die Ausfallsicherheit mit sich bringt, siehe Abschnitt 7.1.2 Abfragen.

9.4 Wiederverwendbarkeit

Aufgrund der Apache Cassandra typischen Verwendung der einzelnen Datenmodelle und des kompletten relationsfreien Aufbau ist zumindest aus der Perspektive der Datenmodelle keine direkte Wiederverwendbarkeit zu erkennen. Der Apache Cassandra Cluster wiederum kann für die Verwendung mit mehreren Projekten verwendet werden, jedoch sollten dafür aber die einzelnen Projekte jeweils darauf untersucht werden, wie viel Last diese maximal auf den Cluster ausüben würden. Dies dient dazu, dass die Summe aller maximalen Belastungen nicht zu einer negativen Entwicklung von Geschwindigkeit oder Verfügbarkeit des Apache Cassandra Clusters führen. Im Falle einer solchen wahrscheinlichen Überbelastung kann natürlich im Vorfeld bereits an der

Leistung des Apache Cassandra Clusters gearbeitet werden, indem zum Beispiel weitere Server eingebunden werden um die Auslastung auf die einzelnen Server zu verringern.

10 Qualitätssicherung

Die Qualitätssicherung teilt sich im wesentlich auf die zwei Aspekte Datenmodellierung, welche bereits bei der Erstellung der Datenmodelle durchgeführt wird, und Programmmodule, welche die Qualität der einzelnen Module gewährleistet, auf. Bei den Programmmodulen wird keine Unterscheidung zwischen den Server und den Android App Modulen gemacht, da für den überwiegenden Teil der Module auf die gleichen Teststrategien gesetzt wird.

10.1 Planung

Für eine durchgehende Absicherung der Qualität wird mit jedem im Projektplan abgeschlossenen Meilenstein die jeweiligen Qualitätssicherungen für alle vorherigen Meilensteine erneut durchgeführt. Dies garantiert das eventuelle Änderungen an zuvor abgeschlossenen Projektphasen auch weiterhin geprüft werden.

10.2 Datenmodellierung

Um zu gewährleisten, dass die Datenmodelle sich im späteren produktiven Einsatz negativ entwickeln wird neben der eigentlichen Modellierung zusätzlich auf Stresstests gesetzt um die Entwicklung der einzelnen Datenmodelle nach mehreren Tausend Schreib- und Lesezugriffe beurteilen zu können. Hierzu wird das von Apache Cassandra mitgelieferte Stress Tool[30] eingesetzt.

10.2.1 Stress Test Definition

Für die einzelnen Stresstests wird jeweils eine YAML[31] Datei angelegt. Die enthält Angaben zum verwendeten Keyspace und einer Definition falls der Keyspace noch nicht vorhanden ist.

[30] Stress Tool Dokumentation. (DataStax, 2014)
[31] YAML Dokumentation. (Ben-Kiki, Evans, & döt Net, 2009)

```
# --- DML --- #
# KEYSPACE NAME
keyspace: stresscql

# (OPTIONAL)
keyspace_definition: |
CREATE KEYSPACE stresscql WITH replication = {'class': 'SimpleStrategy',
'replication_factor': 1};
```

Ähnlich sieht es bei der Angabe der Tabelle aus, auch hier ist optional eine Definition anzugeben.

```
# TABLE NAME
table: blogposts

# (OPTIONAL)
table_definition: |
  CREATE TABLE table1 (
        id INT,
        date TIMEUUID,
        content text,
        PRIMARY KEY(id, date)
  ) WITH CLUSTERING ORDER BY (date DESC)
    AND compaction = { 'class':'LeveledCompactionStrategy' }
    AND comment='a test table'
```

Zur Durchführung können dann für die Spalten unterschiedliche Parameter angegeben werden, somit können zum Beispiel für Schlüsselspalten vorher definierte Längen eingehalten werden oder auch die Anzahl der einzufügenden Datensätze kann so bestimmt werden.

```
# --- COLUMN DISTRIBUTION SPECIFICATIONS --- #
columnspec:
  - name: id
    size: gaussian(50..100)      #id spread is minimal
    population: uniform(1..1M)  #1M possible ids

  - name: date
    cluster: fixed(720)          #2 years of data per id

  - name: content
    size: uniform(30..500)
```

Außerdem kann festgelegt werden wie sich INSERT Statements verhalten sollen, so kann zum Beispiel die Menge der Datensätze pro INSERT Statement beschränkt werden.

```
# --- BATCH RATIO DISTRIBUTION SPECIFICATIONS --- #
insert:
  partitions: fixed(1)
  select:    fixed(1)/90        # last 3 month per batch possible
  batchtype: UNLOGGED
```

Neben der Konfiguration der Schreibvorgänge können selbstverständlich auch die Lesevorgänge konfiguriert werden um somit die später wahrscheinlichen Anfragen direkt mit zu testen.

```
# EINE LISTE VON ZU TESTENDEN ANFRAGEN
queries:
   singledate:
      cql: SELECT * FROM table1 WHERE id = ? LIMIT 1
      fields: samerow

   week:
      cql: SELECT * FROM table1 WHERE id = ? LIMIT 10
      fields: samerow
```

10.2.2 Stresstest Durchführung

Es können für einen Stresstest unterschiedliche Kombinationen der in der Definition hinterlegten Operationen verwendet werden. Nach jedem durchlaufen eines Stresstest wird zu dem durchgeführten Tests eine kompakte Übersicht der Ergebnisse geliefert und eine detaillierte Auflistung der einzelnen Durchläufe. Für eine erste Analyse der Datenmodelle ist die kompakte Übersicht mehr als ausreichend, da sich hieraus schon ablesen würde wenn etwas falsch gemacht wurde. So zeigt diese an wie viele Partitionen und Datensätze erstellt worden sind und lässt damit einen direkten Rückschluss auf die Verteilung innerhalb des Clusters schließen.

```
./cassandra-stress user profile=./test1.yaml ops\(insert=1\)

Results:
op rate                   : 8625
partition rate            : 8625
row rate                  : 8612
latency mean              : 46.8
latency median            : 34.5
latency 95th percentile   : 121.9
latency 99th percentile   : 203.4
latency 99.9th percentile : 600.4
latency max               : 877.0
Total operation time      : 00:00:42
Improvement over 271 threadCount: 1%
```

10.3 Programmmodule

Bei der Qualitätssicherung der einzelnen Programmmodule wird auf die Testwerkzeuge von JUnit[32] zurückgegriffen. Die Konfiguration von JUnit wird pro Module etwas unterschiedlich durchgeführt, da sich die jeweiligen zu testenden Funktionen stark unterscheiden können. Anhand des Apache Cassandra Modules kann eine Beispielkonfiguration dargestellt werden.

Um das Apache Cassandra Modul erfolgreich testen zu können ist immer eine Verbindung zu einem Apache Cassandra Server notwendig, dies wird unter Verwendung der durch Achilles bereitgestellten Testkonfiguration gewährleistet. Somit lässt sich mit wenig Aufwand bereits die komplette Konfiguration eines Apache Servers definieren.

```
@Rule
public AchillesResource resource = AchillesResourceBuilder
      .withEntityPackages("de.test.entities")
      .withKeyspaceName("testKeyspace")
      .truncateBeforeAndAfterTest()
      .build();
```

Die jeweiligen Tests können nun auf den konfigurierten Apache Cassandra Server zugreifen und somit unabhängig von der Ausführungsumgebung erfolgen. Die Tests werden dem JUnit Framework üblich definiert und sollten idealerweise 100% aller Funktionen des Moduls abdecken.

[32] Dokumentation zu JUnit. (JUnit, 2015)

11 Perspektive

Das Projekt wurde im Hinblick auf die mögliche Erweiterbarkeit, Anpassungen und Wiederverwendbarkeit entwickelt und hat bereits während des Projektes klare Tendenzen in die jeweiligen Richtungen gezeigt.

11.1 Erweiterbarkeit

Zu einem späteren Zeitpunkt sollen neben den bisherigen Statistiken auch allgemeine Statistiken für den Vergleich zwischen den jeweiligen Benutzern möglich sein. Somit könnten sich die Benutzer gegenseitig anspornen eine jeweils bessere Statistik zu erreichen. Außerdem werden wohl in Anschluss an das Projekt verschiedene zusätzliche Statistiken modelliert werden um weitere Elemente aus den ursprünglichen Datensätzen verarbeiten zu können.

11.2 Anpassung

Bei der im Rahmen des Projektes entwickelten Android App sind im Verlauf der Abnahme durch die Benutzer noch Anmerkungen bezüglich der Navigation und Sicherheit gemacht worden.

Die Navigation ist unter anderem beim Umschalten der für die Benutzer relevanten Statistiken eine zu lange Interaktionskette bemängelt worden. Dies würde dafür sprechen, dass die erste Anpassung eine Gewichtung der einzelnen Statistiken vorsehen würde um diese dann durch andere Navigationskomponenten leichter erreichbar zu machen. Die Sicherheit wurde im Vorfeld mit den Benutzern für die erste Umsetzung des Projektes als nicht so relevant genannt. Jedoch sehen einige der Benutzer im Zusammenhang mit den nun relativ einfach einsehbaren Statistiken dies etwas anders und würden ein richtiges Authentifizierungsverfahren bevorzugen.

11.3 Wiederverwendbarkeit

Der stark modularisierten Aufbau der einzelnen Komponenten war trotz des dadurch gestiegenen Aufwands eine der wichtigsten Entscheidungen um die spätere Wiederverwendbarkeit zu garantieren. Dadurch sind die einzelnen Programmmodule, wie zum Beispiel das Apache Cassandra Module des Webservers oder die Android Komponenten für die Darstellung von Statistiken, ohne großen Aufwand in andere Projekte verwendbar.

12 Fazit

Nach Abschluss des Projektes können verschiedene Schlussfolgerungen in Bezug auf die Planung und Durchführung eines solchen Projektes gezogen werden. Allgemein ist eine gute Planung maßgeblich für die Machbarkeit eines Projektes, dies verdeutlichte sich auch bei der Durchführung dieses Projektes. So wurden in der Planungsphase die verschiedenen notwendigen Schritte ausgearbeitet, gewichtet und in Relation zueinander gebracht. Trotz einer anfänglich eher großzügigen Auslegung der einzelnen Projektphasen gab es nicht planbare Verzögerungen, wie zum Beispiel ein Ausfall der Internetverbindung an dem für das Projekt relevanten Server oder die Umplanung von Terminen mit Benutzern zur Absprache der Anforderung, die zu leichten Verschiebungen des Projektplans geführt haben.

Bei der Auswertung der Verwendbarkeit von Apache Cassandra sind die meisten Aussagen bezüglich der Geschwindigkeit, Ausfallsicherheit und Wiederverwendbarkeit in Bezug auf das Projekt als richtig zu betrachten. Dies sei aber unter der Voraussetzung einiger Änderungen in der Vorgehensweise bezüglich der Datenspeicherung angemerkt. Diese Voraussetzungen würde zu aller erst das mehrfache Speichern identischer Datensätze betreffen, da nur auf diese Weise Apache Cassandra zu der versprochenen Geschwindigkeit kommt. Die Ausfallsicherheit ist durch die Konfiguration der notwendigen Replikationsangaben, siehe Abschnitt 7.1.1, und einer Planung der notwendigen Vorbedingungen, wie zum Beispiel die gewünschte Anzahl an Apache Cassandra Nodes oder die Verwendung unterschiedlicher Datenzentren, sicherlich gegeben, aber kann unter falscher Nutzung eine Beeinträchtigung der Geschwindigkeit hervorrufen. Die Wiederverwendbarkeit ist der wohl kritischsten Punkt von Apache Cassandra, da aufgrund der notwendigen Abfrage bezogenen Modellierung der Tabellen können diese bei kleinen Anpassungen auf die zu erwartenden Daten meist nicht sinnvoll weiter verwendet werden. Dies führt in der Regel zur Erstellung einer weiteren Datenbanktabelle mit den Daten aus der ursprünglichen Tabelle und den zusätzlich erwarteten Daten.

Die Darstellung der ausgewerteten Daten ist bei der erfolgreichen Abnahme durch die Benutzer positiv bewertet worden und führte bei einigen zu einem überdenken bei der Verwendung von Apps. So war vielen Anwendern vorher nicht klar in welchem Umfang Apps Information über die eigene Person preisgeben bzw. welche Rückschlüsse aus den gesammelten Daten gezogen werden können. Die Statistiken an

sich zeigen zwar nur im minimal Informationen zur Person, wie zum Beispiel an welchen Wochentagen der Benutzer die App am häufigsten verwendet, aber im Zusammenhang mit der Auswertung der GeoDaten konnte einzelnen Benutzern ganz deutlich gezeigt werden, dass ein fast lückenloser Tagesablauf dokumentiert werden kann solang die App genutzt wird.

Zusammenfassend kann bei Apps in Verbindung mit der Sammlung von Daten und im speziellen GeoDaten nur jedem angeraten werden selbst einzuschätzen, ob die gerade installierte App unbedingt zusätzliche Informationen, wie den aktuellen Standort, benötigt und ob man diese Sorte von Apps, welche ohne jene Informationen nicht funktionieren wollen, unbedingt braucht.

13 Anhang

Tabellen zur Auswertung der Geschwindigkeit

Threads	Operation/Sek	Latenz(ms)
4	4459	0.9
8	5177	1.5
16	6439	2.5
24	6933	3.4
36	7345	4.9
54	7976	6.8
81	8238	9.8
121	8267	14.6
181	8409	21.4
271	8561	31.4
406	8625	46.8

Tabelle 2 - Auswertung Schreibzugriffe von einem Server

Threads	Operation/Sek	Latenz(ms)
4	21180	1.13
8	24591	1.88
16	30585	3.13
24	32932	4.59
36	27544	6.62
54	29910	10.54
81	30481	16.17
121	30588	25.55
181	31113	41.73
271	31676	61.23
406	31913	91.26

Tabelle 3- Auswertung Schreibzugriffe von fünf Server

Threads	Operation/Sek	Latenz(ms)
4	3212	1.2
8	4461	1.8
16	5769	2.8
24	6099	3.9
36	5894	6.1
54	6482	8.3
81	6967	11.6
121	7026	17.1
181	7243	24.8
271	7267	37.7
406	7301	55.1

Tabelle 4- Auswertung Lesezugriffe von einem Server

Threads	Operation/Sek	Latenz(ms)
4	15257	1.50
8	21190	2.25
16	27403	3.63
24	28970	5.27
36	23288	8.22
54	24308	12.71
81	25778	19.14
121	26026	29.93
181	26810	48.95
271	26888	73.52
406	27014	112.91

Tabelle 5- Auswertung Lesezugriffe von fünf Servern

Threads	Operation/Sek	Latenz(ms)
4	3682	1.10
8	4626	1.73
16	4883	2.78
24	5278	3.83
36	6620	5.78
54	6940	7.93
81	7298	11.24
121	7341	16.64
181	7513	24.26
271	7597	36.28
406	7644	53.50

Tabelle 6 - Auswertung Schreib/Lesezugriffe von einem Server

Threads	Operation/Sek	Latenz(ms)
4	16397	1.31
8	20601	2.06
16	24645	3.38
24	25999	4.93
36	26686	7.43
54	27380	11.64
81	28411	17.67
121	28420	27.77
181	29251	45.38
271	29575	67.44
406	29610	102.18

Tabelle 7 - Auswertung Schreib/Lesezugriffe von fünf Servern

14 Literaturverzeichnis

Apache Cassandra. (12. Oktober 2011). *CQL*. Abgerufen am 19. August 2015 von Apache Cassandra: https://cassandra.apache.org/doc/cql/CQL.html

Apache Cassandra. (2015). *Hauptseite*. Abgerufen am 20. August 2015 von Apache Cassandra: http://cassandra.apache.org/

Ben-Kiki, O., Evans, C., & döt Net, I. (1. Oktober 2009). *YAML Ain't Markup Language (YAML™) Version 1.2*. Abgerufen am August 2015 von YAML: http://www.yaml.org/spec/1.2/spec.html

Chebotko, A. (Januar 2014). *Artem Chebotko*. Abgerufen am 20. August 2015 von LinkedIn: https://www.linkedin.com/in/artemchebotko

Crispin, L., & Gregory, J. (2008). In *Agile Testing: A Practical Guide for Testers and Agile Teams* (S. 233-239). Addison-Wesley Professional.

DataStax. (4. Februar 2014). *How not to benchmark Cassandra*. Abgerufen am August 2015 von DataStax Developer Blog: http://www.datastax.com/dev/blog/how-not-to-benchmark-cassandra

DataStax. (31. Juli 2014). *Improved Cassandra 2.1 Stress Tool: Benchmark Any Schema – Part 1*. Abgerufen am August 2015 von DataStax Developer Blog: http://www.datastax.com/dev/blog/improved-cassandra-2-1-stress-tool-benchmark-any-schema

DataStax. (2. Februar 2015). *Basic Rules of Cassandra Data Modeling*. Abgerufen am 20. August 2015 von DataStax Developer Blog: http://www.datastax.com/dev/blog/basic-rules-of-cassandra-data-modeling

DataStax. (13. August 2015). *Configure data consistency*. Abgerufen am August 2015 von DataStax Document: http://docs.datastax.com/en//cassandra/2.0/cassandra/dml/dml_config_consistency_c.html

DataStax. (13. August 2015). *CQL for Cassandra 1.2*. Abgerufen am 20. August 2015 von DataStax Document: http://docs.datastax.com/en/cql/3.0/cql/cql_reference/create_table_r.html

DataStax. (13. August 2015). *Data replication*. Abgerufen am August 2015 von DataStax Document: http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication_c.html

Doan, D. (August 2015). *Achilles*. Abgerufen am 20. August 2015 von Github: http://doanduyhai.github.io/Achilles/

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). *From Data Mining to Knowledge Discovery in Databases*. Abgerufen am August 2015 von KDNugget: http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf

Fielding, R. T. (2000). *Representational State Transfer (REST)*. Abgerufen am August 2015 von https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Google. (2015). *AAR Format*. Abgerufen am August 2015 von Android Tool Project Site: http://tools.android.com/tech-docs/new-build-system/aar-format

Google. (2015). *Alpha-/Betatests und gestaffelte Einführungen verwenden*. Abgerufen am August 2015 von Developer Console Hilfe: https://support.google.com/googleplay/android-developer/answer/3131213

Google. (2015). *Android Developer Training*. Abgerufen am August 2015 von Creating a Navigation Drawer: https://developer.android.com/training/implementing-navigation/nav-drawer.html

Google. (03. August 2015). *Dashboard*. Abgerufen am 20. August 2015 von Android Developer: https://developer.android.com/about/dashboards/index.html

Ingress Enlightened Köln. (August 2015). *Ingress Enlightened Köln - Inceptions*. Abgerufen am 20. August 2015 von Google Plus: http://www.enlightened-koeln.de/

Internet Engineering Task Force (IETF). (August 2012). *The 'application/zlib' and 'application/gzip' Media Types*. Abgerufen am August 2015 von Internet Engineering Task Force: https://tools.ietf.org/html/rfc6713

JOSE Working Group. (13. Januar 2015). *JSON Web Algorithms (JWA)*. Abgerufen am August 2015 von Tools IETF: https://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms-40

JUnit. (16. August 2015). *Junit Cookbook*. Abgerufen am 20. August 2015 von Junit : http://junit.org/cookbook.html

Niantic Labs. (2015). *Ingress*. Abgerufen am 2015 von Ingress: https://www.ingress.com/

Niantic Labs. (2015). *Niantic Labs*. Abgerufen am August 2015 von http://www.nianticlabs.com/

Oracle. (2015). *Java*. Abgerufen am August 2015 von Java: http://www.java.com/

Planet Cassandra. (2015). *Client Drivers*. Abgerufen am August 2015 von Planet Cassandra: http://www.planetcassandra.org/client-drivers-tools/

Planet Cassandra. (2015). *What is Apache Cassandra*. Abgerufen am 17. August 2015 von Planet Cassandra: http://www.planetcassandra.org/what-is-apache-cassandra/

Spring. (2015). *Spring*. Abgerufen am August 2015 von http://spring.io/

Square. (17. März 2015). *OkHttp*. Abgerufen am August 2015 von Github: http://square.github.io/okhttp/

Wake, B. (17. August 2003). *Invest in good stories and smart tasks*. Abgerufen am 19. August 2015 von XP123: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Widgetlabs GmbH. (2015). *Widgetlabs*. Abgerufen am 2015 von Widgetlabs: http://widgetlabs.eu/

15 Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsgbehörde vorgelegen.

Köln, 31. August 2015

Benjamin Werker