Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

Fakultät für Elektrotechnik und Informatik
Studiengang Informatik Master

Projektarbeit

Zählerstände

vorgelegt von:

Roman Schneider

29 Juli 2013

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.
Die Arbeit wurde bisher, in gleicher oder ähnlicher Form, keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Unterschrift	Ort, Datum

Vorwort

Die Preise von Ressourcen und Energieträgern steigen überdurchschnittlich stark an. Deshalb ist es für viele Privathaushalte sinnvoll, den Verbrauch von zum Beispiel Wasser, Strom, Gas oder Heizöl im Blick zu behalten. Durch die kontinuierliche Überwachung können unnötige Verbräuche festgestellt werden. Die Aufnahme der Messwerte kann einerseits automatisch erfolgen. Dafür sind allerdings Eingriffe in das Versorgungsnetz und teilweise beträchtliche Investitionen nötig. Um kurzfristig, ohne großen Aufwand, eine Auswertung zu erhalten benötigt man ein nicht- oder nur minimal invasives Überwachungswerkzeug.

Ziel dieser Arbeit ist es, ein solches minimal invasives Werkzeug zur Erfassung von Zählerständen zu erstellen.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Diese Arbeit mit dem Thema *Zählerstände* ist im Rahmen des Bachelorstudiengangs Wirtschaftsinformatik und eBusiness der Hochschule Ravensburg – Weingarten entstanden.

1.1. Problemstellung

Durch stetig steigende Kosten von Ressourcen und Energie kann und sollte speziell in diesem Lebensbereich sparsam und verantwortungsbewusst konsumiert werden. Im Alltag erkennt man schwer ob man den einen oder anderen Liter Wasser oder Öl mehr verbraucht. Geräte im Standby verbrauchen ebenfalls unnötigerweise viel Energie.

Da heutzutage in den wenigsten Haushalten Zähler mit Verbindung zum Internet verbaut sind, gibt es keinen einfachen Weg um Zählerstände zu erfassen und die erfassten Stände zu speichern und zu verwalten.

Um einen Überblick über die anfallenden Verbräuche zu behalten, müssten diese in regelmäßigen Abständen erfasst und gespeichert werden. Diese erfassten Stände könnten dann zu Auswertungszwecken hergenommen werden.

1.2. Zielsetzung

Das Ziel dieser Arbeit ist die Konzeption und Entwicklung einer Software, mit deren Hilfe Zählerstände einfach und komfortabel erfasst und abgespeichert werden können.

Zusätzlich soll eine automatische Auswertung der bisher abgespeicherten Zählerstände, aufgrund von verschiedenen Aspekten, automatisch erfolgen.

Um eine einfache und benutzerfreundliche, am besten sogar kostenlose Erfassung der Daten zu ermöglichen soll, die Technik so gewählt werden, dass der Benutzer ein Gerät verwenden kann, das tragbar ist und möglichst viele Haushalte bereits für andere Zwecke im Einsatz haben.

1.3. Aufbau und Übersicht der Arbeit

Das zweite Kapitel behandelt die Grundlagen zur Entwicklung der Software. Im Speziellen werden hier die eingesetzten Tools und Techniken, die zur Erreichung der Ziele verwendet wurden, kurz vorgestellt und beschrieben.

Im dritten Kapitel wird der theoretische Teil der Arbeit behandelt. Es wird eine Auswahl der bereits auf dem Markt befindlichen Software näher betrachtet. Daraufhin wird die Architektur der zu entstehenden Software beschrieben und erläutert. Des Weiteren wird hier auf den Auswahlprozess der eingesetzten Tools eingegangen.

Das vierte Kapitel beschreibt die praktische Umsetzung der Arbeit, also den Implementierungsprozess selbst.

Im fünften Kapitel werden die Ergebnisse der Arbeit vorgestellt. Als weiterer Punkt wird bildhaft gezeigt, in wie fern sich die Vision der Software von dem entstandenen Ergebnis abweicht.

Das sechste und letzte Kapitel soll eine Zusammenfassung der Arbeit liefern. Zusätzlich wird ein Ausblick auf mögliche Weiterentwicklungen gewährt.

2. Grundlagen

Dieses Kapitel beschreibt kurz die wichtigsten im Entwicklungsprozess dieser Arbeit eingesetzten Tools und Techniken.

2.1. Tools

2.1.1. Eclipse

Eclipse ist eine sogenannte integrierte Entwicklungsumgebung[1]. Es ist ein Open-Source-Projekt zur Entwicklung von Software.

Aufgrund seiner Erweiterbarkeit ist es eine viel genutzte Entwicklungsumgebung.

2.1.2. ADT Bundle

Das ADT (Android Developer Tools) Bundle[2] liefert alle benötigten Bausteine zur Entwicklung von Android Apps.

Es beinhaltet eine Version von Eclipse inklusive des ADT Plugins. Zusätzlich enthält es die Android SDK[2] Tools. Das ADT Bundle mit seinen Tools unterstützt den Entwickler zum Beispiel beim Entwickeln der Benutzeroberfläche mit dem eingebauten Grafikeditor.

Des Weiteren erhält man mit diesem Bundle die neuesten Android System Images für den Emulator. Mit diesen ist es möglich, auf dem Entwicklungsrechner ein virtuelles Android System zu starten sowie entwickelte Android Apps auszuführen und zu debuggen.

2.1.3. Google Plugin for Eclipse

Das Google Plugin für Eclipse ist eine Sammlung von Entwicklungswerkzeugen zum Designen, Bauen und Deployen von cloudbasierten Anwendungen.

Mit diesem Plugin kann Eclipse[7] so erweitert werden, dass es alle notwendigen Mittel zur Entwicklung von Google App Engine Projekten besitzt.

[1] Integrated design environment, kurz IDE
[2] SDK, kurz für Software Development Kit

In Verbindung mit den Android Development Tools kann ein sogenanntes „App Engine connected Android Project“ erstellt werden.

Bei einem solchen Projekt können direkt zu den erstellten Cloud Endpoints[3] (Abbildung 1) die für den Client dazugehörigen Libraries erstellt werden.

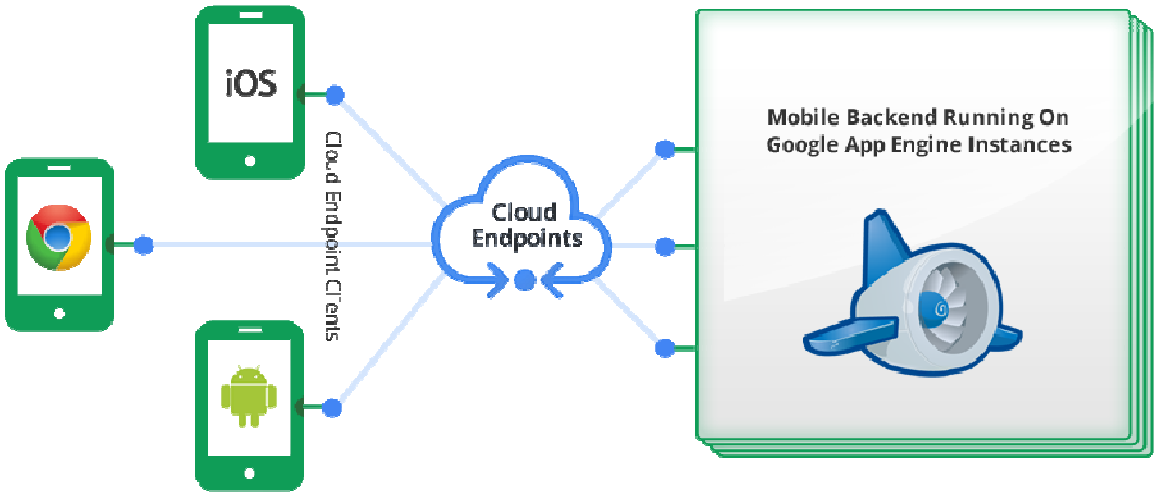

Abbildung 1: Cloud Endpoints:

2.1.4. Git

“Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”[6]

Git ist ein beliebtes Versionskontrollsystem, das auf fast allen modernen Betriebssystemen läuft. Es wird von vielen namhaften Unternehmen[6], wie zum Beispiel Google oder Microsoft zur Entwicklung eingesetzt.

2.2. Techniken

2.2.1. Android

Android ist ein Betriebssystem für mobile Geräte wie Smartphones und Tablet-Computer. Es basiert auf einem Linux-Kernel und wird quelloffen entwickelt.

[3] Schnittstelle zur Kommunikation mit dem Server

2013 erwartet Google 900 Millionen aktivierte Android Geräte[3], was mehr als doppelt so viele wie in 2012 wären.

Auf dem Smartphone Markt liegt der Anteil an Geräten, die mit Android betrieben werden, bei knapp 75%[14].

2.2.2. Google App Engine

Google App Engine ist eine Plattform zum Entwickeln und Hosten von Webanwendungen. „Google App Engine ermöglicht Ihnen die Erstellung von Webanwendungen auf denselben skalierbaren Systemen, auf denen auch die Anwendungen von Google laufen.“[8]

2.2.3. OCR

OCR ist die Abkürzung für *Optical Character Recognition* und bedeutet auf Deutsch optische Zeichenerkennung.
Bei diesem Verfahren werden Zeichen in Bilddateien, die durch Scanner oder Digitalkameras aufgenommen wurden, erkannt.

2.2.4. JDO

JDO steht für *Java Data Objects* und ist eine API[4] zu Persistierung von Java Objekten. Eine Anwendung, die mit JDO arbeitet, funktioniert mit verschiedenen Arten von Datenbanken.

2.2.5. REST

REST ist die Kurzform für *Representational State Tranfer*. Für REST Schnittstellen wird in der Regel HTTP[5] zur Übertragung der Daten verwendet.

2.2.6. JSON

JSON bedeutet ausgeschrieben *JavaScript Object Notation* und ist ein Format zum Austausch von Daten. Es ist komplett unabhängig von der eingesetzten Programmiersprache.

[4] Application programming interface engl. für Programmierschnittstelle
[5] Hypertext-Tranfer Protokoll

3. Konzeption

In diesem Kapitel geht es um die theoretische Umsetzung der Projektarbeit. Zunächst wird das bestehende Angebot anderer ähnlicher Apps auf dem Markt analysiert. Daraufhin folgt eine Beschreibung der Architektur der Anwendung. Des Weiteren werden die Anforderungen beschrieben und kurz auf die Wahl der Tools eingegangen.

3.1. Marktanalyse

Die in *Google play*[10] am besten bewerteten Apps, die eine ähnliche Funktion wie die in dieser Arbeit erarbeitete Anwendung erfüllen, sind:

- Zählerstand[11]
- Energieverbrauchs-Analysator[9]

Die Anwendung *Zählerstände* besitzt ein ansprechendes und modernes Design. Die Benutzeroberfläche ist einfach gehalten und wirkt aufgeräumt. Mit dieser App können Zählerstände für Strom, Wasser und Heizung durch manuelle Eingabe der Werte erfasst werden. Es gibt keine Möglichkeit zusätzliche Zähler anzulegen. Die Daten werden lokal gespeichert, können in der Bezahlversion jedoch exportiert und importiert werden. Auswertungen der Zählerstände werden sowohl in Schriftform als auch grafisch präsentiert.

Die Anwendung *Energieverbrauchs-Analysator* ist vom Funktionsumfang sehr ausgereift. Jedoch ist die Bedienung ganz und gar nicht intuitiv. Diese App erlaubt es beliebig viele Zähler anzulegen, sie speichert die Daten ebenfalls lokal und verfügt über eine Import/Export Funktion. Die Auswertungen werden wie bei der zuvor genannten App ebenfalls in Schriftform oder als Graph dargestellt.

3.2. Anforderungen

3.2.1. Gegebene Anforderungen

- Benutzer können mit ihrem Android Smartphone Zählerstände einlesen.
- Die Anwendung erkennt den gescannten / fotografierten Zähler automatisch.
- Das System speichert die erkannten Werte dauerhaft.
- Der Benutzer kann auf Auswerten der erfassten Daten zugreifen.

3.2.2. Anforderungen aus Marktanalyse

- Möglichkeit mehrere Zähler pro Typ anzulegen
- Intuitive und aufgeräumte UI[6]
- Speichern der Daten in der Cloud

3.3. OCR-Entscheidungen

3.3.1. Serverseitiges OCR

Vorteile:

- Server haben eine höhere Leistung als mobile Endgeräte
- Verbesserungen am OCR-Prozess würden kein Update der App mit sich ziehen
- Schnittstelle unabhängig vom OS der Endgeräts

Nachteile:

- Nutzer muss immer Online sein
- Bei schlechtem Netz lange Uploaddauer der Fotos
- Bei fehlerhafter Erkennung der Daten weitere Uploads
- wesentlich höherer Daten-Traffic bei Fotouploads im Gegensatz zu reinem Daten-Upload

[6] User Interface engl. für Benutzeroberfläche

- Rechenleistung komplett am Server, bei vielen Anwendern muss der Server dementsprechend leistungsfähig sein

3.3.2. Clientseitiges OCR

Vorteile:

- geringer Traffic
- geringe Uploaddauer da nur simple Daten übertragen werden müssen
- kein mehrfach Upload bei fehlerhafter Erkennung der Daten, da OCR Prozess direkt am Endgerät stattfindet
- Offline nutzbar
- Rechenleistung verteilt auf Endgeräte

Nachteile:

- Je nach OCR Bibliothek evtl. nicht Plattform unabhängig
- Änderungen am OCR Prozess würden Updates für Apps bedeuten
- Bei älteren Endgeräten evtl. geringe Rechenleistung
- geringere Leistung als leistungsfähige Server

3.3.3. Entscheidung Server vs. Clientseitiges OCR

Für dieses Projekt überwiegen folgende Vorteile des clientseitigen OCRs.

- geringer Traffic
- geringe Uploaddauer
- Offlinefunktionalität

3.3.4. Auswahl einer geeigneten OCR Bibliothek

Aufgrund der Verfügbarkeit einer Vielzahl von Open Source Bibliotheken wurde hauptsächlich in diese Richtung recherchiert. Auf der Website *findthebestopensource.com*[5] werden einige dieser Bibliotheken zusammen gefasst.

Die Wahl der OCR Bibliothek fiel nach einiger Recherche auf *Tesserat*[4]. Zum einen weil sie von Google selbst eingesetzt wird und es sich bei diesem Projekt um eine Android Anwendung handelt. Zum anderen weil sie bei mehreren Quellen[5][4][13] als eine der präzisesten OCR Engines genannt wird.

3.4. Architektur

Die Anwendung, die bei dieser Projektarbeit entstehen soll, soll eine Client-Server Architektur(Abbildung 2) besitzen.

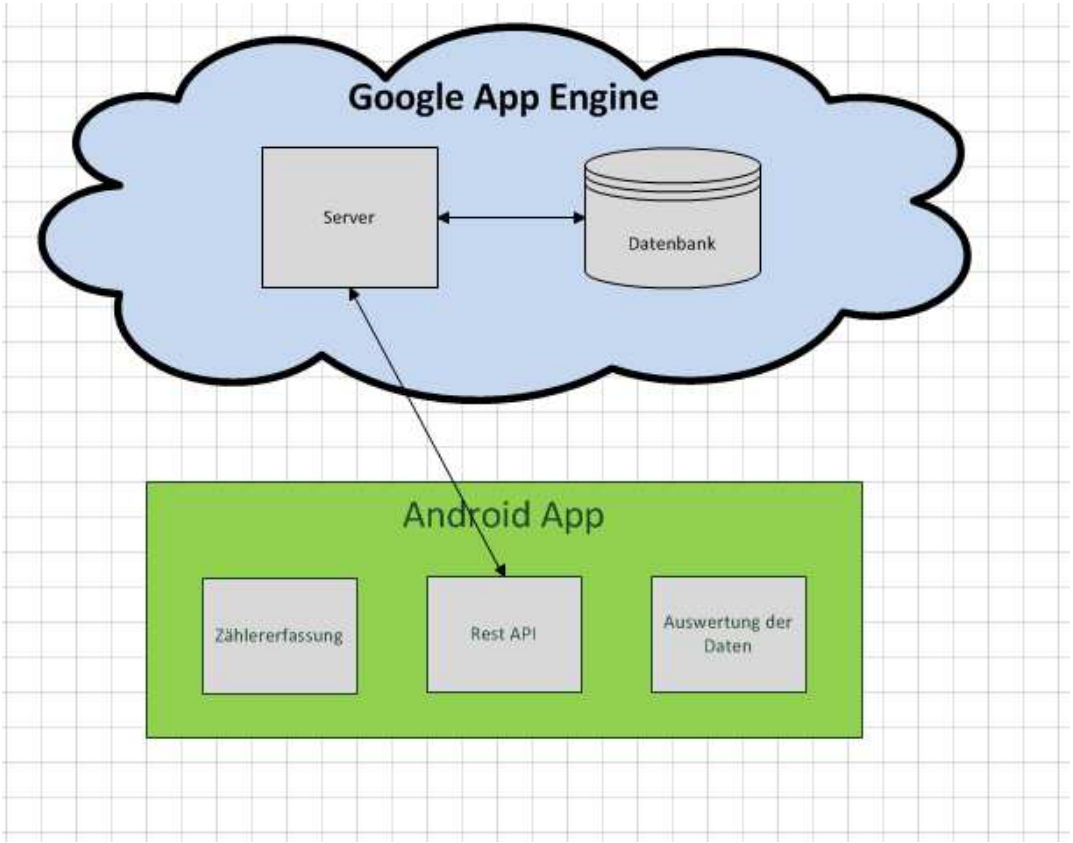

Abbildung 2: Client Server Architektur

3.4.1. Server

Der Server ist für die Speicherung und Bereitstellung der Zählerdaten zuständig. Das Datenbankmodell besteht, wie in Abbildung 3 beschrieben, aus den drei Entitäten User, Meter und MeterCount. Die Beziehung zwischen User zu Meter ist eine *bidirektionale 1 zu n* Beziehung. Ebenso die Beziehung zwischen Meter und MeterCount.

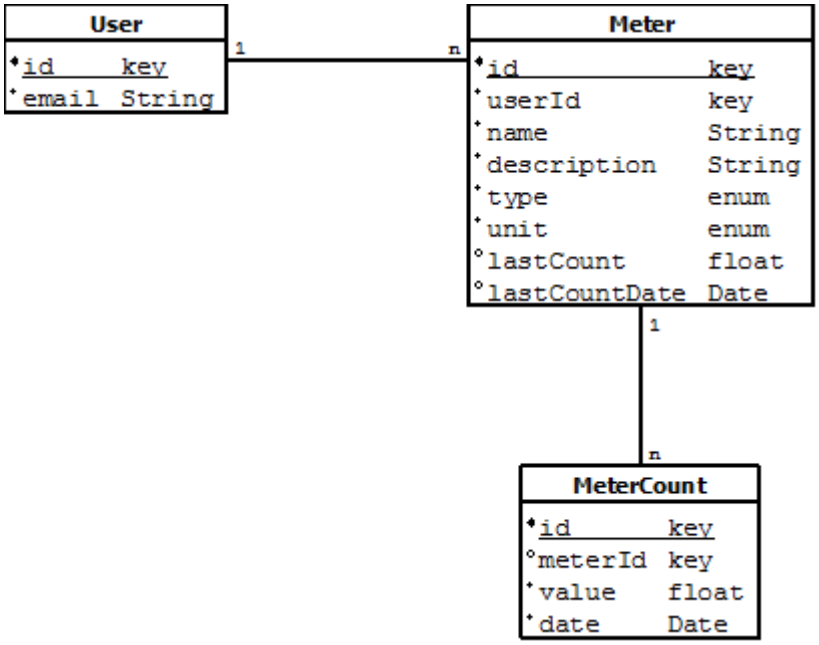

Abbildung 3: Datenbankmodell

Der Server stellt Endpoints für die App zur Verfügung. Diese Endpoints müssen die nötigen Funktionen zur Speicherung und Abfrage von Daten liefern.

Die Kommunikation erfolgt nach dem REST Prinzip. Als Rückgabe Format für die App wird JSON verwendet.

3.4.2. Android App

Die Android App soll folgende Use CasesAbbildung 4 erfüllen:

- Einen Zähler anlegen
- Einen Zählerstand zu einem Zähler anlegen
- Eine Auswertung der Zählerstände anzeigen

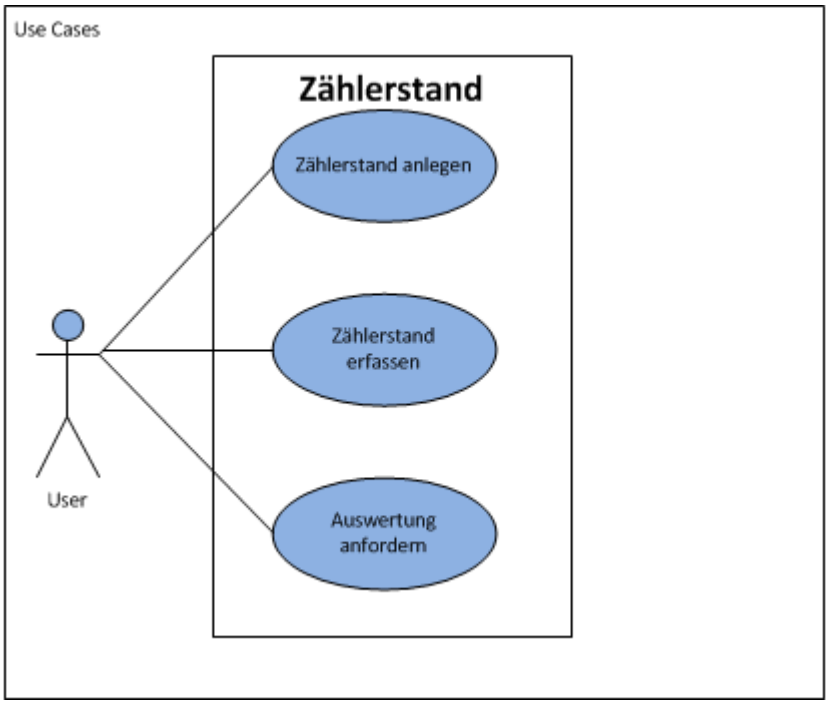

Abbildung 4: Use Cases

3.5. Wahl der Tools

Da es sich bei dieser Projektarbeit um eine Android Anwendung handelt, wird die von Google bereitgestellte Version der ADT (Android Development Tools) verwendet. Auf der offiziellen Seite[2] für Android Entwickler steht ein Bundle zum Download zur Verfügung. Dieses Bundle beinhaltet das Android SDK, das alle wichtigen API Libraries zum Bauen, Testen und Debuggen von Android Apps enthält. Zusätzlich beinhaltet es eine Version von Eclipse mit Integration der Android Development Tools.

Der Server soll mit Googles "App-Engine" realisiert werden. Die Entwicklung hierfür kann ebenfalls, mit Hilfe eines Plugins einfach in Eclipse vorgenommen werden.

Zur Versionsverwaltung wird Git verwendet. Der Einsatz von Git kann ebenfalls, durch das Installieren eines Plugins, einfach in Eclipse ermöglicht werden.

Als Ticketing System wird Picocloud Tracker eingesetzt. In diesem System werden Tasks erstellt und verwaltet. Zeitaufwände und Dokumentationen können hier ebenfalls erfasst werden.

4. Umsetzung

4.1. Ausschluss einer automatischen Erfassung der Zählerstände

Bei gezielter Recherche und einigen eigenen Tests zur Erfassung von Zählerständen via OCR ergab sich, dass es nicht möglich ist mit dieser Technik zuverlässig Zählerstände zu erfassen. Dies beschreibt zum Beispiel auch eine wissenschaftliche Arbeit der "University of Insubria"[1].

Laut dieser wissenschaftlichen Arbeit sind die Hauptprobleme hierbei:

- schlechte Auflösung der Kameras
- verwaschene Bilder, insbesondere Bewegungsunschärfe
- schlechte Belichtung
- niedriger Kontrast

Aus diesem Grund wurde zur Erfassung der Zählerstände eine manuelle Eingabe gewählt.

4.2. Server

4.2.1. Entitäten

Es wurden drei Entitäten angelegt:

- User
- Meter
- MeterCount

Sie wurden in JDO annotiert.

Das *Google Plugin für Eclipse* bietet die Möglichkeit automatisch Client Endpoints der angelegten Entitäten zu generieren. Die automatisch generierten Endpoints decken jedoch nur die einfachsten Funktionen ab und mussten um die benötigten Funktionen erweitert werden.

4.2.2. Endpoints

Die Endpoints wurden um folgende Funktionen erweitert:

- getUserByEmail
- getMeterListWithUserId
- insertMeterToUser
- getMeterCountListWithUserId
- insetMetrCountToUser

Die Funktion *getUserByEmail* macht eine Datenbank Abfrage nach einem User mit der übergebenen Email Adresse und gibt das User Objekt zurück.

getMeterListWithUserId fragt alle Zählerstände eines Benutzers ab und gibt diese in einer Liste zurück.

insertMeterToUser fügt zu einem über einen Parameter übergebenen User einen neuen Zähler ein.

getMeterCountListWithUserId gibt eine Liste von Zählerständen für einen bestimmten Zähler zurück.

insertMeterToUser fügt einen neuen Zählerstand zu einem Zähler hinzu.

4.3. Client

4.3.1. Welcome Activity

Der Client startet immer in der Welcome Activity. In dieser wird die Google Mail Adresse aus dem Gerät ausgelesen. Als weiteren Schritt wird in dieser Activity überprüft ob dieser Nutzer bereits auf dem Server angelegt wurde. Falls der Nutzer bereits existiert werden alle seine Zähler abgerufen. Existiert der Nutzer nicht wird er angelegt.

4.3.2. Home Activity

In der Home Activity werden alle zu einem Benutzer gehörenden Zähler, in Tabs nach Kategorien unterteilt, in einer Liste dargestellt. Jedes Listenelement besitzt zwei mögliche Arten der Interaktion. Bei Click auf das Element selbst wird ein asynchroner Auftrag gestartet der die Zählerstände zu diesem Element vom Server abfragt und die *MeterCount Activity* startet. Bei Click auf das *Plus* Symbol auf einem Element wird ein Dialog geöffnet der zur Eingabe eines neuen Zählerstandes auffordert. Bei Betätigung des *Save* Buttons wird der eingegebene Wert überprüft. Der Wert darf nicht den zulässigen Bereich des Datentyps *float* übersteigen. Genauso darf der Wert nicht niedriger sein als der zuletzt erfasste Zählerstand.

Bei positiver Überprüfung dieser Kriterien wird der neue Zählerstand an den Server übermittelt. Der User bekommt bei erfolgreicher Übertragung der Daten Feedback in Form eines *Toast*[7].

In der Action Bar der App befinden sich zwei weitere Buttons. Der *Refresh* Button aktualisiert die die Ansicht nach Anlegen eines neuen Zählers oder Zählerstandes.

Der *Plus* Button öffnet den Dialog zum Anlegen eines weiteren Zählers. Bei Tippen auf den Bestätigungsbutton werden die Felder des Dialogs ausgelesen und auf ihre Vollständigkeit überprüft. Sind alle Felder richtig ausgefüllt werden die Daten über einen weiteren asynchronen Auftrag zum Server übermittelt. Der Benutzer bekommt bei erfolgreicher Übertragung wieder eine Rückmeldung via *Toast.*

Die asynchronen Aufträge werden benutzt um während den Übertragungen von Daten an den Server nicht die Benutzeroberfläche zu blockieren. Ein solcher Auftrag läuft dann im Hintergrund ab bis er alle Schritte abgearbeitet hat.

4.3.3. MeterCount Activity

Diese Activity stellt alle Zählerstände eines bestimmten Zählers in einer Liste dar. Zusätzlich befinden sich in dieser Activity die Funktionen zur Berechnung der Auswertung.

[7] Nachricht auf dem Bildschirm

5. Ergebnisse

Das Kapitel Ergebnisse beschreibt die entstandene Software und deren Eigenschaften. Des Weiteren wird mit Hilfe von Screenshots der Anwendung die Bedienung erläutert.

5.1. Software

Das Ergebnis dieser Arbeit ist eine Android App mit Anbindung an einen auf *Google App Engine* gehosteten Server. In der Android App können beliebig viele und verschiedene Zähler angelegt werden. Diese werden dann auf dem Server abgespeichert. Die Zähler werden übersichtlich in einer Liste angezeigt. Die verschiedenen Zählerrubriken werden durch Tabs dargestellt.

Zu jedem Zähler können Zählerstände erfasst werden. Diese werden, in einer Detailansicht des Zählers, ebenfalls in einer Liste angezeigt. Dazu wird in dieser Ansicht eine Auswertung der Zählerstände gezeigt.

Das Design der App ist einfach gehalten. Dadurch wird eine intuitive Bedienung möglich.

5.2. Bedienung

Nach der Anmeldung am Server startet die App mit dem Home Screen(**Fehler! Verweisquelle konnte nicht gefunden werden.Fehler! Verweisquelle konnte nicht gefunden werden.**). Hier sieht man eine Übersicht über die angelegten Zähler. Diese Ansicht ist in Tabs aufgeteilt. Jeder Tab entspricht einer Rubrik von Zählern.

Durch eine Wischgeste oder Tippen auf einen bestimmten Tab kann zwischen den Zählerrubriken gewechselt werden.

Die Zähler werden in einer Listenansicht dargestellt. Für jedes Element in der Liste wird der Name und der letzte erfasste Zählerstand inklusive Datum angezeigt.

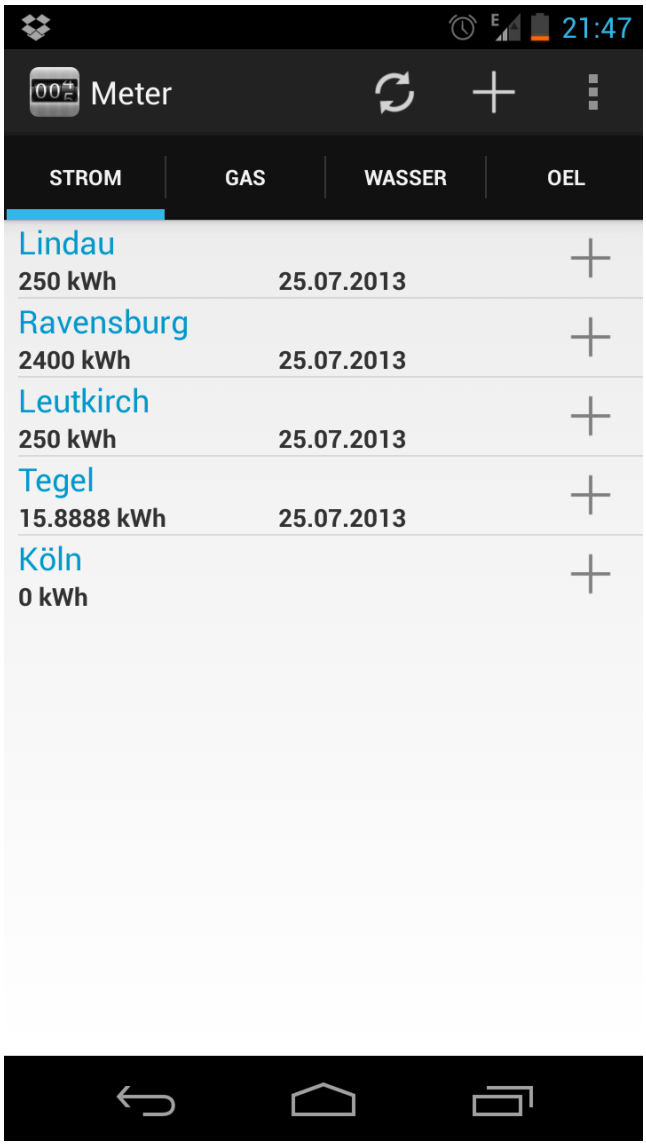

Abbildung 5 : Home Screen / Zählerübersicht

In der Action Bar befindet sich ein *Plus* Symbol. Durch tippen auf diesen Button öffnet sich ein Dialog (Abbildung 6) zum Anlegen eines neuen Zählers.

In diesem Dialog gibt es zwei Spinner[8] mit denen die Kategorie und die Maßeinheit der Zählers ausgewählt werden können.

Neben den Spinners enthält der Dialog noch zwei Textfelder um den Namen und einen Kommentar zum Zähler abzuspeichern.

Durch drücken auf den *Save* Button wird der neue Zähler angelegt.

[8] Android Drop Down Box

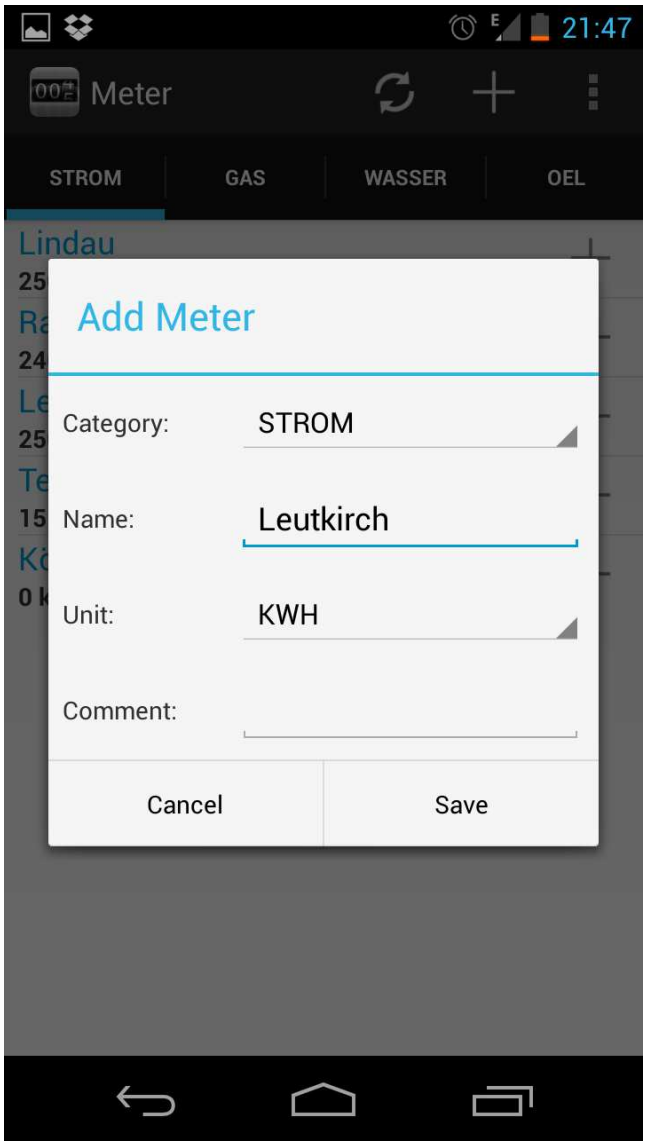

Abbildung 6: Zähler hinzufügen

Jedes Zählerelement in der Liste besitzt einen *Plus* Button. Durch Click auf diesen wird ein weiterer Dialog(Abbildung 7) geöffnet. Mithilfe von diesem Dialog wird ein neuer Zählerstand zu diesem Zähler hinzugefügt.

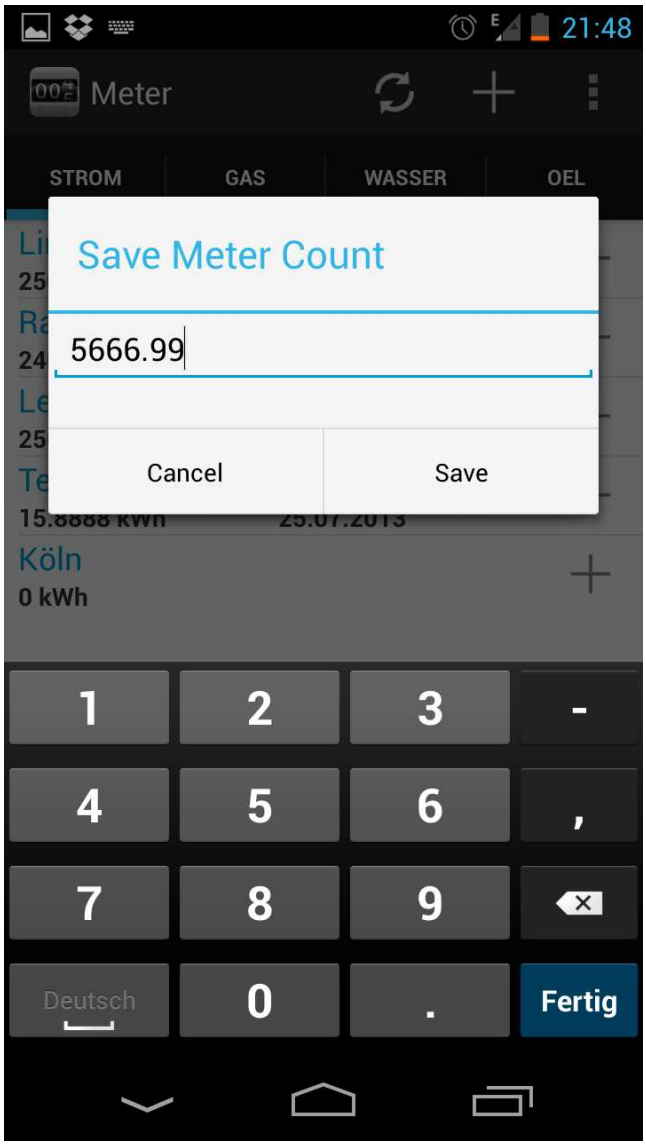

Abbildung 7: Zählerstand erfassen

Durch Tippen auf das Zählerelement selbst öffnet sich die Detailansicht(Abbildung 8) dieses Zählers.

In dieser Ansicht sind alle erfassten Zählerstände aufgelistet. Zusätzlich zu dieser Liste wird eine Auswertung der Zählerstände angezeigt.

Abbildung 8: Detailansicht für Zähler

6. Zusammenfassung und Ausblick

In diesem Kapitel wird die Projektarbeit kurz zusammengefasst und ein Ausblick auf mögliche, zukünftige Entwicklungen und Erweiterungen der Software gewagt.

6.1. Zusammenfassung

Das Ziel der Projektarbeit zur Konzeption und Entwicklung einer Anwendung, mit deren Hilfe Zählerstände aller Art erfasst und auf einem Server gespeichert werden, wurde erreicht.

Der Nutzer kann nun alle seine Zähler und die erfassten Zählerstände mit dieser Software verwalten. Durch die Speicherung der Daten auf einem Server sind sie selbst bei Verlust, Defekt oder Diebstahl des Android Geräts sicher gespeichert.

Durch Verwendung des Google Kontos kann die App auch mit anderen Android Geräten benutzt werden, wenn auf diesen Geräten derselbe Benutzer angemeldet ist. So kann mit mehreren Geräten auf derselben Datenbasis gearbeitet werden.

Leider konnte der Punkt des Testings, obwohl gefordert, nicht erfüllt werden. Dies liegt darin begründet, dass bei der Implementierung des Servers unerwartete Schwierigkeiten mit der Beziehung zwischen den Entitäten aufkamen. Durch diese Schwierigkeiten war ein Weiterkommen für circa zwei Wochen nicht möglich.

Nach der Lösung dieses Problems war die restliche Zeit, die für das Projekt zur Verfügung stand, zu knapp um sich mit der Testumgebung von Android auseinanderzusetzen und die übrige Implementierungsarbeit abzuschließen.

Außer diesem, gab es im weiteren Projektverlauf keine weiteren Probleme. Die Umgebung zur Entwicklung von Apps mit Server Anbindung, die von Google bereitgestellt wird, ist eine sehr durchdachte und gut ineinander greifende Lösung.

6.2. Ausblick

Das Ergebnis dieser Projektarbeit könnte um viele nützliche Features und Eigenschaften erweitert werden. In den folgenden Abschnitten werden einige kurz erwähnt und erläutert.

6.2.1. Offline Funktion

Viele andere gängige Android Apps wie zum Beispiel *Evernote* oder *Google Keep* funktionieren auch ohne durchgehende Internetverbindung. Diese Anwendungen synchronisieren die eventuell offline getätigten Änderungen zu einem späteren Zeitpunkt, an dem das Gerät wieder eine Verbindung zum Internet hat.

Eine solche Funktion wäre auch für die hier entstandene Anwendung sinnvoll. Hierfür müsste zusätzlich zur Datenbank auf dem Server mit der internen Datenbank des Android Systems gearbeitet werden und Daten bei gegebener Internetverbindung mit dem Server abgeglichen werden.

6.2.2. Authentifizierung

Um mehr Sicherheit für den Nutzer der Software zu bieten, könnte eine Authentifizierung mit dem Google Konto erfolgen.

6.2.3. Auswertung

Die Auswertung der gespeicherten Zählerstände erfolgt bei der in dieser Projektarbeit entstandenen Software momentan nur aufgrund von drei harten Kriterien:

- Verbrauch im aktuellen Monat
- Verbrauch im aktuellen Jahr
- Verbrauch im letzten Jahr

Um die Auswertungen für den Nutzer noch nützlicher zu machen, könnten weitere Algorithmen implementiert werden. Damit könnten dann zum Beispiel Verbräuche für bestimmte, und vom Nutzer selbst ausgewählte Zeitabschnitte errechnet und verglichen werden.

Eine weitere denkbare Erweiterung wäre eine Auswertung der Daten in Graphen. Hier könnte man dem Nutzer verschiedene Möglichkeiten bieten. Zum Beispiel einen Graphen mit allen von ihm angelegten Zählern eines bestimmten Typs wie *Wasser*. Damit hätte er einen grafischen Verlauf aller Wasserverbräuche auf einen Blick.

6.2.4. Web Interface

Eine weitere, für den Nutzer eventuell nützliche, Erweiterung wäre, wenn der Server um ein Web Interface erweitert werden würde. Durch dieses Web Interface wäre es dem Nutzer möglich, seine Daten auch von einem PC oder anderen webfähigen Geräten, die nicht Android basiert sind, zu verwalten.

6.2.5. Erinnerung

Eine Möglichkeit, die Benutzerfreundlichkeit zu erhöhen, wäre eine Erinnerungsfunktion. Mit dieser könnte der Nutzer in den Optionen der App einstellen, in welchem Intervall er gerne seine Zählerstände erfassen möchte. Die App würde den Benutzer dann zu den gegebenen Zeitpunkten mit einer Systemmeldung an die Fälligkeit einer Erfassung erinnern.

Auf diesem Wege müsste der Nutzer nicht selbst an die Erfassung der Zählerstände denken.

6.2.6. Kosten / Preise

Um Auswertungen nicht nur auf Basis der Einheiten der Zähler zu ermöglichen, könnte die Anwendung um Preise dieser Einheiten erweitert werden.

Entweder würde der Benutzer selbst die Preise, aus von den Providern bereitgestellten Listen, in der Anwendung ablegen, oder man könnte diese Daten direkt aus einer Datenbank im Internet bekommen.

Zusätzlich wäre hier denkbar, dass man dadurch Provider vergleichen könnte, um den günstigsten dieser zu ermitteln.

6.2.7. Provideranbindung

Da die meisten Zähler noch keine Daten zum Provider übertragen können, wäre es auch sinnvoll, in der App eine Anbindung zu seinem Provider zu ermöglichen. Dadurch würden die manuellen Übermittlungen von Zählerständen zukünftig wegfallen.

7. Literaturverzeichnis

[1] A Multi-Neural Network Approach to Image Detection andSegmentation of Gas Meter Counter, Ignazio Gallo, Juli 2013. http://www.academia.edu/2990878/A_Multi-Neural_Network_Approach_to_Image_Detection_and_Segmentation_of_Gas_Meter_Counter
[2] Android Developers, SDK Download, Juli 2013. http://developer.android.com/sdk/index.html
[3] CNET News, Shara Tibken, Juli 2013. http://news.cnet.com/8301-1023_3-57584484-93/google-android-activations-to-total-900-million-this-year/
[4] Code.google.com, tesseract-ocr, Juli 2013. https://code.google.com/p/tesseract-ocr/
[5] Find The Best Open Source, OCR, Juli 2013. http://www.findbestopensource.com/tagged/ocr
[6] Git, Homepage, Juli 2013. http://git-scm.com/
[7] Google Developers, Google Plugin for Eclipse, Juli 2013. https://developers.google.com/eclipse/?hl=de
[8] Google Developers, Warum App Engine, Juli 2013. https://developers.google.com/appengine/whyappengine?hl=de
[9] Google play, Energieverbrauchs-Analysator, Juli 2013. https://play.google.com/store/apps/details?id=at.topfen.ecas
[10] Google play, Homepage, Juli 2013. https://play.google.com/store
[11] Google play, Zählerstand, Juli 2013. https://play.google.com/store/apps/details?id=dk.ilios.meterreadings
[12] JSON, Homepage, Juli 2013. http://www.json.org/
[13] Wikipedia, Tesseract, Juli 2013. http://en.wikipedia.org/wiki/Tesseract_(software)
[14] ZDnet, Björn Greif, Juli 2013. http://www.zdnet.de/88154889/gartner-android-erreicht-fast-75-prozent-marktanteil/