Crowdsourcing von geo-getaggten Bildern für die Erkennung von Moskito-Brutstätten auf der Basis von mobilen Anwendungen

Artur Wojcik

Erstgutachter: Prof. Dr. Johannes Schöning

Zweitgutachter: Dr. Thomas Barkowsky

Fachbereich 03: Mathematik und Informatik

Universität Bremen

Diese Thesis wird für den Abschluss *Master of Science* eingereicht.

Universität Bremen Oktober 2018

Eidesstattliche Erklärung

Hiermit erkläre ich, Artur Wojcik, eidesstattlich, dass ich diese Arbeit selbstständig und nur mit Zuhilfenahme der angegebenen Quellen und Hilfsmittel angefertigt habe. Diese Arbeit wurde weder in gleicher noch in ähnlicher Form einem anderen Prüfungsamt vorgelegt oder veröffentlicht.

Artur Wojcik

Oktober 2018

Abstract

Das Denguefieber ist eine Viruserkrankung, die durch einen Stich der Gelbfiebermücke (lat. *Aedes Aegypti*) übertragen wird. Weltweit gibt es jährlich über 390 Millionen Infektionen bei Menschen. Das Dengue-Virus ist besonders in Südostasien verbreitet und gefährdet somit stark die Gesundheit der dort lebenden Menschen. Die Brutstätten der Überträger sind meistens künstliche Behälter (alte Reifen, Eimer oder Vasen). Es gibt verschiedene Ansätze um das Dengue-Virus zu überwachen und zu kontrollieren. So versucht das Projekt „Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition“, Brutstätten automatisiert zu detektieren. Dazu werden Bilder von Google-Street-View, mit Hilfe eines Objekterkennungs-Algorithmus, auf Brutstätten untersucht und klassifiziert. Ein Problem, welches dabei entsteht, ist die Aktualisierungsrate und Verfügbarkeit von Google-Street-View-Bildern. Im Rahmen dieser Arbeit werden drei mobile Anwendungen vorgestellt, welche dieses Problem angehen sollen. Mit Hilfe der Anwendungen können freiwillige Helfer die Bilderdatenbank aktualisieren. Bei den ersten zwei Anwendungen, handelt es sich um die Android-Apps: *Dengue-Detector* und *Mobile4D*. Diese ermöglichen es dem Nutzer Orte, an denen Bilder erstellt werden sollen, aufzufinden und zu Fotografieren. Des Weiteres wurde ein Facebook-Chatbot entwickelt, welcher die gleiche Funktionalität wie die *Dengue-Detector*- und *Mobile4D*-App hat. Durch die Plattformunabhängigkeit und die hohe Popularität des Facebook-Messenger werden höhere Nutzerzahlen als bei den herkömmlichen Apps erwartet. Im Rahmen dieser Arbeit werden die Konzepte sowie die Implementierung der drei Anwendungen vorgestellt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Das Denguefieber ist eine Viruserkrankung, die durch einen Stich der Gelbfiebermücke (lat. *Aedes Aegypti*, Abbildung 1.1) übertragen wird [1]. Weltweit gibt es jährlich über 390 Millionen Infektionen bei Menschen [2]. Das Dengue-Virus ist besonders in Südostasien verbreitet und gefährdet somit stark die Gesundheit der dort lebenden Menschen [3]. Um die Verbreitung des Dengue-Virus besser überwachen zu können, gibt die *World Health Organization* (WHO) verschiedene Indikatoren an [4]: Der *House Index* (HI) gibt an, wie viel Prozent der Haushalte von Larven der Gelbfiebermücke befallen sind. Der *Container Index* (CI) gibt den prozentualen Anteilen an Behältern an, in welchen sich Larven befinden. Der *Breteau Index* (BI) gibt die Anzahl an befallenen Behältern auf 100 Haushalte an. Diese Indices sollen dabei helfen, Annahmen über einen möglichen Ausbruch zu treffen und Gegenmaßnahmen einzuleiten. Um diese Indikatoren erstellen zu können, werden die Brutstätten (BS) der Gelbfiebermücke kartiert [4]. Die BS der Gelbfiebermücken sind meistens künstliche Behälter, wie zum Beispiel alte Reifen, Eimer oder Vasen (siehe Abbildung 1.1). Die Kartierung erfolgt zumeist manuell, durch Mitarbeiter des Gesundheitsministeriums (siehe Abbildung 1.1). Dabei werden einzelne BS auf Larven untersucht. Wegen der großen Anzahl an potentiellen BS, ist die Kartierung aufwendig und schlecht skalierbar [5]. Da

Abbildung 1.1 Die Gelbfiebermücke (links), Brutstätte der Gelbfiebermücke (mitte), Kartierung einer Brutstätte (rechts).

manuelle Untersuchungen nur stichprobenartig und nicht großflächig vorgenommen werden, kann dies zu ungenauen und fehlerbehafteten Ergebnissen führen.

Neben dem herkömmlichen Ansatz gibt es verschiedene Projekte, welche die manuelle Kartierung durch Informationstechnologien versuchen zu unterstützen. Dazu werden beispielsweise Satellitenbilder auf BS untersucht [6]. Mobile Apps werden genutzt, um BS mit Hilfe der Bevölkerung zu finden [7] oder es werden Vorhersagemodelle anhand von Wetterdaten erstellt [8]. Diese Arbeit baut auf einem Projekt auf, welches an der Mahidol University in Thailand stattfindet. Beim Projekt „Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition“ [in press], handelt es sich um einen Ansatz, die Kartierung von BS zu automatisieren. Dabei werden geo-getaggte Bilder auf BS untersucht und mit Hilfe von *Machine Learning* klassifiziert. Kann eine Kausalität mit den Indikatoren der WHO gefunden werden, so bietet dieser Ansatz eine effektive Alternative zu der manuellen Kartierung. Auf diese Weise erkannte BS (siehe Abbildung 1.2) lassen sich auf einer Heat-Map darstellen (siehe Abbildung 1.3). Um das System mit Bildern zu versorgen, wird Google-Street-View (GSV) genutzt. GSV hat den Vorteil, dass in allen Regionen eine flächendeckende Verfügbarkeit vorhanden ist, hat jedoch den Nachteil der geringen Aktualisierungsrate und geringer Verfügbarkeit von Bildern auf privaten Grundstücken oder auf schwer erreichbaren Straßen. Veraltete oder nicht vorhandene Bilder müssen durch andere Quellen ergänzt werden, da potenzielle BS sich oft an derartigen Orten befinden. Dazu wurde eine mobile App (*Dengue-Detector*) entwickelt. Diese ermöglicht es freiwilligen Helfern, veraltete Bilder zu aktualisieren und nicht vorhandene zu

Abbildung 1.2 Durch das System erkannte Reifen

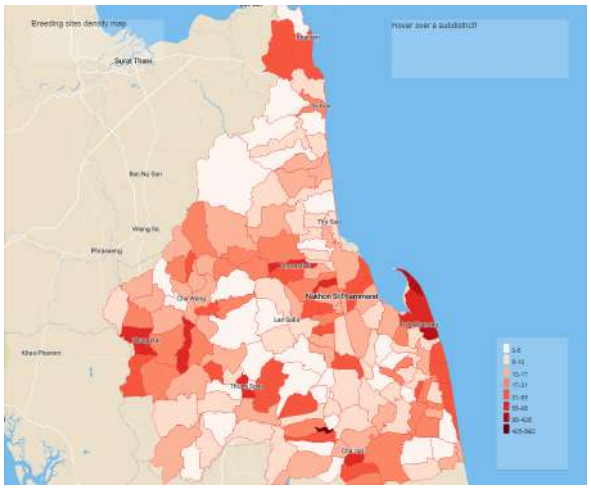

Abbildung 1.3 Darstellung der potenziellen BS auf einer Heat-Map

erstellen. Ein solcher Ansatz wird in der Fachliteratur oft als *(Geo-Spatial-) Crowdsourcing* bezeichnet.

Eine große Herausforderung an *Crowdsourcing* ist das Erreichen einer angemessenen Anzahl an aktiven Teilnehmern [9]. Besonders nicht kommerzielle Projekte haben Schwierigkeiten damit, Teilnehmer zu motivieren, die ohne monetäre Anreize bereit sind am Projekt teilzunehmen. Des Weiteren sind Nutzer oft nicht bereit, eine App zu installieren und regelmäßig zu nutzen [30]. Besonders in Asien, wo Desktoprechner keine hohe Popularität haben und vieles mit Hilfe von mobilen Geräten erledigt wird, sind Messaging-Plattformen von großer Beliebtheit [31]. Die Zahlen der monatlich aktiven Nutzer von Messaging-Plattformen haben die Nutzerzahlen von E-Mails und Websites überholt [32]. Der in Südostasien weit verbreitete *Facebook-Messenger* gehört zum Standard der digitalen Kommunikation. Mit einem Anteil von über 60 Prozent ist dieser marktführend [33]. Eine Besonderheit des *Facebook-Messenger* ist die Möglichkeit der Anbindung von Chatbots. Dadurch ist es möglich, eine Konversation zu automatisieren und somit die Funktionalität der *Dengue-Detector-App* zu implementieren.

1.2 Ziel der Arbeit

Im Rahmen dieser Arbeit wurde zum einem die *Dengue-Detector*-App entwickelt, welche dem Zweck dient, Bilder an Orten zu erstellen, an denen GSV nur veraltete oder keine Bilder zur Verfügung stellt. Des Weiteren wird die mobile *Dengue-Detector*-App in das, an der Universität Bremen entwickelte, Reporting-Alerting-System *Mobile4D* integriert. *Mobile4D* befindet sich momentan in Laos im Einsatz und dient zum Melden von Umweltkatastrophen. Eine weitere Aufgabe ist der Entwurf und die Implementierung eines Chatbots. Dieser dient, wie die *Dengue-Detector*-App, zum Erstellen von geo-getaggten Bildern. Zum Schluss erfolgt die Integration der drei Anwendungen mit der Bilderdatenbank, die aktualisiert werden soll und der Datenbank, welche Informationen darüber enthält, wo Bilder veraltet oder nicht vorhanden sind.

1.3 Aufbau der Arbeit

Kapitel 1 (Einleitung) gibt einen Überblick über die Problemstellung und über Ansätze, um diese zu lösen. Das Kapitel 2 (Verwandte Arbeiten) setzt sich mit verschiedenen Publikationen zu den Themen Dengue-Fieber und IT, Chatbots im Gesundheitswesen und *Crowdsourcing* auseinander. Am Ende jedes Abschnitts werden die Veröffentlichungen verglichen und diskutiert. Konzepte und Ideen der *Dengue-Detector*-App, der *Mobile4D*-App und des Facebook-Chatbots werden im Kapitel 3 (Konzepte) besprochen. Im Kapitel 4 (Implementierung) wird auf die Integration der Anwendungen mit der Bilderdatenbank und der *Spot*-Datenbank eingegangen. Es wird ebenfalls die Implementierung der drei Anwendungen thematisiert. Zum Schluss wird im Kapitel 5 (Fazit und Ausblick) über die gewonnenen Rückschlüsse diskutiert. Des Weiteren werden dort mögliche Themen und Ideen vorgestellt, die zu weiteren Forschungen und Untersuchungen auf diesem Gebiet führen können.

1.4 Komplementärarbeit von Marcel Dechert

Parallel zu dieser Arbeit, entsteht die Arbeit „Implementation and Evaluation of a Chatbot to Crowdsource Geo-Tagged Images to Detect Mosquito Breeding Sites“ von Marcel Dechert. Diese beschäftigt sich ebenfalls mit dem *Crowdsourcing* von geo-getaggten Bildern für die Erkennung von Moskito-Brutstätten. Als Basis dafür dient der, in dieser Arbeit vorgestellte Chatbot. Der gemeinsame Teil der Arbeit besteht aus folgenden Punkten (siehe Abbildung 1.4 Label 1):

- Einleitungskapitel der jeweiligen Arbeiten wurden zusammen verfasst.
- Die Literaturrecherche wurde gemeinsam vorgenommen, jedoch wurde diese voneinander unabhängig aufbereitet.
- Auswahl der Chatplattform.
- Auswahl des Natural-Language-Processing-Frameworks Dialogflow.
- Aufbau der grundlegenden technischen Struktur.
  - Verbindung zwischen Facebook-Messenger und Dialogflow.
  - Einrichten des Webhooks zwischen Dialogflow und dem Python-Server.
  - Verbindung zu der Spot- und Bilderdatenbank.
- Implementierung der Basis-Intents.

Im Zuge der Arbeit von M. Dechert wird unter anderem eine Evaluation des Chatbots vorgenommen. Neben der Evaluation wird auch die Konversation zwischen dem Nutzer und dem Chatbot erweitert. Dazu zählt das Bereitstellen von Informationen über den Verlauf der Krankheit und Informationen über die Überträger von Dengue. Des Weiteren werden Statistiken über Dengue-Fälle in verschiedenen Provinzen bereitgestellt. Es werden auch Gamification-Aspekte eingebracht. Mit wöchentlichen Herausforderungen und einer Bestenliste sollen Nutzer animiert werden, den Chatbot regelmäßig zu nutzen. Die Evaluation des Chatbots wird an der Mahidol University in Thailand vorgenommen. Im Rahmen dieser

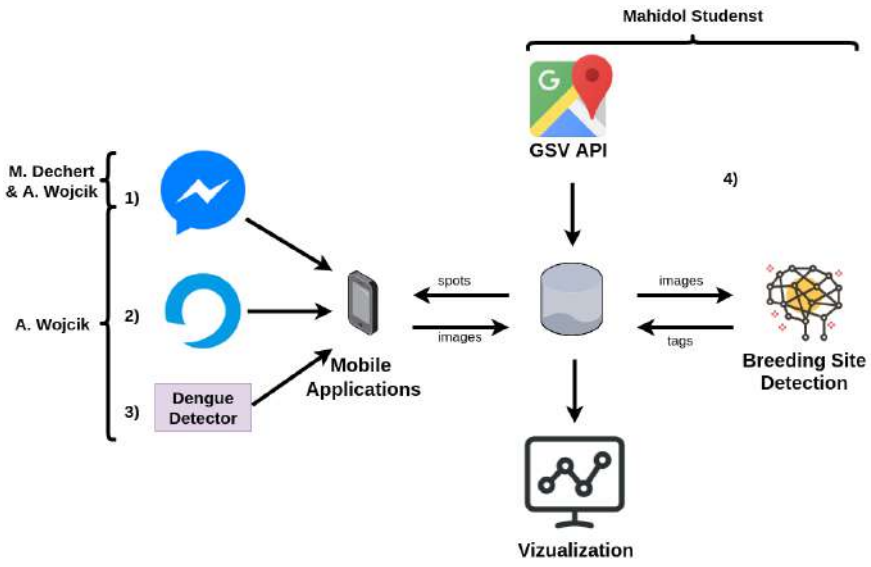

Abbildung 1.4 Abbildung zur Komplementärarbeit

sollen Studierende mit Hilfe des Chatbots die Bilderdatenbank aktualisieren. Die in dieser Arbeit vorgestellte *Dengue-Detector*-App sowie die Integration des Dengue-Moduls in *Mobile4d* (siehe Abbildung 1.4 Label 2 und 3) werden in der Arbeit von M. Dechert nicht berücksichtigt. Das System zur automatisierten Erkennung von BS wurde von Studenten der Mahidol University entwickelt.

1.5 Bildrechte

In dieser Arbeit verwendeten Grafiken sind gemeinfrei oder wurden vom Autor erstellt. Grafiken auf diese dies nicht zutrifft wurden referenziert.

Kapitel 2

Verwandte Arbeiten

Die Literaturrecherche liefert zahlreiche Artikel und Systeme die sich mit der Überwachung und Kontrolle von Dengue beschäftigen. Diese werden im Abschnitt 2.1 erläutert und anschließend diskutiert. Im Abschnitt 2.2 wird auf Chatbot-Technologien eingegangen. Dabei werden Chatbots untersucht, die im Bereich des Gesundheitswesens eingesetzt werden. Zum Schluss wird im Abschnitt 2.3 auf das Thema des *Crowdsourcing* eingegangen.

2.1 Dengue

Neben traditionellen Methoden wie Umweltmanagement, Larvenbekämpfung mit Hilfe von Chemikalien, Insektiziden oder allgemeinem Haushaltsschutz (z.B. Moskitonetze) [4], gibt es zahlreiche Methoden, die durch Informationstechnologien unterstützt werden. Diese lassen sich in verschiedenen Interventionspunkten anwenden. Veröffentlichungen [6], [10], [11] beschäftigen sich mit der Erkennung von BS, bevor die Larven der Gelbfiebermücke geschlüpft sind und es noch die Möglichkeit gibt, diese zu töten. Eine andere Maßnahme ist es, wie in Veröffentlichungen [8], [12], [13], [14] beschrieben, Vorhersagemodell zu erstellen. Dadurch kann ermittelt werden, an welchen Orten und zu welcher Zeit ein Dengue-Ausbruch stattfinden könnte. Der letzte, in dieser Arbeit betrachtete Ansatz, ist das Melden von BS sowie Dengue-Fällen mit Hilfe von mobilen Apps. Dieser wird in den Veröffentlichun-

Abbildung 2.1 Insektizid in Form einer Moskitospiralle

Abbildung 2.2 Moskitonetz als Präventionsmaßnahme

gen [7], [15], [16], [17] betrachtet. Im Folgenden wird auf die Veröffentlichungen näher eingegangen.

In der Arbeit von Chang et al. [6] werden Satellitenbilder von Google Earth genutzt, um potenzielle BS wie alte Reifen, Friedhöfe oder Flächen mit stehendem Wasser zu detektieren. Diese werden dann auf einer Karte eingezeichnet. Zusätzlich wurden auch Haushalte von Dengue-Patienten von *Public Health Worker* (PHW) ausfindig gemacht und in der Karte eingetragen. Chang et al. beschreiben das System als ein günstiges, jedoch aufwendiges Werkzeug, welches sich in Entwicklungsländern gut einsetzen lässt, um das Dengue-Virus zu überwachen.

Einen anderen Ansatz stellen Reich et al. [8] in ihrer Arbeit vor. Es handelt sich dabei um das Erstellen eines Vorhersagemodells. Dazu nutzten die Autoren alle, vom Jahr 1968 bis zum Jahr 2014, bekannten Dengue-Fieber Fälle. Diese Daten wurden in einer SQL-Datenbank eingepflegt und in Trainings- und Testdaten eingeteilt. Das System zeigte prägnante Ergebnisse für Vorhersagen für kurze Zeitperioden (2 Wochen). Für längere Zeitperioden (10 Wochen) waren die Ergebnisse ungenauer.

Aburas et al. [12] beschreiben ein System, welches ein Dengue-Ausbruch mit Hilfe von neuronalen Netzen vorhersagen soll. Die Vorhersage basiert auf drei Parametern: Durchschnittliche Temperatur, durchschnittliche Luftfeuchtigkeit und totaler Niederschlag. Die Daten bestanden aus 330 Datensätzen mit den genannten Parametern. Die vorhandenen Daten deckten einen Zeitraum vom Jahr 2001 bis zum Jahr 2007 ab. Diese Datensätze wurden in Trainingsdaten (Schulung der neuronalen Netze) und Testdaten (Verifikation des Systems)

unterteilt. Die Ergebnisse des Systems waren zufriedenstellend, jedoch wurde der Ausbruch im Jahre 2005 in Singapur nicht vorhergesagt. Laut dem Autor sind Ereignisse eingetreten, die sich mit 3 Parametern nicht erklären lassen.

Mehra et al. [10] versuchen in ihrer Arbeit, BS mit Hilfe der Objekterkennung zu detektieren. Dabei werden Bilder aus verschiedenen Quellen wie Google, Flickr oder selbst erstellte Bilder, auf Wasserstellen untersucht. Der erste Teil der Arbeit fokussiert sich auf die Erkennung von sichtbaren Wasserstellen. Diese Bilder wurden mit herkömmlicher Kamera aufgenommen. Der zweite Teil beschäftigt sich mit Wasserstellen, die mit Hilfe einer Thermokamera aufgenommen wurden. Beide Datensätze werden mit Hilfe von Objekterkennung ausgewertet. Die durch das System erkannten Wasserstellen werden in einer Heat-Map eingezeichnet.

Oft sind jedoch BS an Stellen vorhanden, die nicht sichtbar oder schwer zugänglich sind. Sudawella et al. [11] stellen in ihrer Veröffentlichung eine Methode vor, um BS mit Hilfe von Drohnen ausfindig zu machen. Die gesammelten Bilder werden, ähnlich wie schon in der Veröffentlichung von Mehra et al. [10], durch Objekterkennung auf Wasserstellen untersucht. Erstellte Bilder werden dann von einem Public Health Inspector auf BS untersucht. Nach der Verifizierung werden potenzielle BS auf einer Karte eingezeichnet.

Einen anderen Ansatz zur Vorhersage von Dengue-Fieber Fällen haben Carneiro und Mylonakis [13] in ihrer Arbeit vorgestellt. Diese haben Google-Suchanfragen genutzt, um Vorhersagen zu Viruserkrankungen machen zu können. Das System zeigte hohe Korrelation für die in der Arbeit untersuchten Krankheiten „West Nil Virus“, „Humane Respiratorische Synzytial-Virus“ und „Vogelgrippe“. Eine Schwäche, die das System aufweist ist, dass von Nutzern eingegebenen Suchanfragen nicht einheitlich sind und eine genaue Analyse nicht möglich war. Eine weitere Schwäche ist die falsche Vorhersage von Krankheiten. So wurde die Krankheit „Vogelgrippe“ in Regionen vorhergesagt, in denen es keinen Ausbruch gab. Die beiden Autoren sehen trotz der Schwächen eine Möglichkeit, Krankheitsausbrüche schneller vorherzusagen als herkömmliche Methoden.

Einen ähnlichen Ansatz wählten Althouse et al. [14] mit dem Unterschied, dass nur von vornherein festgelegte Suchbegriffe zur Auswertung genutzt wurden. Die Auswertungen

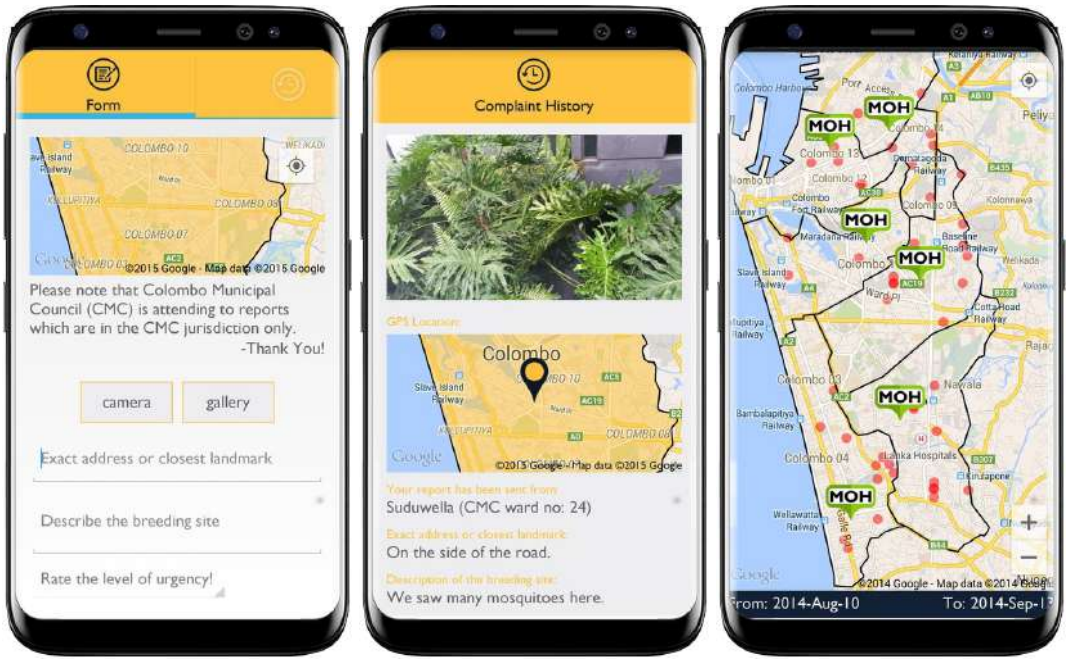

Abbildung 2.3 Mo-Buzz: (links) Leeres Formular zum Melden von BS, (Mitte) Ausgefülltes Formular, (rechts) Auf einer Karte eingezeichnete BS

bezogen sich auf Dengue-Fieber-Suchanfragen in Thailand und Singapur. In beiden Ländern gab es eine hohe Korrelation mit tatsächlichen Fällen.

Im Laufe der Recherche wurden auch mehrere mobile Web-Anwendungen untersucht. Die Veröffentlichungen von Lwin et al. [7], [15] präsentieren ein Social Media System (Mo-Buzz, siehe Abbildung 2.3). Dieses dient zum Melden von Dengue-Fällen in Südostasien. Das System gibt es in zwei Versionen: Eine Version für offizielle Public Health Worker (PHW) und eine für die Bevölkerung. PHW können dadurch BS oder betroffene Haushalte effizient melden. Es ist auch möglich Bilder hochzuladen, welche dann auf einem Server manuell auf BS untersucht werden. Eine Kartendarstellung ist ebenfalls vorhanden. Die zweite Version erlaubt es der Bevölkerung die Heat-Map anzuschauen, sowie sich über Dengue zu informieren. Um die PHW zu motivieren, wurden jährliche Boni, sowie Gehaltserhöhungen in Aussicht gestellt. Dies führte zur Steigerung der Nutzerzahlen.

Wojcik et al. [16] vergleichen in ihrer Arbeit mehrere Systeme zum Melden von Infektionskrankheiten und gehen auf deren Stärken und Schwächen ein. Darunter befindet sich auch das System „Dengue na Web“, welches zum Melden von Dengue-Fällen dient. Der

Nutzer kann online einen infizierten Haushalt melden. Zusätzlich bietet das System eine Echtzeit-Kartendarstellung sowie Lernmaterial an. Wojcik et al. sehen solche Systeme als eine schnellere und günstigere Variante, Krankheiten zu melden und gefährdete Gebiete zu überwachen. Alle untersuchten Systeme weisen eine hohe Übereinstimmung mit Daten aus offiziellen Quellen auf. Das Projekt wurde in Salvador de Bahia, in Brasilien durchgeführt.

Quadri et al. [17] entwickelten eine mobile Web-Anwendung (Target Zika) zum Melden von BS in Brasilien. Das System nutzt das Prinzip des *Micro-Reporting*. Mit der App ist es dem Nutzer möglich, BS zu melden, zu fotografieren und weitere Informationen hinzuzufügen (Art, Zeit, Ort). Diese Informationen werden auf einem Server verarbeitet und analysiert. Die gemeldeten Fälle lassen sich dann auf einer Karte darstellen.

Das Prinzip des *Micro-Reporting* wird in der Arbeit von Pongpaichet et al. vorgestellt [18]. *Micro-Reports* haben folgenden Eigenschaften:

- **Spontan**: Reports sollten schnell erstellt werden. Unnötiges Tippen und Nachdenken soll vermieden werden.
- **Objektiv**: Reports sollen auf Fakten basieren und nicht auf Meinungen der Nutzer.
- **Universell**: Micro-Reports brauchen eine universelle Sprache, unabhängig von der Landessprache oder Region. Dazu eignen sich Fotografien sehr gut.

Die Autoren sehen Bilder als ein perfektes Werkzeug, um *Micro-Reports* zu erstellen. Der Informationsgewinn ist bei Bildern höher als bei anderen Quellen wie z.B. Tweets.

Das Prinzip des *Micro-Reporting* wird auch von der App *Veta* (www.veta.life) genutzt. Veta ist ein Reporting-System für Viruserkrankungen wie das Dengue-Fieber, Zika-Virus oder Malaria. Mit einem Report kann man die Art einer Krankheit, dessen Symptome, sowie den Ort angeben. Die Benutzeroberfläche ist in Abbildung 2.4 zu sehen. Zusätzlich kann angegeben werden, ob eine ärztliche Bestätigung der Erkrankung vorliegt. Die gemeldeten Krankheitsfälle werden auf einer Karte angezeigt. Es besteht auch die Möglichkeit, einen Beitrag für die Nachbarschaft zu erstellen. Die App verfügt auch über einen Chatbot, jedoch funktioniert die Kommunikation nur unidirektional vom Chatbot zum User.

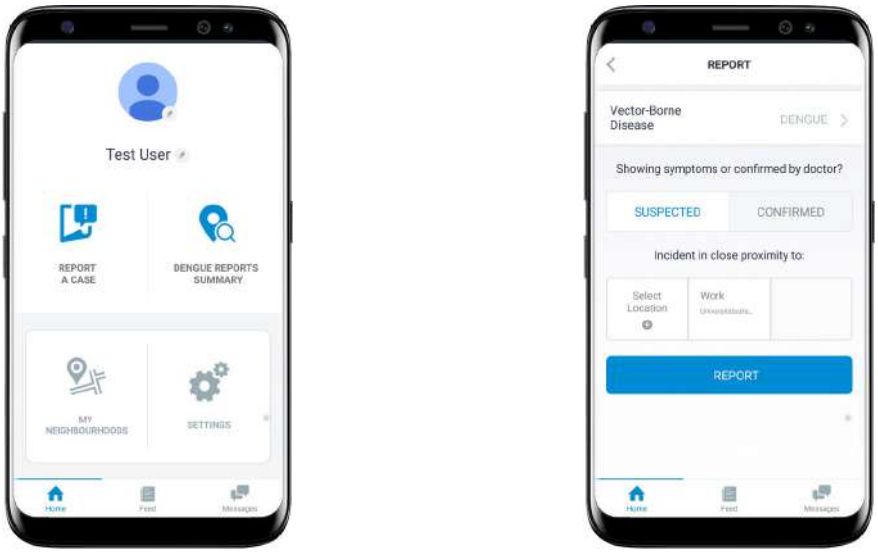

Abbildung 2.4 Benutzeroberfläche der Dengue-Reporting-App Veta

Ein weiteres System ist der *DengueChat* (www.denguechat.org). Es handelt sich hierbei um eine Chat-Plattform zur Überwachung und Kontrolle vom Dengue-Fieber und dessen Überträger. Das System entstand in einer Zusammenarbeit der brasilianischen Regierung und der Berkeley Universität in Kalifornien, USA. Nutzer können BS, Moskito-Hotspots oder Krankheitsfälle über einen Chat melden und sich darüber untereinander austauschen. Zusätzlich bietet das System Informationen darüber, wie man eine bestimmte BS beseitigen kann. Es besteht auch die Möglichkeit *Micro-Blogs* zu erstellen. In einem *Micro-Blog* kann der Nutzer mit der Nachbarschaft kommunizieren und über verschiedene Maßnahmen diskutieren. Durch ein Punktesystem werden Nutzer für das Erstellen von *Micro-Blogs* belohnt.

Alle oben betrachteten Veröffentlichungen beschäftigen sich mit der Überwachung und Kontrolle von Viruserkrankungen mit Hilfe von IT. Diese unterscheiden sich jedoch von ihren Herangehensweisen und Konzepten. Tabelle 2.1 soll einen Überblick über die Technologien geben und einen Vergleich schaffen. Beim Vergleich werden folgende Aspekte berücksichtigt:

- Einsatz von Geoinformationssystemen (GIS)
- Echtzeit-Kartendarstellung
- Objekterkennung

- Crowdsourcing
- Mobile-App

Autor	GIS	Echtzeit-Karte	Objekterkennung	Crowdsourcing	Mobil
WHO [4]	✓	(✓)	-	-	-
Chang et. al. [6]	✓	✓	✓	-	-
Reich et. al. [8]	✓	✓	-	-	-
Aburas et. al. [12]	-	✓	-	-	-
Mehra et.al. [10]	✓	✓	✓	-	-
Suduwella et. al. [11]	✓	✓	✓	-	-
Carneiro [13]	-	✓	-	(✓)	-
Althouse et. al [14]	-	✓	-	(✓)	-
Mo-Buzz [7]	✓	✓	-	-	✓
Dengue na Web [16]	-	✓	-	✓	-
Target Zika [17]	✓	✓	-	✓	✓
Veta [34]	✓	✓	-	✓	✓
DengueChat [35]	-	✓	-	✓	-
Dengue-Detector-App	✓	✓	✓	✓	✓

Tabelle 2.1 Vergleich von Technologien, die sich mit der Überwachung und Kontrolle des Dengue-Virus beschäftigen.

Wie es in der Tabelle 2.1 zu erkennen ist, erfüllt keines der untersuchten System alle fünf Aspekte. Zum größten Teil liegt dies daran, dass die meisten Systeme nicht mobil sind. Lediglich 3 der 13 (ausgenommen der *Dengue-Detector*-App) untersuchten Systeme bieten die Möglichkeit der Mobilität: Mo-Buzz, Target-Zika und Veta. Diese unterstützen jedoch nicht die automatisierte Erkennung von BS. Bilder werden entweder ohne nähere Untersuchungen auf einen Server hochgeladen oder durch offizielle Mitarbeiter analysiert. Dies ist jedoch sehr aufwändig und schlecht skalierbar. Das Thema der automatisierten BS-Erkennung wird in den Veröffentlichungen von Chang et al. [6], Mehra et al. [10] und Suduwella et al. [11] berücksichtigt. Dazu werden Satellitenbilder von Google, geo-getaggte Bilder von Flickr oder Drohnennaufnahmen auf BS untersucht. Die Ergebnisse dieser Systeme weisen hohe Korrelation mit Zahlen aus offiziellen Stellen auf. Der Ansatz der automatisierten Erkennung von potenziellen BS ist vielversprechend. Dieser wurde auch im Rahmen des *Dengue-Detector*-Projekts berücksichtigt. Mehr als die Hälfte der Systeme

(8 von 13) machten Gebrauch von GIS. Dies hat den Vorteil, dass ein genauer Standort der BS ermittelt werden kann, um diese auf einer Karte darzustellen. Carneiro [13] und Althouse et al. [14] nutzten für das Erstellen von Vorhersagemodellen Google-Suchanfragen, die sich mit Dengue beschäftigen. Wie es in Tabelle 2.1 zu sehen ist, wurden beide Systeme in der Spalte *Crowdsourcing* mit einem (✓) versehen. Dies bedeutet, dass das *Crowdsourcing* auf indirektem Wege geschieht. Es ist ein interessanter Ansatz, jedoch wird diese Methode in dieser Arbeit nicht weiter berücksichtigt. Der letzte untersuchte Ansatz ist das Melden von BS und Moskito-Hotspots mit Hilfe einer Chat-Plattform. Das System *DengueChat* erlaubt es den Nutzern, über einen Chat BS zu melden. Um die Bevölkerung zu motivieren, wurde ein Belohnungssystem eingeführt. Das Problem welches bei einer Chat-Plattform entsteht, ist dass Chatunterhaltungen nicht objektiv sind. In der letzten Zeile der Tabelle ist die *Dengue-Detector*-App aufgeführt, welche im Rahmen dieser Arbeit entwickelt wird. Wie zu sehen ist, soll diese Alle fünf Aspekte erfüllen.

Die Recherche lieferte viele Möglichkeiten, um Dengue mit Hilfe von Informationstechnologien zu überwachen und zu kontrollieren. Für diese Arbeit werden zwei Ideen berücksichtigt, um alle fünf Aspekte aus der Tabelle 2.1 zu erfüllen. Das an der Mahidol University in Thailand entwickelte System *Dengue-Detector* kann bereits BS mit Hilfe der Objekterkennung ausfindig machen und diese auf einer Karte darstellen. Eine Schwäche des Systems ist, dass lediglich GSV-Bilder als einzige Informationsquelle dienen. Aus diesem Grund wurden im Rahmen dieser Arbeit drei mobile Anwendungen entwickelt, welche als eine weitere Informationsquelle dienen sollen.

2.2 Chatbots im Gesundheitswesen

Chatbots sind keine neuen Technologien, die erst in den letzten Jahren entwickelt wurden. Im Jahr 1966 stellte Joseph Weizenbaum den ersten Chatbot *Eliza* vor [19]. Dieser simulierte eine Psychotherapeutin im Gespräch mit einem Patienten. Das Prinzip des Chatbots war sehr einfach: Das Programm durchsuchte den eingegebenen Satz nach Schlüsselwörtern. Je nach Schlüsselwort und Kontext wurde eine Antwort von *Eliza* generiert (Abbildung 2.5). Heut-

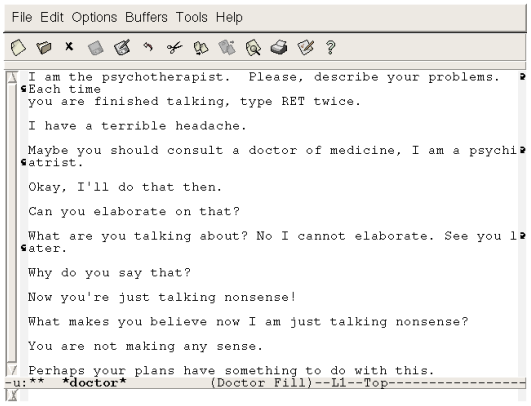

Abbildung 2.5 Konversation mit Eliza [36]

Abbildung 2.6 Amazon Echo mit Alexa [37]

zutage treten sogenannte *Conversational Interfaces* (CI) vermehrt auf. Dies wurde von den sprachgesteuerten Assistenten der *Big Four* (Apples Siri, Microsofts Cortana, Amazons Alexa und Googles Assistent) vorangetrieben [20]. In Abbildung 2.6 ist Amazons sprachgesteuerter Assistent Alexa zu sehen. Aber auch nicht-sprachgesteuerte CI (Chatbots) treten vermehrt auf. Diese sind durch den Einsatz des maschinellen Lernens und von neuronalen Netzen mächtiger als *Eliza*, was der großen Verbreitung von Smartphones und sozialen Netzwerken zu verdanken ist [20]. Diese bieten eine perfekte Umgebung für Chatbots. Diese sind längst nicht nur Computerprogramme, die zur Unterhaltung dienen. Sie helfen Nutzern bestimmte Aufgaben im Alltag zu erledigen. So kann jemand, der eine Pizza online bestellen möchte, eine Bestellung direkt beim Chatbot aufgeben [21]. Eine App des Lieferanten muss nicht mehr installieren werden. Den gleichen Ansatz könnte man nutzen, um BS zu melden, ohne eine spezielle App herunterladen zu müssen. Die Recherche beschäftigt sich mit Chatbots, die im Bereich des Gesundheitswesens eingesetzt werden.

Cameron et al. [22] stellen in ihrer Veröffentlichung vor, wie Chatbots bei psychischen Krankheiten wie Angstzuständen, Stress oder Depressionen helfen können. Der Chatbot wurde entwickelt, um dem Nutzer eine interaktive Umgebung zu bieten, bestimmte Informationen zu seiner Krankheit zu finden. Anstatt, dass der Nutzer eine Online-Bibliothek nach Dokumenten zu Erkrankungen durchsuchen muss, wird dieser vom Chatbot geleitet. Eine weitere Funktion des Chatbots ist die Konversation mit dem Nutzer über sein Wohlbefinden.

Eine weitere Einsatzmöglichkeit von Chatbots im Gesundheitswesen wird von Lokman et al. [23] vorgestellt. Es handelt sich dabei um einen Vorschlag, wie Chatbots Diabetes-Patienten helfen können. Hierfür wird online, durch einen Chatbot, eine erste Diagnose erstellt, wodurch ein Arztbesuch vermieden werden kann. Es entsteht ein Dialog, in dem der Chatbot konkrete Fragen stellt, welche vom Patienten beantwortet werden. Der Dialog wird beendet, sobald eine Diagnose gestellt werden konnte. Um eine Diagnose zu stellen, muss der Chatbot sich alle relevanten Eingaben des Nutzers merken.

Der von Madhu et al. [24] vorgestellte Chatbot soll dem Nutzer dabei helfen, Medikamente zu finden, die seinen Symptomen entsprechen. Basierend auf Informationen (Symptome, Alter, Größe, Gewicht) gibt der Chatbot eine Medikamentenempfehlung mit Dosierungshinweisen ab. Die Autoren stellen mehrere Anforderungen an ihren Chatbot: Es soll ein interaktives Echtzeitsystem sein, es soll Plattform übergreifend arbeiten und es soll einfach zu integrieren und erweiterbar sein.

Ein ähnliches System stellen Divya et al. [25] vor. Es handelt sich um einen Chatbot zur Diagnose von Erkrankungen anhand der vom Nutzer eingegebenen Symptome. Der Autor verspricht sich von diesem Ansatz eine günstigere und zeiteffizientere Variante, als ein Arztbesuch verursachen würde. Der Dialog mit dem Patienten verläuft linear und ist immer gleich strukturiert. Nach dem Login wird der Nutzer begrüßt und nach seinen Symptomen gefragt. Wurde eine Diagnose gestellt, wird entschieden ob es sich um eine schwere oder leichte Erkrankung handelt. Handelt es sich um keine schwere Erkrankung, so verabschiedet sich der Chatbot. Ist es jedoch eine schwere Erkrankung, so wird ihm ein Arzt empfohlen. Die wichtigsten Punkte der Unterhaltung werden in einer Datenbank gespeichert. Dieser kann der Arzt relevante Details entnehmen.

Das Start-Up *Gyant.com, Inc* aus Sillicon Valley stellt einen Chatbot zur Verfügung, der in den Facebook-Messenger integriert ist. Dieser dient zur Einschätzung von Symptomen einer Infektion mit dem Zika-Virus. Durch eine Reihe von Ja-/Nein-Fragen, kann sich der Nutzer vom Chatbot auf das Zika-Virus diagnostizieren lassen. Die meisten Fragen können mit Hilfe von Schnellantworten erledigt werden. Der häufige Einsatz von Emojis verleiht dem Chatbot einen menschlichen Charakter. Es wird auch verschiedenes Informationsmaterial

angeboten, um sich über das Zika-Virus zu informieren. Eine Beispielskonversation mit dem Chatbot ist in Abbildung 2.7 zu sehen.

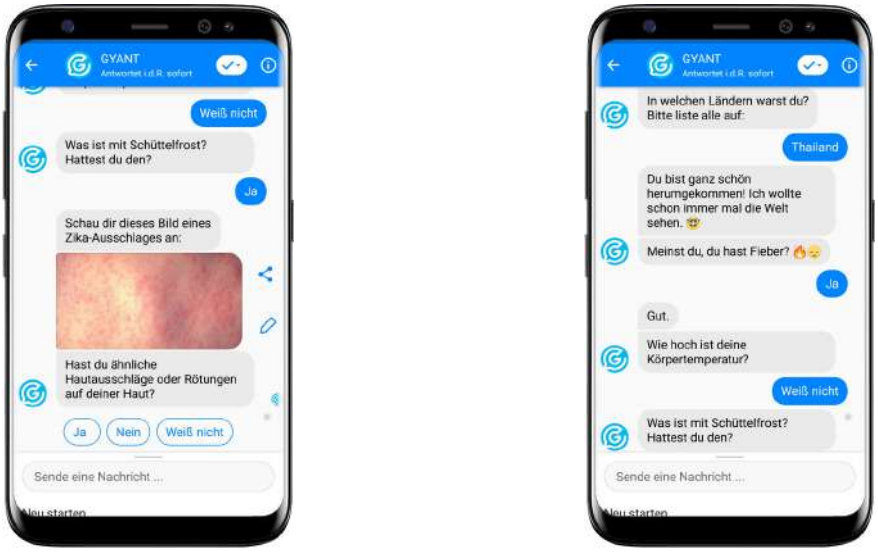

Abbildung 2.7 Beispiel einer Konversation mit dem GYANT-Chatbot über das Zika-Virus [38]

Die vorgestellten Veröffentlichungen geben einen Überblick über die Einsatzmöglichkeiten von Chatbots im Gesundheitswesen. In den meisten Publikationen geht es darum, eine erste Diagnose zu stellen oder Material zur Informationsbeschaffung bereitzustellen. In diesen Fällen ist die Intention des Chatbot dem Nutzer zu helfen. Im Fall dieser Arbeit ist die Intention des Chatbots eine Bilderdatenbank zu aktualisieren, wozu die Hilfe des Nutzers benötigt wird. Es wurden auch Anforderungen an die Chatbots untersucht. In allen Veröffentlichungen wurden als relevante Faktoren die Kosten- und Zeiteffizienz genannt. Es wurden ebenso technische Anforderungen gestellt. Es soll ein interaktives Echtzeitsystem, plattformübergreifend, einfach integrierbar und erweiterbar sein [24]. Diese Anforderungen werden für die Auswahl der Werkzeuge und der Umgebung für die Entwicklung des Chatbots berücksichtigt. Nützlich war es auch, einen solchen Chatbot auszuprobieren und zu sehen, wie eine Konversation aussehen kann. Beim Ausprobieren des Gyant Chatbots (gyant.com) ist der häufige Einsatz von Emojis und Memes aufgefallen. Dieses Verhalten verleiht dem Chatbot einen menschlichen Charakter. Des Weiteren ist während der Recherche aufgefallen, dass Chatbots keine Aufgaben für den Nutzer erledigen, sondern als Assistenten agieren. So wie in der Veröffentlichung von Cameron et al. [22] hilft der Chatbot dem Nutzer dabei

Informationen über psychische Erkrankungen zu finden. Ebenso assistieren die Chatbots in den Veröffentlichungen von Lokman et al. [23] und Madhu et al. [24] bei der Erstellung einer Diagnose bzw. bei der Dosierung von Medikamenten. Diese Aufgaben könnten auch ohne einen Chatbot gelöst werden, jedoch mit einem größeren Aufwand.

2.3 Crowdsourcing

Beim Begriff *Crowdsourcing* handelt es sich um das Auslagern (*Outsourcing*) von Aufgaben auf die „Masse“ (*Crowd*). Dabei nehmen freiwillige, untrainierte Personen an Aufgaben teil, die von einer kleinen Gruppe von Menschen oder Computern nicht zu bewältigen sind [26]. Dieses Prinzip macht sich auch das Projekt *Mobile4D* zu Nutze. Es handelt sich hierbei um ein Reporting-Alerting-System, welches in einer Zusammenarbeit des Capacity Lab an der Universität Bremen und des Ministry of Agriculture and Forestry (MAF) in Laos entstanden ist [27]. *Mobile4D* verfügt über eine mobile App, die von der Bevölkerung in Laos genutzt wird. Mit Hilfe dieser Anwendung können Umweltkatastrophen wie Überschwemmungen, Brände, Heuschreckenplagen, Dürre, Pflanzenbefall sowie Krankheiten beim MAF gemeldet werden. Reports von *Mobile4D* basieren auf dem Prinzip des *Micro-Reporting*. Gemeldete Fälle werden dann auf einer Karte dargestellt. Durch die Kartendarstellung kann schnell erkannt werden, an welchen Orten Hilfe benötigt wird. *Mobile4D* ist für diese Arbeit von großer Bedeutung, da eine Integration des *Dengue-Detectors* in das System vorgenommen wird.

Die Plattform *Ushahidi* (www.ushahidi.com) [28] diente anfangs dem Melden von Unruhen, die nach der Präsidentenwahl im Jahr 2007, in Kenia stattgefunden haben. Die Bevölkerung konnte Unruhen mit Hilfe von E-Mail oder SMS melden. Auch Radiostationen bezogen ihre Informationen von *Ushahidi*. Betreiber der Plattform hatten jedoch Bedenken, wie hoch der Wahrheitsgehalt einer Meldung ist. Es hat sich jedoch herausgestellt, dass durch *Crowdsourcing* Falschmeldungen leicht aufgedeckt werden können. Die Popularität des Systems ist so gestiegen, dass der Einsatz auch außerhalb der Grenzen von Kenia stattfindet.

Ein weiteres Beispiel für *Crowdsourcing* ist die Gesundheitsplattform *Flu Near You* (www.flunearyou.org). Diese beschäftigt sich mit dem Melden von Grippefällen. Entscheidet sich eine Person dafür, an dem *Crowdsourcing*-Projekt teilzunehmen, muss sich diese zuerst registrieren. Wöchentlich werden die Nutzer erinnert, an der Umfrage teilzunehmen. Sie haben die Möglichkeit vorgegebene Symptome auszuwählen. Zusätzlich wird gefragt, ob der Nutzer sich gegen die Grippe geimpft hat, um die Effektivität der Impfung zu untersuchen. Ein ähnliches System wurde auch in Europa eingeführt - *Influenzanet* (www.influenzanet.eu).

Die *Crowdsourcing*-Plattform *InnoCentive* (www.innocentive.com) [29] gibt Unternehmen die Möglichkeit Problemstellungen zu veröffentlichen, welche nicht innerhalb der Firma oder nur sehr kostspielig gelöst werden könnten. *InnoCentive* ist für jeden frei zugänglich. Nach einer Registrierung kann der Nutzer Probleme von Unternehmen wie *Boeing* oder *Procter & Gamble* lösen. Der Anreiz, ein Problem zu lösen, ist eine Belohnung in Form von 10.000$ bis 100.000$.

Auch *Amazon* macht sich das *Crowdsourcing* zu Nutze. Bei dem *Amazon Mechanical Turk* (www.mturk.com) [29] handelt es sich um einen webbasierten Marktplatz, der den Unternehmen dabei hilft, Menschen zu finden, welche sich mit bestimmten Aufgaben beschäftigen. Meistens sind es Aufgaben, die mit Hilfe von Computern schwer zu lösen sind, wie zum Beispiel das Erkennen von Gegenständen auf Bildern, das Schreiben von Produktbeschreibungen oder das Übersetzen von Texten. Belohnungen für das Erledigen einer solchen Aufgabe liegen im Cent-Bereich.

Die oben betrachteten Veröffentlichungen und Plattformen zum Thema *Crowdsourcing* geben einen groben Überblick. *Crowdcourcing* ist in vielen Bereichen einsetzbar, angefangen bei humanitären Zwecken (*Mobile4D* und *Ushahidi*), über Gesundheitswesen (*Flu Near You*, *Influenzanet*), bis hin zur Lösung von technischen Problemen (*InnoCentive*, *Amazon Mechanical Turk*). Jeder dieser Bereiche stößt auf Probleme: So ist es bei Non-Profit-Projekten schwierig, Menschen dazu zu bewegen, an solchen Projekten teilzunehmen. Viele mögliche Nutzer werden schon alleine durch die Registrierung abgeschreckt. Entscheidet sich ein Nutzer dafür teilzunehmen, entsteht das Problem von falschen Informationen. So beschrieb Ory Okoloh in seiner Arbeit einen Fall von einer Falschmeldung über Unruhen in Kenia [28].

Diese konnte jedoch durch andere Nutzer als falsch dargestellt werden, was ohne eine große Anzahl an Nutzern nicht möglich gewesen wäre. Plattformen wie *InnoCentive* oder *Amazon Mechanical Turk*, haben keine Probleme, aktive Nutzer für sich zu gewinnen, was damit begründet werden kann, dass diese geldlich belohnt werden. Das Problem, welches dabei auftritt, ist dass Nutzer versuchen, das System zu missbrauchen, indem sie ohne Beachtung der Aufgaben die Belohnung erhalten.

Für diese Arbeit können monetäre Anreize nicht in Betracht gezogen werden, da es sich um ein humanitäres Non-Profit-Projekt handelt. Wichtig ist dennoch, dass ein Mechanismus gefunden wird, welcher überprüft, ob die Aufgabe auch wie vorgesehen erledigt wurde und ob die gewonnenen Informationen auch wirklich stimmen.

2.4 Abgrenzung der Arbeit zu verwandten Arbeiten

Im Abschnitt 2.1 wurden mehrere Systeme zur Überwachung und Kontrolle des Dengue-Fiebers vorgestellt. Die dort aufgeführten Systeme lassen sich in zwei Gruppen unterteilen: Einerseits sind es mobile Anwendungen, die es der Bevölkerung ermöglichen, potentielle BS zu melden. Diese haben den Vorteil, dass die Kartierung von BS digital vorgenommen werden kann. Das Problem welches dabei weiterhin besteht, ist die manuelle Auswertung und Verarbeitung der erstellten Bilder. Diese müssen immer noch von Menschenhand untersucht werden. Eine schlechte Skalierbarkeit ist somit weiterhin vorhanden. Des Weiteren wurden ebenfalls mehrere Systeme vorgestellt, die sich mit der automatisierten Erkennung von BS beschäftigen. Dazu werden Bilderdatenbanken mit Hilfe eines Objekterkennungs-Algorithmus auf BS untersucht. Die Bilderdatenbanken beziehen ihre Daten aus Quellen wie GSV, Flickr oder es handelt sich um Drohnennaufnahmen. Die Idee dieser Arbeit, ist es diese zwei Gruppen von Systemen miteinander in Verbindung zu bringen. Dazu werden drei mobile Applikationen entwickelt, um Bilder von potentiellen BS zu erstellen. Diese kommunizieren mit einem Server, auf welchem erstellte Bilder mit Hilfe eines Objekterkennungs-Algorithmus auf BS untersucht. Dadurch muss die Auswertung nicht mehr manuell vorgenommen werden. Hierdurch ist die Untersuchung von größeren Gebieten möglich. Orte an denen Bilder erstellt

werden sollen, werden ebenfalls von einer Datenbank zur Verfügung gestellt. Bei den drei mobilen Applikationen handelt es sich unter anderem um einen Facebook Chatbot. Im Abschnitt 2.2 wurden mehrere Chatbots aus dem Bereich Gesundheitswesen vorgestellt. Diese agieren meistens als Assistenten um Diagnosen zu stellen oder Informationsmaterial bereitzustellen. Beim Dengue-Chatbot sollen Nutzer jedoch, mit dem Ansatz des *Crowdsourcing*, eine Bilderdatenbank ergänzen, welche auf BS untersucht wird. Während der Recherche konnte kein Chatbot gefunden werden, der als *Crowdsourcing*-Werkzeug dient.

Kapitel 3

Konzept

Die Übersicht des Systems ist in Abbildung 3.1 zu sehen. Im Zentrum steht die Bild-Datenbank, welche mit GSV-Bildern befüllt ist. Von dieser Datenbank lädt sich das Modul *Breeding Site Detction* die Bilder herunter und untersucht sie auf BS. Dazu wird im ersten Schritt ein Objekterkennungs-Algorithmus angewandt. Die erkannten Objekte werden dann klassifiziert. Die auf diese Weise erkannten BS werden zurück in die Datenbank geschrieben. Soll ein Gebiet untersucht werden, so wird ein Raster mit 100m Abstand erstellt und Geo-Koordinaten ermittelt. An jedem Punkt werden nun mit Hilfe der GSV-API die entsprechenden Aufnahmen heruntergeladen. Da das Herunterladen von Panoramabilder von der GSV-API nicht unterstützt wird, werden 5 Einzelbilder heruntergeladen. Die Bilder enthalten Metadaten wie Erstellungsdatum und -Ort. Diese Informationen werden genutzt, um die gefundenen BS in einer Heat-Map zu visualisieren (siehe Einleitung Abbildung 1.3). Es kann allerdings sein, dass GSV über keine oder nur veraltete Bilder von manchen Orten verfügt. Orte, an denen es der Fall ist, werden als *Spot* bezeichnet. Dieses Problem soll mit Hilfe von mobilen Anwendungen und *Crowdsourcing* gelöst werden. Das *Crowdsourcing* kann durch die *Dengue-Detector-App* (3.1), die mobile App des Reporting-Alerting-Systems *Mobile4D* (3.2) oder den Facebook-Chatbot (3.3) erfolgen.

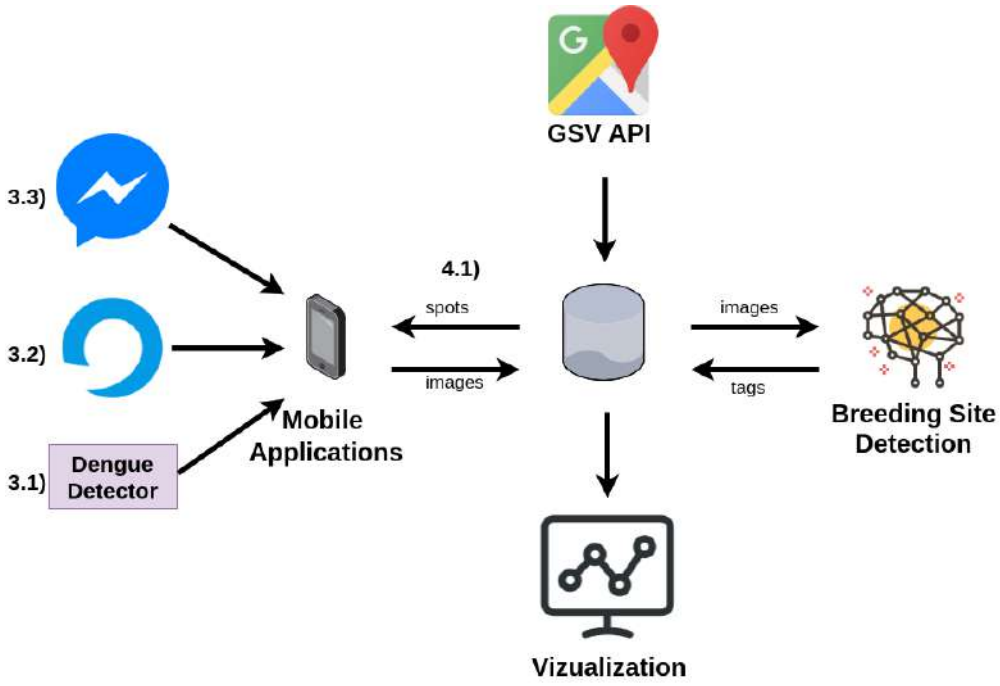

Abbildung 3.1 Übersicht des Gesamtsystems

Alle drei Anwendungen arbeiten nach dem gleichem Prinzip:

1. *Spot* auswählen
2. Zum *Spot* hinbegeben
3. Bilder am *Spot* erstellen
4. Bilder hochladen

Um zu erfahren wo Bilder erstellt werden müssen, fordern die mobilen Anwendungen *Spots*, über eine REST-Schnittstelle (4.1), von der Datenbank an. Bilder, die mit Hilfe der mobilen Anwendungen gesammelt wurden, werden in die Bilderdatenbank geschrieben und wie GSV-Bilder auf BS untersucht. Das Modul *Visualization* greift auf die Bilderdatenbank zu und zeichnet die erkannten BS auf einer Karte ein. Des Weiteren ist es in der Karte möglich, *Spots* anzuzeigen.

3.1 Dengue Detector App

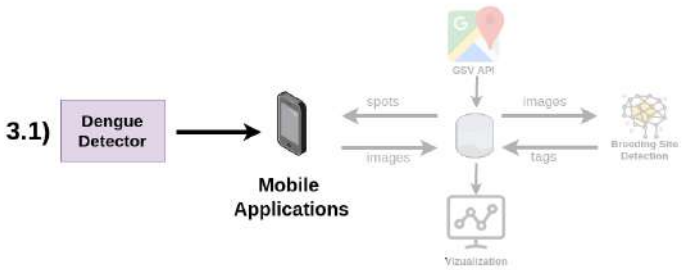

Abbildung 3.2 Dengue-Detector-App

In diesem Abschnitt wird auf die Funktionalität der *Dengue-Detector*-App eingegangen. Dabei wird die Benutzeroberfläche erläutert. Der Prototyp der App ist im Rahmen eines Praktikums an der Mahidol University in Thailand entstanden. In dieser Arbeit erfolgt die Integration der App mit dem oben beschriebenen Server.

3.1.1 List of Spots - Startseite

Wird die *Dengue-Detector-App* gestartet, so gelangt man zur Liste von *Spots* (Siehe Abbildung 3.3 links). Sortiert sind diese nach der Entfernung zu der aktuellen Position des Nutzers. Die Entfernung zu einem *Spot* ist in jedem Listeneintrag zu sehen. Sind Informationen wie Stadt, Straßenname, District und Subdisctrict vorhanden, so werden diese angezeigt. Ist dies nicht der Fall, so wird ein einheitlicher Text „Missing photo location“ angezeigt. Die Liste der *Spots* wird durch eine REST-Anfrage angefordert. Diese beinhaltet den aktuellen Standort und einen Radius, um die *Spots* einzugrenzen. Als Antwort erhält man *Spots* mit Geo-Koordinaten, von denen Bilder erstellt werden sollen. Mit einem Schieberegler lassen sich *Spots* filtern. Klickt man auf einen Eintrag, so gelangt man zu einer Detailansicht (siehe Abbildung 3.3 rechts). Dort ist der *Spot* und die aktuelle Position des Nutzers zu sehen. Somit weiß dieser, wohin er sich begeben soll, um den *Spot* zu fotografieren. Alternativ zu der Listenansicht kann in der unteren Tab-Leiste zur Kartenansicht gewechselt werden (siehe mittlere Abbildung 3.3). Um in die Kameraansicht zu wechseln, muss das Kamera-Icon angeklickt werden.

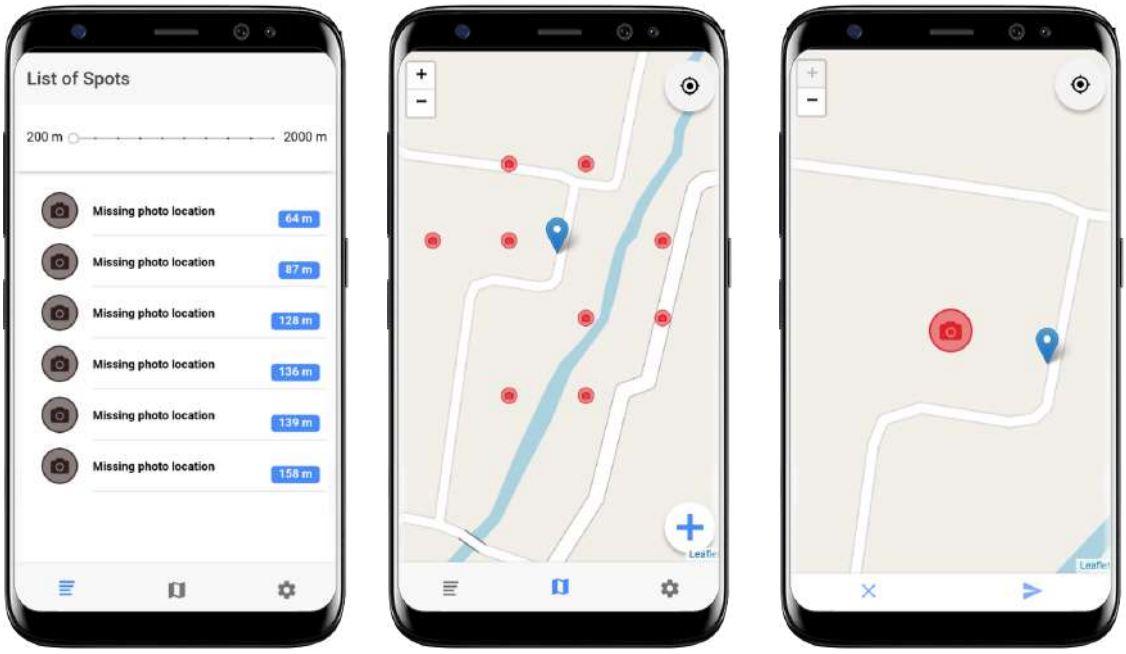

Abbildung 3.3 Ansichten der Dengue-Detector-App: (links) Listenansicht, (Mitte) Kartenansicht, (rechts) Detailansicht

3.1.2 Spots-Kartenansicht

In der Kartenansicht (siehe mittlere Abbildung 3.3) sind alle *Spots* mit jeweils einem roten Kamera-Icon eingezeichnet. Durch die Kartenansicht fällt es leichter, eine Route zu planen, auf der man *Spots* abarbeiten möchte. Zusätzlich können Nutzer sehen, wie gut die Bildabdeckung an ihrer aktuellen Position ist. Je mehr Kamera-Icons zu sehen sind, desto schlechter ist die Bildabdeckung. Um einen *Spot* abzuarbeiten, muss der Nutzer zu diesem hingehen. Beim *Spot* angekommen muss das rote Kamera-Icon angeklickt werden, wodurch sich die Kameraansicht öffnet.

3.1.3 Spot-Detailansicht

Die Detailansicht (siehe Abbildung 3.3 rechts) erscheint nach dem Klicken auf ein *Spot*, entweder in der Listen- oder Kartenansicht. Es öffnet sich eine Karte, auf der nur der ausgewählte *Spot* und die aktuelle Position eingezeichnet sind. Klickt man auf das rote Kamera-Icon, öffnet sich die Kameraansicht. Zusätzlich zeigt die Farbe der Markierung den Zustand des

Spots an:

 Spot nicht abgearbeitet, keine Fotos wurden erstellt.

 Spot nicht vollständig abgearbeitet, nicht alle Fotos wurden erstellt.

 Spot abgearbeitet, alle Fotos wurden erstellt.

Wurden Bilder erstellt, können diese an den Server abgeschickt werden. Der *Spot* ist somit abgearbeitet.

3.1.4 Kameraansicht

In der Kameraansicht soll eine Rundumsicht erstellt werden. Es wird dabei nicht ein einzelnes Panoramabild, sondern mehrere Einzelbilder erstellt. Als Hilfestellung werden mehrere Markierungen auf einer horizontalen Geraden angeordnet (siehe Abbildung 3.4). Diese passen sich der Bewegung des Nutzers an. Bei einer Drehbewegung nach links bewegen sich die Markierungen nach rechts und umgekehrt. Dadurch erscheint es dem Benutzer als wären die Markierungen bezüglich der Umgebung fix. Um ein Bild zu erstellen, muss die Markierung „anvisiert“ werden (siehe mittlere Abbildung 3.4). Wurde eine Markierung anvisiert, füllt sich ein runder Fortschrittsbalken. Ist dieser aufgefüllt, wird ein Foto automatisch erstellt und die Markierung färbt sich grün (siehe Abbildung 3.4 rechts). Im oberen Rand sind Vorschaubilder aller erstellten Fotos zu sehen. Klickt man auf ein Foto, so gelangt man zu der Galerieansicht. Dort können die erstellten Bilder gelöscht werden. Die Markierung an der entsprechenden Stelle färbt sich dann wieder rot. Um ein Bild zu ersetzen, kann dies in der Galerie gelöscht und neu erstellt werden. Alternativ kann die entsprechende grüne Markierung wieder anvisiert und ein neues Bild wird erstellt.

Wurden alle Bilder erstellt (alle Markierungen sind grün gefärbt), wechselt die Sicht wieder in die Detailansicht. Ein Dialog erscheint, in dem der Nutzer gefragt wird, ob die Bilder an den Server abgeschickt werden sollen. Alternativ dazu kann der ✓ Button benutzt werden, um in die Detailansicht zu wechseln. Durch das Klicken auf ✕ werden alle erstellten Bilder

Abbildung 3.4 Kameraansicht der Dengue-Detector-App

verworfen und man wechselt ebenfalls in die Detailansicht. Der Assistent zum Erstellen der Bilder lässt sich in den Einstellungen deaktivieren. Dies ist notwendig, wenn Sensoren wie Gyroskop und Accelormeter nicht zur Verfügung stehen oder eine mangelnde Qualität haben. Diese sind jedoch erforderlich, um die Rotation des Nutzers herauszufinden.

3.2 Mobile4D

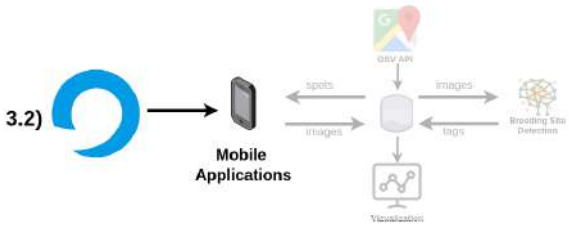

Abbildung 3.5 Mobile4D-App

Eine weitere Aufgabe der Arbeit ist die Integration des *Dengue-Detector*-Moduls in das Reporting-Alerting-System *Mobile4D*. Das gleichnamige Projekt findet an der Universität

Bremen statt und kooperiert mit dem laotischen Ministerium für Agrikultur und Forstwirtschaft. Ziel von *Mobile4D* ist es, der laotischen Bevölkerung eine effiziente Möglichkeit zu geben, Umweltkatastrophen zu melden. Zu Umweltkatastrophen zählen Brände, Dürren, Überschwemmungen, Infrastrukturelle Probleme, Menschen- und Tierkrankheiten sowie Heuschrecken-Plagen und Pflanzenbefall. Diese Katastrophen können mit Hilfe einer mobilen App beim Ministerium gemeldet werden. Die gemeldeten Katastrophen lassen sich auf einer administrativen Webseite abrufen. Somit können Mitarbeiter des Ministeriums Entscheidungen fällen, ob und welche Gegenmaßnahmen, im Falle einer Umweltkatastrophe, eingeleitet werden. Im Rahmen dieser Arbeit erfolgte eine Integration des *Dengue-Detector*-Moduls, welches die Funktionalität der *Dengue-Detector*-App beinhaltet. Da die Sichten der *Mobile4D*-App den der *Dengue-Detector*-App stark ähneln, erfolgt die Beschreibung weniger detailliert.

Nach dem Start der *Mobile4D*-App wird dem Nutzer eine Karte mit Reports in seiner Nähe angezeigt. Um zum Dengue-Modul zu gelangen, muss der Nutzer auf den Tab „Create a Report“ klicken. Es erscheint eine Übersicht der verschiedenen Arten von Reports (siehe Abbildung 3.6 links). In der unteren rechten Ecke befindet sich der Button für das Dengue-Modul (*Breeding Site*). Durch einen Klick auf den Button gelangt man zur Liste der *Spots* (vergleiche Abbildung 3.3). Diese werden, wie in der *Dengue-Detector*-App, vom Server angefordert und in eine Liste geschrieben. Klickt man auf einen Listeneintrag, so gelangt man in die Detailansicht (vergleiche Abbildung 3.3). Dort ist der *Spot* und die Position des Nutzers zu sehen. Klickt man auf die rote Markierung, gelangt man in die Galerieansicht. Diese ist anfangs leer und beinhaltet einen Text, der dem Nutzer mitteilt, Bilder im Landscape-Modus zu erstellen (siehe mittlere Abbildung 3.6). Zum Erstellen von Bildern muss auf den Button „Take Pictures“ geklickt werden und man gelangt in die Kameraansicht. Die *Mobile4D*-App verfügt über keinen Assistenten, um Bilder automatisch zu erstellen. Die erstellten Bilder sind in der Galerieansicht zu sehen und können dort gelöscht werden (siehe Abbildung 3.6 rechts). Um einen abgearbeiteten *Spot* an den Server abzuschicken, muss der Nutzer auf ✓ klicken. Ein abgearbeiteter *Spot* wird dann auf dem Server auf BS untersucht.

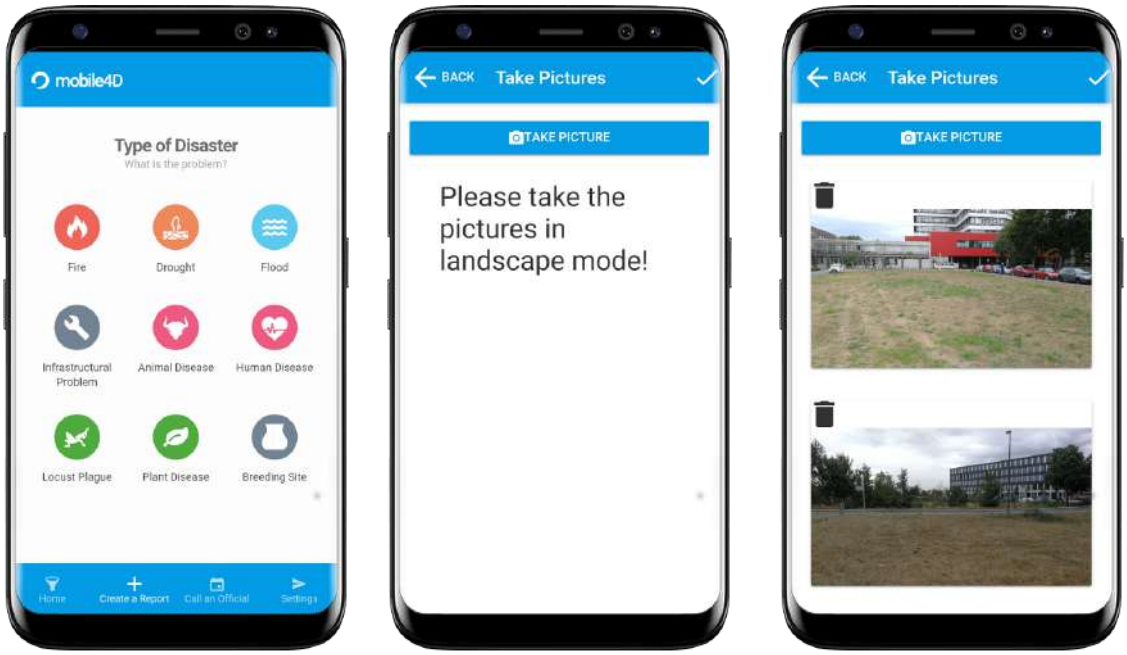

Abbildung 3.6 Ansichten der Mobile4D-App: (links) Create a Report, (mitte) Leere Galerieansicht, (rechts) Galerieansicht mit Bildern

3.3 Chatbot

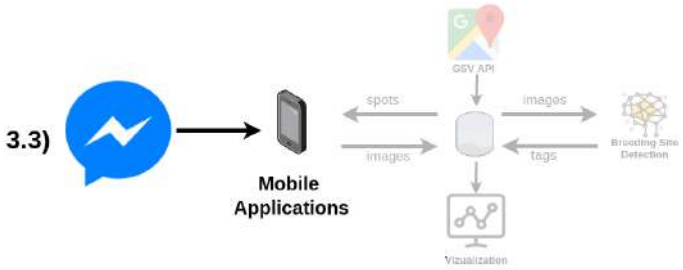

Abbildung 3.7 Facebook Chatbot

Als letzter Teil der Arbeit erfolgt der Entwurf und die Implementierung eines Facebook-Chatbots. Dieser hat dieselbe Aufgabe, wie die *Dengue-Detector*- und *Mobile4D*-App. Durch die Verwendung eines Chatbots entfällt das Herunterladen und Installieren einer Anwendung. Der *Facebook Messenger* hat in Thailand einen Marktanteil von über 60 Prozent und ist, zusammen mit dem *LINE-Messenger*, marktführend [33]. Dadurch hat der Chatbot viele potenzielle Nutzer. Des Weiteren bietet Facebook eine Chatbot-freundliche API, mit der man

Natural-Language-Processing-Frameworks (NLP) nutzen kann. NLP-Frameworks werden dazu verwendet, um natürlichsprachige Eingaben zu behandeln. Zu den bekanntesten zählen Googles *Dialogflow*, Facebooks *wit.ai*, IBMs *Watson NLU* und Microsofts *Luis.ai*. Um das passende Framework auszuwählen, wurde eine Internetrecherche durchgeführt. Diese lieferte ähnliche Ergebnisse für alle vier Frameworks. Sie sind alle kostenlos und arbeiten nach einem ähnlichen Prinzip (*Intents*, *Entities* und *Actions*) [39]. Die Entscheidung fiel auf Dialogflow, da es eine niedrige Komplexität, eine einfache Bedienung sowie einen hohen Community-Support hat. Des Weiteren ist das Einrichten eines *Webhooks* unproblematisch, was bei wit.ai nicht der Fall ist [40]. Auch die mögliche spätere Anbindung an andere Messaging-Plattformen war für die Entscheidung ausschlaggebend. Ein Chatbot, der mit Dialogflow entwickelt wurde, lässt sich neben Facebook auch an Plattformen wie der *LINE-Messenger*, *Slack* oder *Telegram* anbinden. Da Microsofts Louis sich in der Beta-Phase befindet und IBMs Watson NLU für das Vorhaben der Arbeit zu komplex ist, schieden diese beiden Frameworks aus.

Um einen Facebook-Chatbot zu veröffentlichen, müssen folgende Schritte erledigt werden:

- Facebook-Nutzer und Seite erstellen
- Modellierung des Dialogs mit Dialogflow
- Integration von Dialogflow und Facebook

Im Folgenden wird die Funktionsweise von Dialogflow beschrieben.

3.3.1 Natural-Language-Processing-Framework: Dialogflow

In dieser Arbeit wird das NLP-Framework Dialogflow genutzt. Kernkomponenten des Frameworks sind *Intents*. Ein *Intent* nimmt Nutzereingaben entgegen, verarbeitet diese, wählt eine Antwort aus und verschickt diese zurück an den Nutzer. Dazu müssen *Intents* zuerst mit möglichen Nutzereingaben (*Training Phrases*) trainiert werden. Es handelt sich hierbei

um Begriffe, bei denen das *Intent* ausgelöst werden soll. So wird zum Beispiel das `Default Welcome Intent` ausgelöst, wenn der Nutzer den Chatbot begrüßt hat. In der Abbildung 3.8 sind *Training Phrases* zu dem `Default Welcome Intent` zu sehen. Das `Default Welcome`

Abbildung 3.8 Trainingsphrasen zum Welcome Intent

`Intent` lässt sich mit Begriffen, wie: „Hi“, „Good morning“ oder „Hello“ auslösen (siehe Abbildung 3.8). Mit Hilfe von Machine-Learning, werden auch Nutzereingaben erkannt, die nicht explizit in den *Training Phrases* aufgeführt sind. Der Chatbot reagiert ebenfalls auf Eingaben wie „Hey“ oder „Good evening“. Genauso wie *Training Phrases*, lassen sich auch Antworten (*Responses*) festlegen. Antworten zum `Default Welcome Intent` sind in Abbildung 3.9 zu sehen. Diese erhält der Nutzer als Antwort auf eine Begrüßung. Im Fall dieser Arbeit reichen Antworten, die in einem *Intent* festgelegt werden können, nicht aus. Der Chatbot soll die Nutzer mit *Spots* versorgen und Bilder entgegennehmen. Dazu muss eine Verbindung zwischen Dialogflow und der Datenbank hergestellt werden. Um Dialogflow mit einem externen Service zu verbinden, muss ein *Webhook* eingerichtet werden. Dieser leitet Nutzereingaben an den externen Service weiter, welcher dann mit einer Liste von Spots antwortet. Eine detaillierte Beschreibung des *Webhooks* erfolgt im Abschnitt 4.3.3.

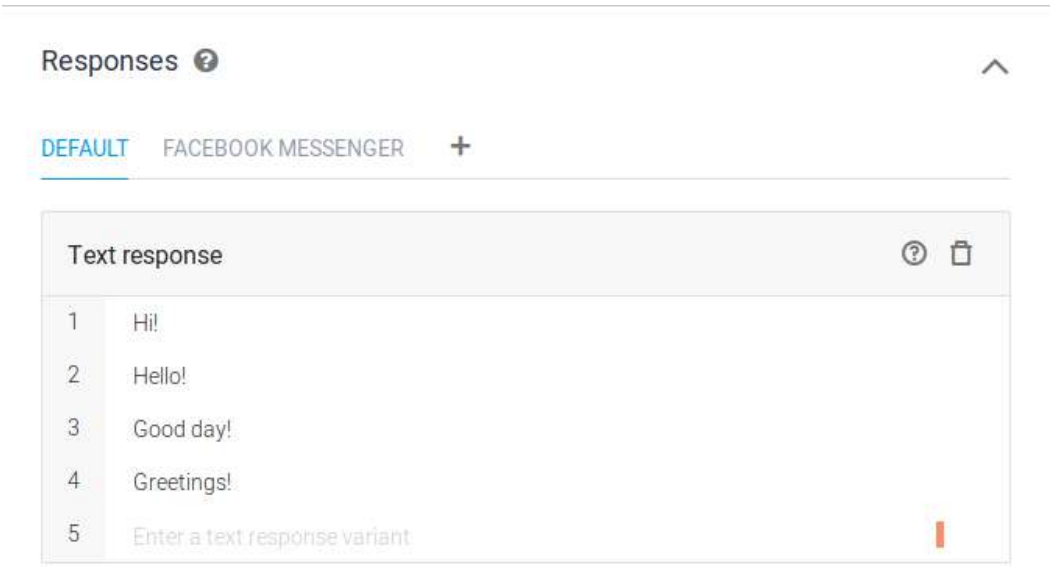

Abbildung 3.9 Antwortsphrasen zum Welcome Intent

3.3.2 Modellierung des Workflows

In diesem Abschnitt erfolgt die Modellierung des Ablaufs der Konversation zwischen dem Nutzer und dem Chatbot. Wie bei den Android-Apps, müssen folgende Schritte durchgeführt werden: *Spot* auswählen, zum *Spot* hingehen, Fotos erstellen und an den Server schicken. Die einzelnen Schritte sind in dem Flussdiagramm in Abbildung 3.10 zu sehen. Um eine Konversation zu beginnen, muss der Nutzer den Chatbot begrüßen (1.). Dieser antwortet darauf ebenfalls mit einer Begrüßung und fragt, ob der Nutzer ihm helfen möchte (2.). Die Antwort kann mithilfe von *quick-replies* erfolgen. Ein *quick-reply* ist eine vorgegebene Antwortmöglichkeit in Form eines Buttons. Dadurch erspart man dem Nutzer das Eintippen einer Antwort und senkt gleichzeitig das Risiko einer falschen Eingabe. Zur Auswahl stehen „*yes*“ und „*no*“. Falls die Antwort „*no*“ gewählt wird, verabschiedet sich der Chatbot (11.). Hat sich der Nutzer entschieden zu helfen, so wird er gebeten, seinen Standort mitzuteilen (3.). Dazu steht ebenfalls ein *quick-reply* zur Verfügung. Als Antwort auf den Standort (4.) erhält der Nutzer die Liste der *Spots* in seiner Nähe (5.). Er wählt hieraus einen aus und begibt sich dort hin (6.). Am *Spot* angekommen, erhält der Nutzer den Auftrag, Bilder zu erstellen und diese an den Chatbot zu senden (7.). Nachdem ein Bild an den Chatbot gesendet wurde, wird der Nutzer gefragt, ob sich eine BS im Bild befindet (8.). Falls ja, wird der Nutzer gebeten,

die genaue Art der BS zu klassifizieren (9.). Die Klassifizierung soll dem Zweck dienen, den Objekterkennungs-Algorithmus weiter zu schulen. Hat der Chatbot diese Information erhalten, wird der Nutzer gefragt, ob er weiter helfen möchte oder nicht (10.). Falls nicht, verabschiedet sich der Chatbot (11.). Andernfalls wird dem Nutzer wieder eine Liste von *Spots* zugeschickt (3.).

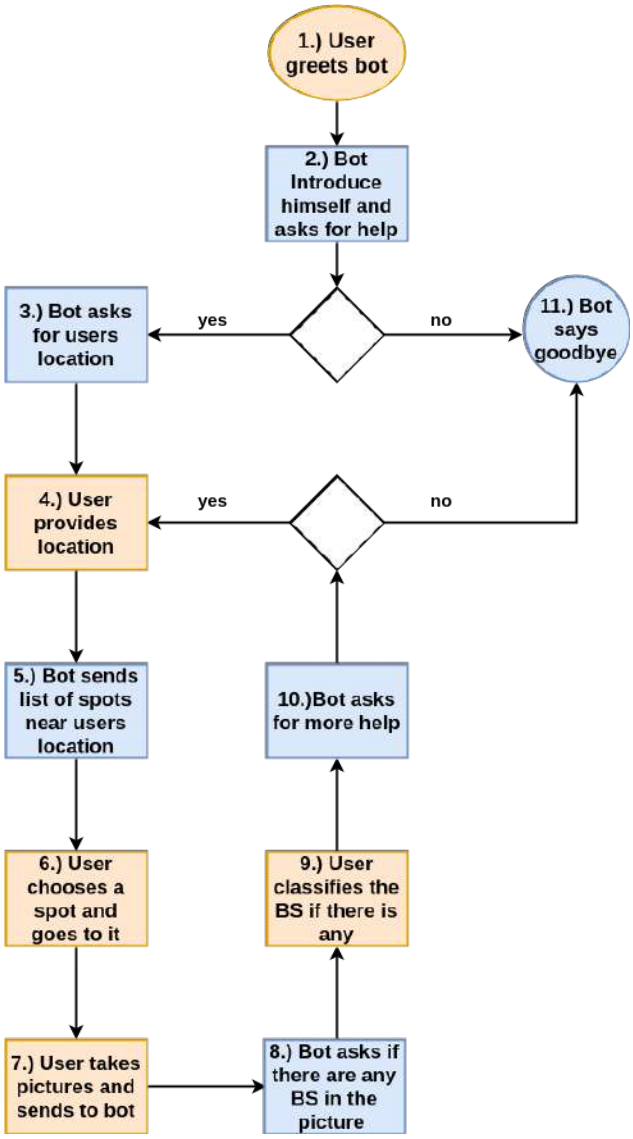

Abbildung 3.10 Ablauf einer Konversation mit dem Chatbot

3.3.3 Beispiel

In der Abbildung 3.11 ist links die Begrüßung des Chatbots zu sehen. Dieser begrüßt den Nutzer und fragt, ob er helfen möchte das Dengue-Fieber zu bekämpfen. Der Nutzer antwortet mit „*yes*“. Dadurch bekommt er mitgeteilt, dass er Bilder erstellen soll, die dann auf BS untersucht werden. Um zu erfahren, wo Bilder erstellt werden sollen, wird der Nutzer gebeten seinen Standort mitzuteilen. Dies ist mit Hilfe einer *quick-reply* möglich. Nachdem der Nutzer seinen Standort mitgeteilt hat, bekommt er eine Anleitung, wie ein *Spot* ausgewählt wird und wie Bilder erstellt werden sollen (siehe mittlere Abbildung 3.11). Zusätzlich wird die Liste der *Spots* in einem Karussell angezeigt (siehe Abbildung 3.11 rechts). Um ein *Spot* auszuwählen, muss ein Eintrag im Karussell geklickt werden. Durch die Verwendung der Google-API, kann die Navigationsfunktion von Google Maps genutzt werden. Somit ist es für den Nutzer leichter, einen *Spot* zu erreichen (siehe Abbildung 3.12 links). Nachdem ein *Spot* erreicht wurde, soll der Nutzer Bilder erstellen und an den Chatbot schicken. Dies ist entweder direkt im Messenger möglich oder es kann auch die native Kamera-App des Geräts genutzt werden. Ein mit der nativen Kamera-App erstelltes Foto muss zuerst in der

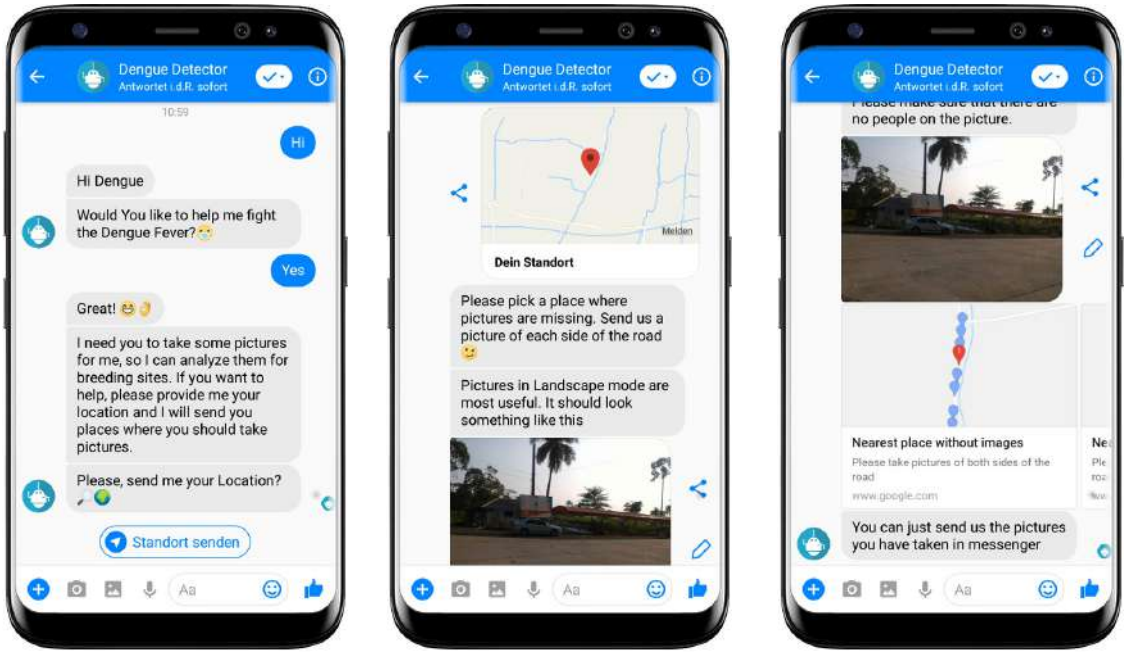

Abbildung 3.11 Beispiel der Konversation mit dem Chatbot Teil 1

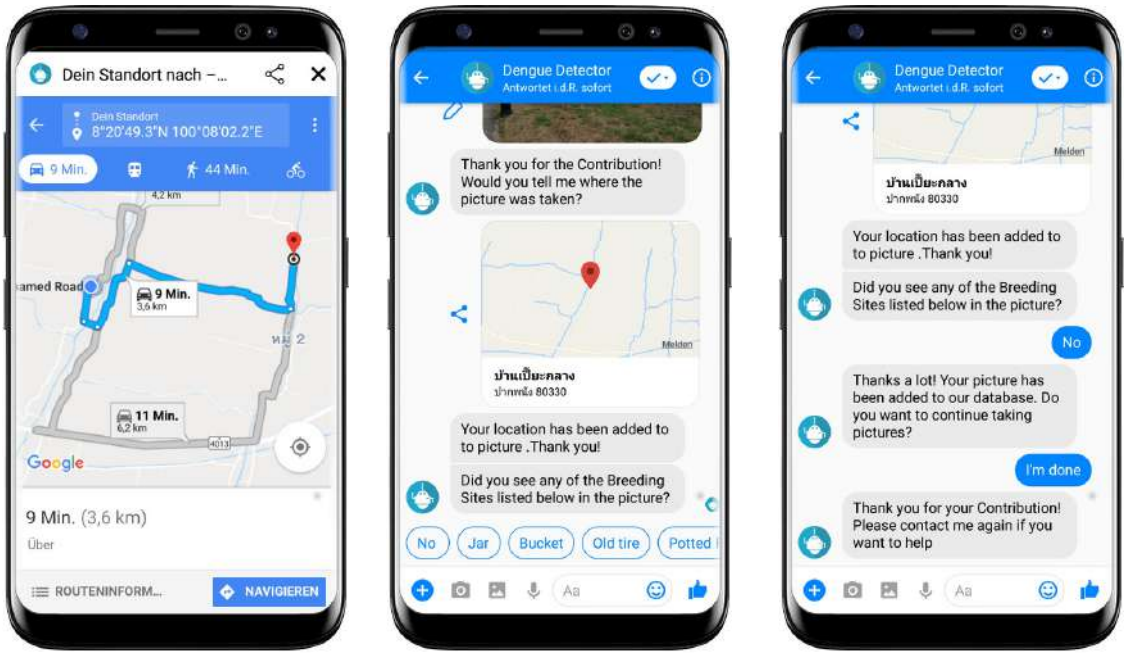

Abbildung 3.12 Beispiel der Konversation mit dem Chatbot Teil 2

Galerie ausgewählt und dann im Messenger verschickt werden. Nachdem der Chatbot ein Bild erhalten hat, wird der Nutzer noch einmal gebeten, seinen Standort mitzuteilen. Diese Information ist wichtig, um das Bild mit einem Geo-Tag zu versehen, was wieder über eine *quick-reply* erledigt werden kann. Anschließend wird gefragt, ob sich eine BS im Bild befindet. Die Frage kann durch *quick-replies* beantwortet werden. Falls kein BS auf dem Bild zu sehen ist, klickt der Nutzer auf „*no*“. Andernfalls wird der Nutzer gebeten, die BS zu klassifizieren. Dazu stehen mehrere Klassen zur Auswahl (siehe mittlere Abbildung 3.12). Nachdem diese Frage beantwortet wurde, kann der Nutzer entscheiden, ob er weiter helfen möchte. Falls ja, werden ihm wieder *Spots* zugeschickt. Ansonsten verabschiedet sich der Chatbot und die Konversation ist beendet (siehe Abbildung 3.12 rechts).

Kapitel 4

Implementierung

In diesem Kapitel werden die technischen Details der Implementierung und der Integration beschrieben. Dazu zählt die Implementierung der *Dengue-Detector*-App, des Dengue-Moduls für *Mobile4D*, des Chatbots sowie die Integration mit dem Server.

4.1 REST-Schnittstelle

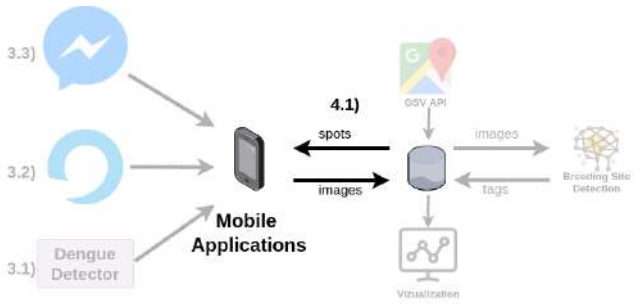

Abbildung 4.1 REST-Schnittstelle zwischen den mobilen Anwendungen und dem Server

Alle Anwendungen müssen mit dem Server kommunizieren. Dazu steht eine REST-Schnittstelle zur Verfügung. Es gibt zwei Operationen:

1. `SPOTS_IN_RANGE`: Mit Hilfe dieser Operationen werden die Spots vom Server angefordert. Dazu muss eine http-post-Anfrage mit dem `GeoJson` aus dem Codeausschnitt 4.1 an den Server geschickt werden. In dem `GeoJson`-Objekt werden Latitude und

Longitude sowie ein ganzzahliger Radius in Meter angegeben. Als Antwort erhält man ebenfalls eine `GeoJson`, welche die Koordinaten der fehlenden Spots enthält (siehe Codeausschnitt 4.3).

2. `SPOT_DONE`: Mit dieser Operation werden Bilder an den Server verschickt. Dazu werden diese im `Base_64` Format umgewandelt und in einer Liste gespeichert. Es handelt sich hierbei wieder um eine http-post-Anfrage. Das `GeoJson`-Objekt beinhaltet ebenfalls Geo-Koordinaten, welche als Geo-Tag für die Bilder dienen. War die Anfrage erfolgreich, so wird vom Server eine `Success`-Nachricht gesendet. Koordinaten die hier mit übergeben werden, dienen dazu die Bilder mit einem Geo-Tag zu versehen (siehe Codeausschnitt 4.2).

```
{
 type: "Feature",
  geometry: {
   type: "Point",
   coordinates: [lng, lat],
  },
 properties: {
  radius: r,
 },
};
```

Listing 4.1 `SPOTS_IN_RANGE`

```
{
 type: "Feature",
  geometry: {
   type: "Point",
   coordinates: [lng, lat],
  },
 properties: {
  base64_images: photos,
 },
};
```

Listing 4.2 `SPOTS_DONE`

Das nach der Ausführung der `SPOTS_IN_RANGE`-Operation empfangene `GeoJson`-Objekt ist in Codeausschnitt 4.3 zu sehen. Dieses beinhaltet eine List von `geometry`-Objekten, in welchen die Koordinaten der fehlenden Spots geschrieben sind.

```
{
 data: {
  type: "FeatureCollection"
  features: [{
    geometry: {
     coordinates: [lat, lng],
     type: "Point"
    }
    properties: {},
    type: "Feature"
   }, ... ]
 }
 message: "Missing-points found"
 status: "success"
}
```

Listing 4.3 GeoJson mit fehlenden Spots

Mit `status: "success"` und `message: "Missing-points found"`, lässt es sich überprüfen, ob die http-post-Anfrage erfolgreich war. Ist dies nicht der Fall, so erhält man eine Antwort mit `status: "error"` und der `message: "No missing-points found in the radius"`. Konnten die erstellten Bilder erfolgreich hochgeladen werden so erhält man als Antwort: `message: "The images have been uploaded."` und `status: "success"`.

4.2 Dengue-Detector-App und Mobile4d

Für die Implementierung der *Dengue-Detector*-App wurde das Framework *Ionic* Version 3.2 verwendet. Es handelt sich dabei um ein Open-Source-Framework zur Entwicklung von hybriden Apps mit Webtechnologien wie Typescript, HTML5 und Sass. *Ionic* basiert auf dem Front-End-Webapplikationsframework *Angular*. Um einen Zugriff auf Hardware-Komponenten der jeweiligen Plattformen zu bekommen, nutzt *Ionic* das *Cordova*-Framework. Dadurch kann auf Hardware-Komponenten wie Kamera, GPS, oder Accelerometer zugegriffen werden. Zusätzlich stellt *Ionic* ein Command-line Interface zur Verfügung, in welchem sich Projekte erstellen, Plugins hinzufügen sowie Projekte bauen lassen. In diesem Abschnitt wird nicht die Implementierung der einzelnen Sichten beschrieben, da diese als trivial angesehen werden können. Stattdessen wird expliziter auf die Implementierung des Assistenten der Kamera eingegangen. Um die Rotation der Markierungen in der Kameraansicht zu realisieren, wurde der *Accelerometer*-Sensor und der *Game-Rotation-Vector* verwendet. Diese werden benötigt, um die Ausrichtung des Smartphones zu berechnen. Gleiches Prinzip wird bei Spiele-Apps, welche anhand der Neigung und Rotation gesteuert werden, verwendet. Obwohl es sich um eine hybride App handelt, kann die Anwendung nur auf der Android-Plattform ausgeführt werden. Grund dafür ist das in dieser Arbeit verwendete Plug-In zum Auslesen von Daten der Sensoren. Dieses wurde nur für die Android-Plattform implementiert. Entwickelt wurde die App für die Android-Version 8.0.0.

4.2.1 Dengue-Detector-Kameraansicht

Wie in Abschnitt 3.1.4 beschrieben, wird das Erstellen von einer Rundumsicht durch einen Assistenten unterstützt. Dafür werden Markierungen in der Kameraansicht angezeigt, welche auf die Bewegung des Nutzers reagieren. Dadurch erscheint es dem Nutzer so, als würden die Markierungen in der Umgebung verankert sein. Der Nutzer begibt sich zu einem *Spot* und kann sich nun auf der Stelle drehen und bei jeder Markierung ein Foto erstellen. Dadurch wird die Umgebung gleichmäßig abgedeckt. Dieser Mechanismus benötigt die Orientierung des Geräts, um die Markierungen richtig zu positionieren. Die Vorgehensweise wird im Folgendem beschrieben.

Game-Rotation-Vector

Die Orientierung des Geräts kann mit dem sogenannten *Game-Rotation-Vector* ermittelt werden. Dieser Vektor stellt den Messwert eines virtuellen Sensors dar. Dieser greift auf *Accellerometer* und *Gyroscope* zurück. Beim *Accelerometer* handelt es sich um einen Beschleunigungssensor. Dieser liefert Werte bezüglich der Beschleunigung des Geräts auf der $x-$,$y-$ und $z-$Achse (siehe Abbildung 4.2). Mit Hilfe des *Gyroscops* lässt sich die Rotation des Geräts um die Achsen berechnen. Durch die Benutzung des *Game-Rotation-Vectors*, werden die vom *Accellerometer* und *Gyroscope* ausgelesenen Werte, durch Softwareoperationen vereinfacht und bekommen eine höhere Genauigkeit.

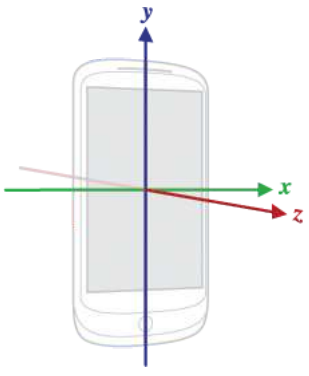

Abbildung 4.2 Koordinatensystem (relativ zum Gerät) [41]

Positionierung der Markierungen

Wie in Abbildung 3.4 zu sehen, ist neben dem Kamera-Icon auch ein „Fadenkreuz“ (grauer Kreis) in der Mitte des Displays vorhanden. Dieser wird benötigt, um die Markierungen anzuvisieren. Da *Ionic* standardmäßig das Erstellen von Panoramabildern nicht unterstützt, müssen mehrere Einzelbilder erstellt werden. Um eine Rundumsicht zu erhalten, muss zunächst der Kamerawinkel herausgefunden werden. Bei der Entwicklung wurde ein *Huawei P10* verwendet. Dieses hat einen Blickwinkel von etwa 51°. Um ein 360°-Bild zu erhalten, werden also knapp 7 Aufnahmen benötigt. Der Assistent muss also immer nach 51° ein Kamera-Icon positionieren. Um die aktuellen Winkel um die Achsen zu erhalten, müssen diese ständig ausgelesen werden. Dazu wurden zwei Methoden in der `camera.ts`-Klasse implementiert. Die Methode `readGameRotationVector` gibt den Winkel der Rotation um die y-Achse und die Methode `readAccelerometer` gibt die Winkel der Rotation um die x- und z-Achse an. Die Werte der Winkel um die x- und z-Achse werden benötigt um das Kamera-Icon auf einer virtuellen Horizontalen zu stabilisieren. Das Auslesen dieser Sensoren wird von *Ionic* nicht standardmäßig unterstützt, es wird das externe Plug-In *Cordova Sensor Plug-In* verwendet [42].

4.2.2 Kartenansicht

Um einen *Spot* einfacher zu erreichen, wurde in der *Dengue-Detector-* und *Mobile4d-*App eine Kartenansicht implementiert. Da *Mobile4d* bereits über das Open-Source Kartensystem *Leaflet* verfügt, wurde dieses ebenfalls in der *Dengue-Detector-*App verwendet. *Leaflet* ist eine *JavaScript*-Bibliothek, welche Karten für mobile Geräte zu Verfügung stellt. Um *Leaflet* zu nutzen, muss dieses über den *JavaScript* Paketmanager *npm* installiert und zum Projekt hinzugefügt werden. Nach der Installation können *Spots* und die aktuelle Position des Nutzers eingezeichnet werden. Für das Einzeichnen der *Spots*, werden diese zuerst vom Server angefordert. Die erhaltenen Geo-Koordinaten werden dann von der Methode `addMarkers()` auf der Karte eingezeichnet. Um die aktuelle Position des Nutzers zu erhalten, wird das GPS-Plug-In von *Ionic* verwendet. Das Plug-In verfügt über die Methode `watchPosition()`,

welche die Geo-Koordinaten des Nutzers ständig bereitstellt. Dadurch wird die Bewegung des Nutzers in der Karte dargestellt.

4.3 Implementierung des Chatbots

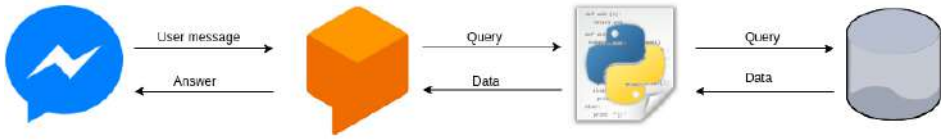

Abbildung 4.3 Übersicht der Komponenten zur Kommunikation mit dem Chatbot

In diesem Abschnitt wird die Implementierung und die Integration des Facebook-Chatbots beschrieben. In Abbildung 4.3 ist der Kommunikationsfluss zwischen den benötigten Komponenten zu sehen. Als erstes wird eine Verbindung zwischen Facebook und Dialogflow hergestellt. Über diesen Kanal kommuniziert der Nutzer mit dem Chatbot. Für spezielle Anfragen, die von Dialogflow nicht behandelt werden können (*Spots* anfordern und Bilder verschicken), muss ein *Webhook* eingerichtet werden. Durch einen *Webhook* kann Dialogflow auf externe APIs oder Datenbanken zugreifen. Dabei handelt es sich um eine *Cloud-App-Engine*, auf der ein Python-Skript ausgeführt wird. Das Python-Skript stellt eine Verbindung zum Server her, welcher *Spots* bereitstellt und Bilder entgegennimmt. Im Folgenden wird auf die Komponenten und deren Interaktion eingegangen.

4.3.1 Dialogflow Facebook Integration

Abbildung 4.4 Integration von Dialogflow in den Facebook-Chatbot

Um einen Facebook-Chatbot zu erstellen, muss zuerst ein Facebook-Account und eine Facebook-Seite angelegt werden. Über diese erfolgt die Konversation mit dem Nutzer. Als

nächstes muss eine Facebook-App erstellt werden. Dazu meldet man sich in der *Facebook Developer Console* an. Zum Erstellen einer Facebook-App wird ein App-Name sowie eine E-Mail-Adresse benötigt. Um eine Verbindung zwischen der Facebook-App und Dialogflow herzustellen, muss ein *Page Access Token* generiert werden. Es handelt sich dabei um eine generierte Zeichenkette. Der *Page Access Token* erlaubt einen administrativen Zugriff auf die Facebook-App. Nachdem eine Facebook-App erstellt wurde, wird ein *Agent* in Dialogflow erstellt. Dazu meldet man sich in der *Dialogflow Console* an, klickt auf *Create Agent* und vergibt einen Namen. Anschließend erfolgt die Integration mit der Facebook-App. Dazu muss im Menüpunkt *Integrations* der Facebook-Messenger ausgewählt werden. Dort wird der *Page Access Token* eingetragen und ein *Verify Token* erstellt. Zusätzlich wird eine *Callback URL* generiert. Der *Verify Token* und die *Callback URL* werden später für die Konfiguration des *Webhooks* benötigt. Ist die Integration abgeschlossen, kann mit dem Erstellen von *Intents* begonnen werden.

4.3.2 Intents

Abbildung 4.5 Dialogflow-Komponente zum Erstellen von Intents

Wie in Abschnitt 3.3.1 beschrieben, werden für das Verarbeiten der Nutzereingaben *Intents* benötigt. Nachdem ein neuer Chatbot erstellt wurde, verfügt dieser nur über den Standard `Default Welcome Intent`, welcher nur Begrüßungen behandelt. Um die Abläufe aus dem Abschnitt 3.3.2 abzubilden, wurden zusätzlich folgende *Intents* erstellt:

`Default Welcome Intent:` Der Standard `Default Welcome Intent` wurde um die Begrüßung seitens des Chatbots erweitert. Nach der Begrüßung durch den Nutzer stellt sich der Chatbot vor, äußert seine Absichten und fragt ob der Nutzer ihm helfen möchte. Dialogflow bietet die Möglichkeit, Antworten in Form von *quick-replies* vorzugeben. Wird durch den

Nutzer „*yes*“ angeklickt, so wird der `Default Welcome Intent-yes` ausgeführt. Möchte der Nutzer nicht helfen, so wird der `Default Welcome Intent-no` ausgeführt.

`Default Welcome Intent-yes:` Dieser *Intent* wird ausgeführt, falls der `Default Welcome Intent` mit „*yes*“ beantwortet wurde. In diesem Fall wird der Nutzer gebeten, seinen Standort mitzuteilen. Da Dialogflow standardmäßig keine Möglichkeit bereitstellt, eine Standortabfrage durchzuführen, wird diese mit Hilfe der Facebook API abgefragt. Dazu wird anstatt einer Textantwort ein *Custom Payload* verschickt (siehe Codeausschnitt 4.4). Dieser ist nichts anderes als eine *quick-reply*, um den Standort mitzuteilen.

```
{
 "facebook": {
  "text": "Please, send me your location?",
  "quick_replies": [
   {
   "content_type": "location"
   }
  ]
 }
}
```

Listing 4.4 Custom Payload für Facebook-Standortabfrage

`Default Welcome Intent-no:` Ausgeführt wird dieser *Intent* nachdem der `Default Welcome Intent` verneint wurde. In diesem Fall verabschiedet sich der Chatbot.

`Location Received:` Wurde dem Chatbot der Standort mitgeteilt, so wird dieser *Intent* ausgeführt. Als Antwort erhält der Nutzer eine Anleitung, um Bilder zu erstellen sowie eine Liste von *Spots*.

`Media Received:` Hat der Nutzer einen *Spot* ausgewählt, ist dort hingegangen und hat Fotos erstellt, sollen diese an den Chatbot geschickt werden. Nachdem der Nutzer das Foto an den Chatbot geschickt hat, wird dieser nochmal gebeten, seinen aktuellen Standort mitzuteilen. Diese Information wird benötigt, um das Foto mit einem Geo-Tag zu versehen. Nachdem das Foto und der Standort an den Chatbot geschickt wurden, wird der `Media Received - Location Received`-*Intent* ausgelöst.

`Media Received - Location Received:` Dieser *Intent* wird ausgelöst, nachdem der Chatbot das Foto vom *Spot* und dem dazugehörigen Standort erhalten hat. Als Antwort wird der Nutzer gefragt, ob er weiter helfen möchte. Falls ja, so bekommt er wieder eine Liste von Spots zugeschickt. Andernfalls verabschiedet sich der Chatbot.

4.3.3 Webhook

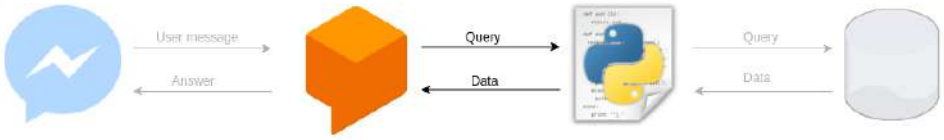

Abbildung 4.6 Webhook zur Kommunikation mit externen Services

Wie bereits erwähnt, kann Dialogflow nicht direkt an eine Datenbank angebunden werden. Dazu wird über einen *Webhook* eine Verbindung zu einer *Cloud-App-Engine* von Google hergestellt, auf welcher benutzerdefinierte Anwendungen ausgeführt werden können. In dieser Arbeit wird ein Python-Server verwendet. Dieser nimmt Anfragen von Dialogflow entgegen, verarbeitet diese und sendet eine Antwort. Auf diese Weise können *Spots* angefragt und Bilder in die Datenbank eingefügt werden. Ein *Intent* kann so konfiguriert werden, dass eine Anfrage über den *Webhook* gesendet wird, wenn dieser ausgelöst wird. Dies wurde bei folgenden *Intents* getan: `Location Received`, `Media Received` und `Media Received - Location Received`. Wann immer eines dieser *Intents* ausgelöst wird, wird die Nachricht des Nutzers über den *Webhook* an die *Cloud-App-Engine* gesendet. Dabei ist es egal, ob es sich um einen Standort, ein Bild oder eine Textnachricht handelt. Nachrichten werden in einem `Json`-Objekt versendet. Neben den Nachrichteninhalt wird auch eine *Action* versendet. Die *Action* ist ein Schlüssel, der eindeutig mit dem *Intent* assoziiert ist. Dadurch ist im Python-Skript bekannt, welches *Intent* behandelt werden muss. Für jedes *Intent* steht eine Unterfunktion bereit, die diesen behandelt. Um Dialogflow über einen *Webhook* mit der *Cloud-App-Engine* zu verbinden, wird eine URL benötigt. Diese wird von der *Cloud-App-Engine* generiert.

4.3.4 Python-Skript

Abbildung 4.7 Kommunikation zwischen dem Python-Server und der *Spot*- sowie Bilderdatenbank

Das Python-Skript dient als Kommunikationsschnittstelle zwischen Dialogflow und der Datenbank. Geschrieben wurde das Skript in der Python-Version 3.7. Das Skript implementiert mehrere Funktionen, um Anfragen von Dialogflow zu verarbeiten. Im Folgenden werden die Wichtigsten beschrieben.

`webhook`**:** Diese Funktion wird am Anfang jeder Anfrage ausgeführt. Sie nimmt Nachrichten entgegen und versendet Antworten. Ankommende Anfragen von Dialogflow werden an die Unterfunktion `processRequest(req)` weitergeleitet. Rückgabewerte der Unterfunktion werden wieder an Dialogflow zurück verschickt.

`processRequest(req)`**:** Diese Unterfunktion behandelt die eingehenden Anfragen. Durch `if-else`-Abfragen wird zuerst untersucht, um was für eine *Action* es sich handelt. Diese werden dann an die entsprechenden Unterfunktionen weitergeleitet. Die Rückgabewerte der Unterfunktionen werden dann an die Funktion `webhook()` zurückgeliefert.

`sendLocation(lat, lng)`**:** Diese Funktion wird ausgeführt, falls es sich um den `Location Received`-*Intent* handelt, also dann wenn der Nutzer sein Standort mitteilt. Erkannt wird dieses an der *Action* `locationReceived`. Als Parameter bekommt die Funktion Geo-Koordinaten, die für die Operation `SPOTS_IN_RANGE` benötigt werden (siehe Codeausschnitt 4.1). Als Rückgabewert liefert diese Funktion eine Liste von Spots, welche vom Server angefordert wurden.

`sendPictureURL(imageURL, lat, lng)`**:** Diese Funktion versendet Bilder und Geo-Koordinaten an den Server. Verschickt werden keine Bilddateien. Facebook versendet lediglich eine URL, unter welcher das Bild abrufbar ist. Die Geo-Koordinaten dienen als Geo-Tag für das Bild. Ausgelöst wird diese Funktion durch die *Action*: `Mediareceived.-Mediareceived-location_received` vom *Intent* `Media Received - Location Received`

Kapitel 5

Fazit und Ausblick

In diesem Kapitel wird die Idee dieser Arbeit noch einmal aufgegriffen sowie diskutiert. Dabei wird im Abschnitt 5.1 auf die Anwendungen, welche im Rahmen dieser Thesis entwickelt wurden, eingegangen. Des Weiteren werden die gewonnen Erkenntnisse besprochen. Im Abschnitt 5.2 wird Thematisiert, wie die Idee der Arbeit weitergeführt werden kann.

5.1 Fazit

Das Projekt „Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition“ hat es sich zur Aufgabe gemacht, mit Hilfe von Objekterkennung und -Klassifizierung, Brutstätten von Gelbfiebermücken automatisch zu erkennen und zu kartieren. Kann eine Korrelation zu den von der WHO vorgestellten Indikatoren festgestellt werden, so bietet dieser Ansatz eine effiziente Möglichkeit der Kartierung und Überwachung. Die benötigten Bilder stammen aus GSV. Es hat sich herausgestellt, dass diese Quelle bezüglich der Abdeckung und Aktualität nicht ausreichend ist. Im Rahmen dieser Arbeit wurden drei mobile Applikationen entwickelt, welche die GSV-Datenbank ergänzen. Die Aktualisierung der Datenbank soll durch *Crowdsourcing* erfolgen. Dazu sollen freiwillige Helfer GSV ähnliche Bilder erstellen. Eine Datenbank liefert den Nutzern Geo-Koordinaten von Orten, an denen Bilder aktualisiert oder neu erstellt werden müssen. Erstellte Bilder werden dann an eine Datenbank gesendet, wo sie auf BS untersucht werden.

Für die Einarbeitung in die Thematik, wurde zunächst eine intensive Literaturrecherche durchgeführt. Es wurden dabei mehrere Systeme untersucht, die zur Kontrolle und Überwachung des Dengue-Fieber dienen. Diese lassen sich in drei Kategorien einordnen: 1) Systeme zum Erstellen von Vorhersagemodellen, 2.) Systeme zum Melden von BS und 3) Systeme zur automatisierten Erkennung von potentiellen BS. Die Idee dieser Arbeit war es, Systeme der Kategorien 2) und 3) zu verbinden. Es wurden dafür drei mobile Anwendungen entwickelt: *Dengue-Detector*-App, ein Facebook Chatbot und es wurde eine Integration des Dengue-Moduls in das Reporting Alerting System *Mobile4D* vorgenommen. Die Recherche lieferte ähnliche Systeme, jedoch nicht in der Kombination mit automatisierter Erkennung von potenziellen BS. Obwohl alle drei Anwendungen nach dem gleichen Prinzip arbeiten (*Spots* Anfordern, zum *Spot* hingehen, *Spot* fotografieren, Bilder abschicken), haben diese unterschiedliche Herangehensweisen. Als Erstes wurde die mobile *Dengue-Detector*-App vorgestellt (siehe Abschnitt 3.1). Diese verfügt über einen Assistenten, um eine Rundumsicht zu erstellen. In der Kameraansicht wurden Markierungen eingefügt, welche den Nutzer dabei anleiten, mehrere Einzelbilder zu erstellen. Um die Markierungen zu positionieren, werden Sensordaten des Smartphones benötigt. Ein weiterer Teil der Arbeit war die Integration des Dengue-Moduls in das Reporting-Alerting-System *Mobile4D* (siehe Abschnitt 3.2). Neben dem Melden von Umweltkatastrophen ist es nun möglich die Bilderdatenbank zu aktualisieren. Neben den beiden Android-Anwendungen wurde auch ein Facebook-Chatbot entwickelt (siehe Abschnitt 3.3). Dieser hat gegenüber den Apps den Vorteil der Plattformunabhängigkeit: Es muss keine zusätzliche App heruntergeladen werden, die Nutzer müssen sich lediglich über den Facebook-Messenger mit den Chatbot verbinden. Da der Facebook-Messenger in Thailand sehr verbreitet ist, wird eine höhere Anzahl an potentiellen Nutzern erwartet als bei den Android-Apps. Für die Entwicklung des Chatbots wurde das *Natural-Language-Processing*-Framework *Dialogflow* verwendet. Dieses ist dafür verantwortlich, Nachrichten die vom Nutzer verschickt wurden, zu verarbeiten und Antworten zu generieren. Da es nicht möglich ist, *Spots* direkt von *Dialogflow* an den Chatbot zu verschicken, wurde über einen *Webhook* eine Verbindung zwischen *Dialogflow* und einem *Python*-Server hergestellt (siehe Abschnitt 4.3.3). Der *Python*-Server sowie die beiden Android-Apps, kommunizieren über

eine REST-Schnittstelle mit dem Server, der die Bilderdatenbank und Geo-Koordinaten von *Spots* enthält. Zusammengefasst wurden im Rahmen dieser Arbeit drei mobile Anwendungen entwickelt, welche die Überwachung und Kontrolle des Dengue-Fiebers mit Hilfe von Informationstechnologien unterstützen sollen. Diese heben sich gegenüber den bestehenden Systemen dadurch ab, dass nicht nur Online-Quellen (zum Teil nicht aktuelle oder nicht vorhandene Bilder) verwendet werden, um Bilder von potentiellen BS zu erhalten. Mit Hilfe von freiwilligen Helfern (*Crowdsourcing*) kann eine neue Informationsquelle erschlossen werden. Zusätzlicher Vorteil gegenüber Apps wie *Mo-Buzz*, *Veta* oder *Target Zika* ist die automatisierte Erkennung von BS. Dadurch wird eine höhere Skalierbarkeit gewährleistest und somit können größere Gebiete untersucht werden.

Während der Entstehung der Arbeit konnten viele neue Erkenntnisse gewonnen werden. Angefangen bei den Möglichkeiten der Überwachung und Kontrolle des Dengue-Fiebers durch den Einsatz von Informationstechnologien. Im Rahmen dieser Arbeit wurden zwei Ideen miteinander verbunden: Das Melden von BS mit Hilfe von mobilen Anwendungen und die automatisierte Erkennung von BS. Für die Implementierung wurden zum Teil bereits bekannte Technologien wie das *Ionic*-Framework verwendet, mit dem die *Dengue-Detector*-App und das Dengue-Modul für *Mobile4D* entwickelt wurden. Ein vollkommen unbekannter Ansatz war die Entwicklung eines Facebook-Chatbots. Dazu gehörte auch der Einsatz des NLP-Framework *Dialogflow*, dessen Funktionalitäten zuerst untersucht und erlernt werden mussten. Dadurch konnten Erkenntnisse gewonnen werden, wie Chatbots funktionieren und in welchen Bereichen diese eingesetzt werden können. Neu war ebenfalls der Ansatz, einen Chatbot als *Crowdsourcing*-Werkzeug einzusetzen. Keiner der im Rahmen dieser Arbeit untersuchten Chatbots wurde als solches genutzt. Diese agierten meistens als Assistenten um eine Aufgabe zu erledigen.

5.2 Ausblick

Wie im Abschnitt Fazit erwähnt, wurden drei mobile Anwendungen entwickelt, die durch den Einsatz von *Crowdsourcing* eine Bilderdatenbank ergänzen sollen. Diese wird dann auf

BS untersucht. Als Fortführung der Arbeit könnte zum einem eine Evaluation der Systeme vorgenommen werden. Diese könnte vor Ort in Thailand stattfinden, da dort bereits eine Datenbank mit *Spots* vorhanden ist. Durch diese Untersuchung könnte man Rückschlüsse über die Interaktion der Nutzer mit den Anwendungen gewinnen, sowie über die Tauglichkeit der Bilder, die auf BS untersucht werden. Die Evaluation könnte in zwei Phasen stattfinden: Zuerst sollte eine Laborstudie unter kontrollierten Bedingungen stattfinden. Dadurch könnten erste Erkenntnisse gewonnen werden, wie potentielle Nutzer mit den Applikationen interagieren. Bei Bedarf können dann Änderungen vorgenommen werden. Die zweite Phase wäre dann eine Feldstudie. Dieser Ansatz wird in der Komplementärarbeit „Implementation and Evaluation of a Chatbot to Crowdsource Geo-Tagged Images to Detect Mosquito Breeding Sites“ von M. Dechert verfolgt. Die Evaluation wird an der Mahidol University in Thailand durchgeführt. Für diesen Zweck wurde eine Datenbank von *Spots*, welche sich auf dem Campus der Universität befinden, erstellt. Diese sollen durch den Einsatz des Chatbots, von Studierenden abgearbeitet werden. Dadurch können Informationen über das Nutzerverhalten gewonnen werden, wie beispielsweise die Dauer und Intensität der Nutzung. Diese Art von Evaluation könnte auch für die *Dengue-Detector*-App und *Mobile4D* durchgeführt werden. Des Weiteren könnte untersucht werden, wie man Menschen dazu motivieren kann an solchen Projekten teilzunehmen. Da das Dengue-Fieber nicht nur in Thailand, sondern in ganz Südostasien verbreitet ist, könnte das System auch in anderen Ländern zum Einsatz kommen. Da das Reporting-Alerting-System *Mobile4D* sich bereits in Laos im Einsatz befindet und in Rahmen dieser Arbeit das Dengue-Modul integriert wurde, bietet es sich an, das System auch dort einzusetzen. Dazu müsste jedoch zuerst eine Datenbank mit *Spots* eingerichtet werden. Des Weiteren könnte eine Implementierung des Assistenten zum Erstellen von Bildern in *Mobile4D* vorgenommen werden. Ein weiterer interessanter Aspekt, der untersucht werden kann, ist die Tauglichkeit von Chatbots für *Crowdsourcing*. In dieser Arbeit wird ein Chatbot verwendet um geo-getaggte Bilder von BS zu erhalten. Dieser Ansatz könnte auch genutzt werden, um Informationen über andere Erkrankungen zu sammeln, wie es zum Beispiel die Plattform *Flu Near You* tut. Es könnte genauso auch eine Integration der *Mobile4D*-App in den Chatbot vorgenommen werden. Somit könnten Reports über einen Chatbot erstellt wer-

den. Insgesamt hat die Beschäftigung mit dem Thema der Überwachung und Kontrolle von Dengue mit der Unterstützung von IT gezeigt, dass ein enormes Potenzial besteht, dass auch zukünftig Anwendungen (weiter-)entwickelt werden, mit Hilfe welcher man BS ausfindig machen kann.

Literaturverzeichnis

[1] N. Vasilakis and S. C. Weaver, "Chapter 1 the history and evolution of human dengue emergence," in *Advances in Virus Research*, pp. 1–76, 2008.

[2] S. Bhatt, P. W. Gething, O. J. Brady, J. P. Messina, A. W. Farlow, C. L. Moyes, J. M. Drake, J. S. Brownstein, A. G. Hoen, O. Sankoh, M. F. Myers, D. B. George, T. Jaenisch, G. R. W. Wint, C. P. Simmons, T. W. Scott, J. J. Farrar, and S. I. Hay, "The global distribution and burden of dengue," *Nature*, vol. 496, pp. 504–507, Apr. 2013.

[3] World Health Organization, *Dengue: Guidelines for Diagnosis, Treatment, Prevention and Control*. World Health Organization, 2009.

[4] D. M. B. N. Dr Philip McCall, Dr Linda Lloyd, *Dengue: Guidelines for Diagnosis, Treatment, Prevention and Control: New Edition*. Geneva: World Health Organization, June 2013.

[5] K. L. Laura C. Harrington, Thomas W. Scott, "Dispersal of the dengue vector aedes aegypti within and between rural communities," *The American Society of Tropical Medicine and Hygiene*, pp. 209–220, 2005.

[6] A. Y. Chang, M. E. Parrales, J. Jimenez, M. E. Sobieszczyk, S. M. Hammer, D. J. Copenhaver, and R. P. Kulkarni, "Combining google earth and GIS mapping technologies in a dengue surveillance system for developing countries," *Int. J. Health Geogr.*, vol. 8, no. 1, p. 49, 2009.

[7] M. O. Lwin, S. Vijaykumar, O. N. N. Fernando, S. A. Cheong, V. S. Rathnayake, G. Lim, Y.-L. Theng, S. Chaudhuri, and S. Foo, "A 21st century approach to tackling dengue: Crowdsourced surveillance, predictive mapping and tailored communication," *Acta Trop.*, vol. 130, pp. 100–107, Feb. 2014.

[8] K. S. Nicholas G. Reich, Stephen A. Lauer, "Challenges in real-time prediction of infectious disease: A case study of dengue in thailand," *Online J. Public Library of Science*, vol. 7, no. 1, 2016.

[9] E. Boakes, G. Gliozzo, V. Seymour, M. Harvey, C. Smith, D. B. Roy, and M. Haklay, "Patterns of contribution to citizen science biodiversity projects increase understanding of volunteers' recording behaviour," *Scientific Reports*, vol. 6, p. 33051, 09 2016.

[10] M. Mehra, A. Bagri, X. Jiang, and J. Ortiz, "Image analysis for identifying mosquito breeding grounds," in *2016 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, 2016.

[11] C. Suduwella, A. Amarasinghe, L. Niroshan, C. Elvitigala, K. De Zoysa, and C. Keppetiyagama, "Identifying mosquito breeding sites via drone images," in *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications - DroNet '17*, 2017.

[12] H. M. Aburas, B. Gultekin Cetiner, and M. Sari, "Dengue confirmed-cases prediction: A neural network model," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4256–4260, 2010.

[13] H. A. Carneiro and E. Mylonakis, "Google trends: a web-based tool for real-time surveillance of disease outbreaks," *Clin. Infect. Dis.*, vol. 49, pp. 1557–1564, Nov. 2009.

[14] B. M. Althouse, Y. Y. Ng, and D. A. T. Cummings, "Prediction of dengue incidence using search query surveillance," *PLoS Negl. Trop. Dis.*, vol. 5, p. e1258, Aug. 2011.

[15] M. O. Lwin, K. Jayasundar, A. Sheldenkar, R. Wijayamuni, P. Wimalaratne, K. C. Ernst, and S. Foo, "Lessons from the implementation of Mo-Buzz, a mobile pandemic surveillance system for dengue," *JMIR Public Health Surveill*, vol. 3, p. e65, Oct. 2017.

[16] O. P. Wójcik, J. S. Brownstein, R. Chunara, and M. A. Johansson, "Public health for the people: participatory infectious disease surveillance in the digital age," *Emerg. Themes Epidemiol.*, vol. 11, p. 7, June 2014.

[17] S. M. Quadri, T. K. Prashanth, S. Pongpaichet, A. A. A. Esmin, and R. Jain, "TargetZIKA: Epidemic situation detection and risk preparedness for ZIKA virus," in *2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)*, 2017.

[18] S. Pongpaichet, M. Tang, L. Jalali, and R. Jain, "Using photos as Micro-Reports of events," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval - ICMR '16*, 2016.

[19] J. Weizenbaum, "ELIZA—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[20] R. Dale, "The return of the chatbots," *Natural Language Engineering*, vol. 22, pp. 811–817, Sept. 2016.

[21] M. Heo and K. J. Lee, "Chatbot as a new business communication tool: The case of naver talktalk," *Business Communication Research and Practice*, vol. 1, no. 1, pp. 41–45, 2018.

[22] G. Cameron, D. Cameron, G. Megaw, R. Bond, M. Mulvenna, S. O'Neill, C. Armour, and M. McTear, "Towards a chatbot for digital counselling," in *Proceedings of the 31st British Computer Society Human Computer Interaction Conference*, HCI '17, (Swindon, UK), pp. 24:1–24:7, BCS Learning & Development Ltd., 2017.

[23] A. Lokman and J. Mohamad Zain, "Designing a chatbot for diabetic patients," 08 2007.

[24] D. Madhu, C. J. N. Jain, E. Sebastain, S. Shaji, and A. Ajayakumar, "A novel approach for medical assistance using trained chatbot," in *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 243–246, March 2017.

[25] D. S. I. V. I. S. P. M. K. D. S, "A self-diagnosis medical chatbot using artificial intelligence," *Journal of Web Development and Web Designing*, vol. 3, 2018.

[26] D. Mitry, T. Peto, S. Hayat, J. E. Morgan, K.-T. Khaw, and P. J. Foster, "Crowdsourcing as a novel technique for retinal fundus photography classification: Analysis of images in the epic norfolk cohort on behalf of the ukbiobank eye and vision consortium," *PloS one*, vol. 8, no. 8, p. e71154, 2013.

[27] L. Frommberger and F. Schmid, "Mobile4d: crowdsourced disaster alerting and reporting," in *Proceedings of the Sixth International Conference on Information and Communications Technologies and Development: Notes-Volume 2*, pp. 29–32, ACM, 2013.

[28] O. Okolloh, "Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information," vol. 59, pp. 65–70, 06 2009.

[29] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

Online-Quellen

[30] S. Perez, “Majority of u.s. consumers still download zero apps per month, says comscore.” https://techcrunch.com/2017/08/25/majority-of-u-s-consumers-still-download-zero-apps-per-month-says-comscore/, Aug. 2017. Accessed: 2018-8-13.

[31] S. Globalstats, “Desktop vs mobile vs tablet market share asia.” http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/asia/, 2018. Accessed: 2018-9-1.

[32] S. Ho, “Why southeast asia is leading the world’s most disruptive mobile business models.” https://techcrunch.com/2015/09/08/why-southeast-asia-is-leading-the-worlds-most-disruptive-mobile-business-models/, Sept. 2015. Accessed: 2018-6-13.

[33] W. are social, “Penetration of leading social networks in thailand as of 3rd quarter 2017.” https://www.statista.com/statistics/284483/thailand-social-network-penetration/, 2018. Accessed: 2018-9-1.

[34] Veta, “Veta - fight dengue.” https://veta.life/, 2017. Accessed: 2018-5-16.

[35] U. B. Social Apps Lab at CITRIS, “DengueChat.” https://www.denguechat.com/, 2017. Accessed: 2018-5-16.

[36] Wikipedia, “Eliza.” https://de.wikipedia.org/wiki/ELIZA, 2018. Accessed: 2018-09-26.

[37] S. Catalog, “Alexa amazon.” https://www.flickr.com/photos/stockcatalog/40095307924, 2018. Accessed: 2018-09-26.

[38] gyant Inc., “Gyant Gesundheits-Roboter.” https://gyant.com/deutsch, 2017. Accessed: 2018-5-31.

[39] “A comparative analysis of chatbots apis.” https://medium.com/activewizards-machine-learning-company/a-comparative-analysis-of-chatbots-apis-f9d240263e1d, Apr. 2018. Accessed: 2018-8-13.

[40] A. Kang, “Understanding the differences between alexa, api.ai, wit.ai, and luis/-cortana.” https://medium.com/@abraham.kang/understanding-the-differences-between-alexa-api-ai-wit-ai-and-luis-cortana-2404ece0977c, Apr. 2017. Accessed: 2018-8-11.

[41] Google, “Android sensor types.” https://source.android.com/devices/sensors/sensor-types, 2018. Accessed: 2018-08-15.

[42] F. Rogerios, “Cordova sensors plugin.” https://github.com/fabiorogeriosj/cordova-plugin-sensors, Mar. 2015. Accessed: 2018-2-17.

Anhang A

CD-Inhalt

A.1

- Digitale Version der Thesis
- README.txt mit Einleitung zur Ausführung der Anwendungen

A.2 Dengue-Detector-App

- Ausführbare Dengue-Detector-apk
- Quellcode der Dengue-Detector-App
- Videoaufzeichnung der Dengue-Detector-App

A.3 Mobile4D

- Ausführbare Mobile4D-apk
- Quellcode der Mobile4D-App
- (Quellcode ist zusätzlich auf dem Git-Repository des Projekts Mobile4D verfügbar)

A.4 Chatbot

- Quellcode des Chat-Bots (Dialogflow)
- Quellcode des Python-Server (Stand 30.08.2018)
- Videoaufzeichnung der Konversation mit dem Chatbot
- Link zum Chatbot https://www.messenger.com/t/DengueDetector (Login-Daten in der README.txt)