Masterarbeit

im Studiengang

Informatik

Konzeptionierung einer App zur Reduktion von Messkosten und subjektiven Fehlereinflüssen für die Validierung von Fernerkundungsdaten

von

Jan Füsting
13688

Erstprüfer:	Prof. Dr. Christian Bunse
Zweitprüfer:	Hon.-Prof. Dr. Erik Borg
Unternehmen:	Deutsches Zentrum für Luft- und Raumfahrt e.V.
Eingereicht am:	09. Oktober 2019

Vorwort

Die verfasste Masterarbeit „*Konzeptionierung einer App zur Reduktion von Messkosten und subjektiven Fehlereinflüssen für die Validierung von Fernerkundungsdaten*“ wurde am Deutschen Zentrum für Luft- und Raumfahrt e.V. am Standort Neustrelitz geschrieben. Dabei möchte ich mich besonders bei meinem Betreuer Hon.-Prof. Dr. Erik Borg bedanken, der mir das Thema während meines Sommerjobs nahegelegt hat, und das Zweitgutachten übernommen hat.

Auch möchte ich mich bei Herr Klaus-Dieter Mißling, Frau Dr. Friederike Klan, Frau Sina Truckenbrodt und Herr Carsten Pathe vom DLR bedanken. Sie beantworteten mir stets meine Fragen und halfen mir meine Forschungen voranzubringen.

Insbesondere möchte ich mich bei Herrn Prof. Dr. Christian Bunse bedanken. Er hat das Erstgutachten übernommen und mich stets bei meinen Fragen und Anliegen hilfreich unterstützt.

Schließlich möchte ich meinen Eltern und Freunden danken, die mich während meines gesamten Studiums unterstützt haben und mit klugen Ratschlägen zur Seite standen.

Jan Füsting

Abstract

Diese Arbeit befasst sich mit der Entwicklung einer modular erweiterbaren Applikation für mobile Geräte, im weiteren als App bezeichnet. Diese soll plattformunabhängig auf Smartphones und Tablets lauffähig sein. Es wird auf mehrere Frameworks eingegangen, welche für die Entwicklung verwendet werden können. Zudem wird ein Konzept für die Softwarearchitektur entwickelt und erläutert. Dieses beschreibt den Aufbau der App. Sie beschreibt der Aufbau der App. Als weiteren Bestandteil des Konzepts wird ein Design für die Usability und die User Experience entwickelt, welche in einem Prototypen teilweise implementiert werden. Das entwickelte Konzept mit Systemarchitektur und Design wird gemeinsam mit dem zugehörigen Prototypen in einer Diskussion bewertet. Ebenfalls werden die bei der Entwicklung aufgetretenen Fehler beurteilt und analysiert.

Inhaltsverzeichnis

1. Einleitung **1**

2. Grundlagen **3**
2.1. Begriffsdefinitionen . . . 3
2.2. Verwendete Programme . . . 4
2.3. Datenschutz . . . 6
2.4. Frameworks . . . 7
2.4.1. Native . . . 7
2.4.2. Web-App . . . 9
2.4.3. Hybrid-App . . . 10
2.4.4. Vergleich der Frameworks . . . 11
2.5. Anwendungsfälle . . . 13
2.5.1. Forschungsprojekt DEMMIN . . . 13
2.5.2. Erfassung von Stadtverbesserungen . . . 14

3. Stand der Forschung **15**
3.1. Spezifische Frameworks zur Methodenentwicklung . . . 15
3.2. Sensorerfassung . . . 17
3.3. Studien zur Datenerfassung via Smartphone . . . 17
3.4. Usability und User Experience (UX) . . . 18

4. Konzept **19**
4.1. Projektaufbau . . . 19
4.1.1. DLR_Data_App . . . 19
4.1.2. DLR_Data_App.Android . . . 19
4.1.3. DLR_Data_App.iOS . . . 20
4.1.4. DLR_Data_App_Test . . . 20
4.2. Anforderungen . . . 21
4.2.1. Nicht funktionale Anforderungen . . . 21
4.2.1.1. Generelle Anforderungen . . . 21
4.2.1.2. Anforderungen an die Nutzerfreundlichkeit . . . 21
4.2.1.3. Anforderungen an die Daten-Sicherheit . . . 22
4.2.2. Funktionale Anforderungen . . . 22
4.2.2.1. Authentifizierung . . . 22
4.2.2.2. Benutzerkonto . . . 22
4.2.2.3. Individuelle App-Funktionen . . . 23
4.2.2.4. Prüfmodule . . . 23
4.2.2.5. Basismodule . . . 23
4.2.2.6. Profiling . . . 23
4.2.2.7. Feldkampagne . . . 24
4.3. Übersicht der Funktionalität . . . 25
4.3.1. Login . . . 25
4.3.2. Menü . . . 25
4.3.3. Einstellungen . . . 26
4.4. Entwurf . . . 27
4.4.1. Vorgehen beim Entwurf . . . 27

4.4.2. Entwurfsmuster . . . 27
4.5. Open Data Kit . . . 29
4.6. Kommunikationsprotokolle . . . 30
4.6.1. Übertragungsformate . . . 30
4.6.2. Verschlüsselung . . . 31
4.7. Szenarien im Konzept . . . 32
4.7.1. Neues Profil anmelden . . . 32
4.7.2. Hauptmenü . . . 35
4.7.3. Projekt auswählen . . . 36
4.7.4. Projekt erstellen . . . 38
4.7.5. Projekt hinzufügen . . . 38
4.7.6. Daten erfassen . . . 40
4.7.7. Daten bearbeiten . . . 43
4.8. Datenbank . . . 45
4.9. Elemente . . . 47
4.9.1. Profil . . . 47
4.9.2. Projekt . . . 47
4.9.3. Fragebogen . . . 47
4.10. Module . . . 48
4.10.1. Allgemein . . . 48
4.10.2. Login . . . 48
4.10.3. Einstellungen . . . 48
4.10.4. Projekte . . . 49
4.10.5. Services . . . 49
4.11. Sichten . . . 50
4.11.1. Kontextabgrenzung . . . 50
4.11.2. Bausteinansicht . . . 51
4.11.3. Laufzeitsicht . . . 56

5. Usability und User Experience (UX) 57
5.1. Standards . . . 57
5.1.1. ISO 9241-11 . . . 57
5.1.2. ISO 9241-210 . . . 57
5.1.3. Android Design Guidelines . . . 57
5.1.4. iOS Human Interface Guidelines . . . 57
5.2. Heuristiken . . . 58
5.3. Personas . . . 59
5.3.1. Datenerhebung . . . 59
5.3.2. Inhalt Persona . . . 59
5.3.3. Entwickelte Personas . . . 61
5.4. Design . . . 63
5.4.1. Farben . . . 63
5.4.2. Navigation . . . 63
5.4.3. Header . . . 63
5.4.4. Formulare . . . 63
5.5. Szenarien . . . 64
5.5.1. Neues Profil anmelden . . . 64
5.5.2. Projekt erstellen . . . 64
5.5.3. Projekt hinzufügen . . . 65
5.5.4. Projekt auswählen . . . 65
5.5.5. Daten erfassen . . . 65
5.5.6. Daten bearbeiten . . . 65

6. Prototyp **66**
6.1. Verwendete Technologien . . . 66
6.2. Services . . . 67
6.2.1. Sensoren . . . 67
6.2.2. Helfer . . . 68
6.2.3. Datenbank . . . 68
6.3. Implementierte Funktionen . . . 69
6.3.1. Login . . . 69
6.3.2. Profilerstellung . . . 69
6.3.3. Einstellungen . . . 69
6.3.4. Sensortest . . . 69
6.3.5. Projektverwaltung . . . 69
6.3.6. Impressum . . . 70
6.3.7. Unterstützung mehrerer Sprachen . . . 70
6.4. Tests . . . 71

7. Validierung und Verifikation **72**
7.1. Nicht-funktionale Anforderungen . . . 72
7.1.1. Generelle Anforderungen . . . 72
7.1.2. Anforderungen an die Nutzerfreundlichkeit . . . 72
7.1.3. Anforderungen an die Daten-Sicherheit . . . 73
7.2. Funktionale Anforderungen . . . 73
7.2.1. Authentifizierung . . . 73
7.2.2. Benutzerkonto . . . 73
7.2.3. Individuelle App-Funktionen . . . 73
7.2.4. Prüfmodule . . . 73
7.2.5. Basismodule . . . 73
7.2.6. Profiling . . . 74
7.2.7. Feldkampagne . . . 74

8. Diskussion **75**
8.1. Hypothesen . . . 75
8.1.1. Verbesserung der Qualität der Messwerteingabe . . . 75
8.1.2. Modulare Erweiterbarkeit . . . 75
8.1.3. Generierung einer dynamischen Oberfläche . . . 75
8.2. Usability und User Experience mittels Heuristiken . . . 75
8.3. Aufgetretene Probleme . . . 77
8.3.1. Konzept . . . 77
8.3.2. Programmierung . . . 77

9. Fazit **79**

A. Datenschutzerklärung **81**

B. Beispiel JSON **85**

Glossar **87**

Literaturverzeichnis **90**

Tabellenverzeichnis

2.1. Vergleich der Frameworks Nativ und Web-Apps 11
2.2. Vergleich der Frameworks Hybrid-Apps 11

3.1. Vergleich der Frameworks zur Datenerfassung 17

4.1. Tabelle User 45
4.2. Tabelle Projekt User Connection 45
4.3. Tabelle Projekt Formular Elemente 45
4.4. Tabelle Projekt Form 46
4.5. Tabelle Projekt Formular Metadata 46
4.6. Tabelle Projekt 46
4.7. Tabelle Projekt Formular Elementliste 46

5.1. Fiktive Personas Projektersteller 61
5.2. Fiktive Personas Teilnehmer Forschungsprojekt 61
5.3. Fiktive Personas Anwender öffentliches Projekt 62
5.4. Aufbau Eingabefelder 63

6.1. Verwendete NuGet Pakete 66

7.1. Verifikation genereller Anforderungen 72
7.2. Datensicherheit von NuGet Paketen 73

Abbildungsverzeichnis

1.1. Entwicklungsablauf 2

2.1. Statistik mobiler Betriebssysteme 7
2.2. Logo Forschungsprojekt DEMMIN 13
2.3. Feldkampagne 14

3.1. Quellen UX 18

4.1. MVVM Muster 27
4.2. Open Data Kit Build Webapp 29
4.3. Modell User 32
4.4. Programmablaufplan Login 33
4.5. Programmablaufplan Neues Profil 34
4.6. Programmablaufplan Hauptmenü 35
4.7. Modell Projekt 36
4.8. Programmablaufplan Alle Projekte 37
4.9. Programmablaufplan Projekt laden 39
4.10. Programmablaufplan UI generieren 41
4.11. Programmablaufplan Aktuelles Projekt 42
4.12. Programmablaufplan Daten bearbeiten 44
4.13. Kontextabgrenzung 50
4.14. Bausteinansicht Gesamt 51
4.15. Bausteinansicht Login 52
4.16. Bausteinansicht Neues Profil 52
4.17. Bausteinansicht Einstellungen 53
4.18. Bausteinansicht Projektverwaltung 54
4.19. Bausteinansicht Aktuelles Projekt 54
4.20. Bausteinansicht Sensortest 55
4.21. Bausteinansicht Impressum 55
4.22. Gesamte Laufzeitansicht 56

1. Einleitung

In der Forschung werden in unterschiedlichsten Bereichen Daten erhoben und ausgewertet. Die Erfassung von Messdaten unterstützt die Forschung dabei, neue Erkenntnisse daraus herzuleiten. Um einen möglichst großen und aussagekräftigen Datensatz zu generieren, werden in vielen Fällen zahlreiche Teilnehmer mit in die Gewinnung eingebunden, die präzise Daten mit geringen Fehlern erstellen können sollen. Dabei bietet es sich an, die Smartphones der Teilnehmer zur Erfassung zu verwenden. Dieses Vorgehen bietet mehrere Vorteile:

- weite Verbreitung von Smartphones,
- Unterstützung der Anwender mit Hilfetexten,
- genauere Messwerte durch verbaute Sensoren.

Das Deutsche Zentrum für Luft- und Raumfahrt e.V. (DLR) hat 1999 das Projekt Durable Environmental Multidisciplinary Monitoring Information Network (DEMMIN) ins Leben gerufen. Es geht darum, das Wachstum der Pflanzen zu dokumentieren und dadurch Sensoren in Satellitensystemen zu konfigurieren. Bei der Erfassung der Messwerte helfen Studenten, welche auf einem Feld Messungen unternehmen. Hierbei verwenden die Studenten Papierformulare und Messgeräte. Daraus entstand die Idee von Sina Truckenbrodt (68) für die Entwicklung einer mobilen Anwendung, mit deren Hilfe die In-situ Messungen im Feld optimiert und vereinfacht werden können, um auch die möglichen Fehler zu minimieren.

An dem Projekt DEMMIN ist zudem das in Jena ansässige neue Institut für Datenwissenschaften als Teil des DLR und die Friedrich-Schiller-Universität Jena beteiligt. Dort wird unter anderem am Thema Citizen Sciences (Bürgerwissenschaften) geforscht. Es geht dabei um die Bürgerbeteiligung bei Forschungprojekten, sowie der Einbindung der Bürger bei der Konzeption, Vorbereitung und Durchführung wissenschaftlicher Arbeiten. Es gibt bereits entsprechende Projekte, welche sich mit dieser Fragestellung befassen und Software anbieten, diese kann dafür genutzt werden, Daten mit dem Smartphone zu erfassen. Ein wesentlicher Teil der Erfassung von Daten ist zudem die Erfassung des Wissensstands des Anwenders. Daraus lassen sich die Messwerte gewichten und genauer auswerten. Viele bestehende mobile Anwendungen zur Datenerfassung sind nicht plattformunabhängig und bieten nicht die Möglichkeit mehrere Projekte zu verwalten.

Dementsprechend befasst sich die Masterarbeit mit der Erstellung einer mobilen Anwendung, nachfolgend App genannt, welche die Erfassung von Messdaten im Rahmen von Forschungsprojekten verbessern soll. Das Ziel der App ist die verbesserte Erfassung der Messdaten, die modulare Erweiterbarkeit der App mit weiteren Projekten, sowie die Möglichkeit ein Profil über den Anwender zu erstellen. Dabei ist zu evaluieren, welche Möglichkeiten das eingesetzte Framework für die Entwicklung bietet und wie die Softwarearchitektur entsprechend aufgebaut werden muss.

Es wurde von Mitarbeitern des DLR ein Anforderungskatalog entwickelt, der u.a. Plattformunabhängigkeit, Profilverwaltung und Projektverwaltung umfasst, entwickelt und bereitgestellt wird. Da der Umfang des Anforderungskatalogs sehr groß ist, wurde sich bei der Realisierung auf eine Untermenge der Anforderungen in Absprache mit dem DLR beschränkt. Die Entwicklung der Anwendung findet arbeitsteilig mit Studenten der Friedrich Schiller Universität Jena statt welche die Entwicklung des Profilings des Anwenders übernehmen. Diese Arbeit befasst sich mit der Entwicklung des Grundgerüsts der App, der Verwaltung der Profile, Verarbeitung von Projekten und Erfassung von Messdaten für die Projekte.

Die App wird in mehreren Etappen entwickelt. Dabei wurden zu Beginn der Entwicklungen alle Anforderungen an die App mit den Mitarbeitern des DLR besprochen und festgelegt, in welchem Umfang diese implementiert werden. Im Anschluss daran werden die unterschiedlichen Frameworks untersucht. Dabei wird darauf geachtet, dass das Framework möglichst viele Anforderungen erfüllt. Durch eine Literaturrecherche wird der aktuelle Stand der Entwicklung vergleichbarer Apps untersucht. Zudem werden die o.g. Anforderungen in Studien untersucht, welche auf die Reduzierung von Kosten, sowie die Verbesserung der Messergebnisse eingehen. Danach wird die Architektur der App entwickelt. Unter anderem werden Programmablaufpläne, als auch das Programmiermodell und Konzepte zum Aufbau der Datenbank beschrieben. Ein wichtiger Teil der Arbeit ist zudem die Entwicklung einer anwenderorientierten Menüführung und Design. Dies wird als Usability und User Experience (UX) bezeichnet und sorgt für mehr freiwilliges Engagement des Anwenders. Noch während der Konzeptionierung und der Entwicklung des Designs der App wird ein Prototyp entwickelt, in dem die Konzepte teilweise implementiert werden. In der nächsten Etappe wird in der Diskussion die Entwicklung der App besprochen und Ergebnisse, sowie auch Probleme, die bei der Entwicklung aufgetreten sind. In dem nachfolgenden Diagramm 1.1 ist der Entwicklungsablauf dargestellt.

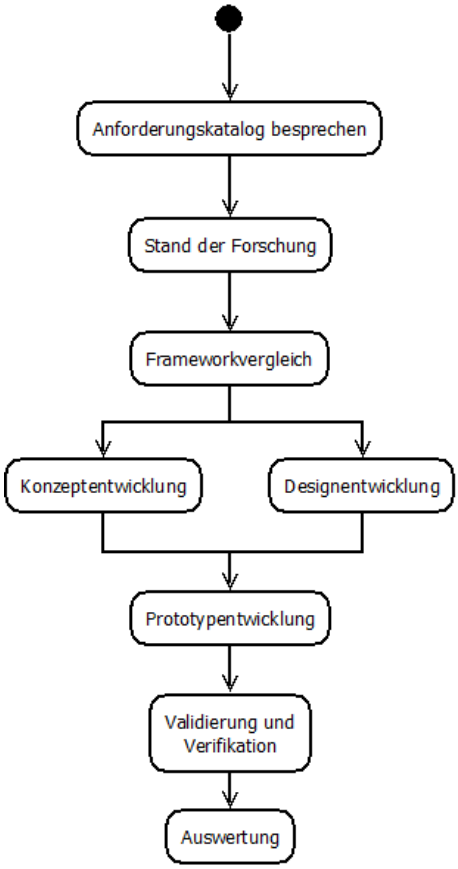

Abbildung 1.1.: Entwicklungsablauf

2. Grundlagen

Dieses Kapitel befasst sich mit den grundlegenden Inhalten, welche Bezug auf die Entwicklung, sowie die Einsatzzwecke der App nehmen. Dabei wird aufgezeigt, welche Umgebungen (Frameworks) für die Entwicklung einer mobilen Anwendung vorhanden sind, und abgewogen, inwieweit diese den Anforderungen entsprechen. Neben der Entwicklung wird zudem auch darauf eingegangen, in welchen möglichen Anwendungsfällen die App verwendet werden kann.

2.1. Begriffsdefinitionen

In dieser Masterarbeit werden mehrere Begrifflichkeiten verwendet, welche für das leichtere Verständnis hier beschrieben werden.

Unter einer **App** versteht man ein Programm, welches für Smartphones entwickelt wurde. Dabei stammt das Wort aus dem Englischen und ist die Kurzform für Application. So kann jedes Programm, unabhängig von der Plattform, auf der es läuft, theoretisch als App bezeichnet werden. Jedoch ist im deutschen Sprachgebrauch der Begriff hauptsächlich auf Programme auf dem Smartphone beschränkt.

Zur Entwicklung von Programmen werden häufig **Frameworks** mit eingesetzt. Dabei handelt es sich um ein Grundgerüst, welches für die Entwicklung von Programmen verwendet werden. Es umfasst bereits einige Funktionen, welche dann einfach in die Programmierung eingebunden werden können. Dies beschleunigt die Entwicklung und reduziert den Wartungsaufwand.

Der Begriff **Usability** bedeutet Bedienbarkeit. „Usability bezeichnet das Ausmaß, in dem ein Produkt, System oder Dienst durch bestimmte Benutzer in einem bestimmten Anwendungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.“ (29)

„**User Experience (UX)** erweitert den Begriff Usability um ästhetische und emotionale Faktoren wie eine ansprechende, „begehrenswerte” Gestaltung, Aspekte der Vertrauensbildung oder Spaß bei der Nutzung (Joy of use).
Dieser ganzheitliche Ansatz umfasst das gesamte Nutzungserlebnis, welches man bei der Verwendung eines Produktes erfährt. Die Nutzer sollen nicht nur schnell und reibungslos zum Ziel kommen, sondern - abhängig vom Anwendungsbereich - auch positive Gefühle wie Spaß oder Freude bei der Benutzung erleben.“ (29)

2.2. Verwendete Programme

Bei der Erstellung der Masterarbeit wurden unterschiedliche Programme verwendet. Diese umfassen die konzeptionelle Entwicklung mit Darstellungen. Für die Entwicklungen wurde eine Entwicklungsumgebung (IDE) verwendet, sowie ein Versionierungssystem. Dies wurde auch zur Dokumentation verwendet.

Adobe XD

Für die Entwicklung einer Oberfläche, sowie eines User Interface (UI) Prototypen wurde das Programm Adobe XD verwendet. Dies ist kostenfrei auf der Webseite der Firma Adobe verfügbar und bietet einfache Möglichkeiten für die UI-Entwicklung.

Als Alternative zu Adobe XD bieten sich an:

- Mockplus
- Sketch
- Figma

Dia

Für die Arbeit wurden Unified Modeling Language (UML) Diagramme erstellt. Dabei wurde das freie Programm *Dia* genutzt. Dies ist für alle gängigen Betriebssysteme erschienen und unterliegt der freien Lizenz GPL.

Neben Dia gibt es weitere alternative Programme:

- yEd
- Microsoft Visio
- Diagram Designer

Microsoft Visual Studio 2017 Enterprise

Für die Entwicklung der App wurde auf eine IDE von Microsoft zurückgegriffen. Das Programm Visual Studio 2017 integriert bereits alle notwendigen Tools, welche für das Xamarin Framework notwendig sind. Dieses wird in Kapitel 2.4.3 näher beschrieben.

Neben Visual Studio kann man auf folgende Alternativen zurückgreifen:

- Jetbrains Rider
- Atom

Jetbrains ReSharper

Um die Qualität des Quellcodes sicherzustellen wurde das Add-on ReSharper von Jatbrains für Visual Studio verwendet. Es überprüft den Quellcode auf eine konstante Einhaltung der Namenskonventionen bei der Benennung von Variablen. Aber auch Wiederholungen und Code Style.

Doxygen

Zur Dokumentation des Quellcodes und der App wurde auf Doxygen zurückgegriffen. Dies generiert aus dem Quellcode und Kommentaren eine Dokumentation, welche in unterschiedlichen Formaten verwendet werden kann. Für die Masterarbeit wurde auf eine Dokumentation in Webseiten-Format erstellt.

Für die Dokumentation von Quellcode bietet sich zudem an:

- Apiary
- Sphinx
- Natural Docs

Github

Da an dem Projekt mehrere Personen beteiligt sind, dient die Plattform *Github* als Plattform zum Dokumentaustausch und der gemeinsamen Dokumentation. Die Dokumente werden in dem Versionsverwaltungssystem Git verwaltet und können daher einfach gemeinsam verwendet und bearbeitet werden.

Neben Github wurde auch auf GitLab bei dem Austausch von Informationen mit den Studenten in Jena eingesetzt, da diese dort bereits ein Projekt eingerichtet hatten. Es wird in der Liste der alternativen Programme mit aufgeführt:

- GitLab
- BitBucket
- SourceForge

SmartGit

Um auf die Daten im Versionsverwaltungssystem Git zuzugreifen, wurde auf das Programm SmartGit, der Firma Syntevo zurückgegriffen. Dieses kann für freie Projekte kostenfrei verwendet werden. Es bietet eine einfache und nachvollziehbare Oberfläche, sowie unterstützt es den Anwender bei der Lösung von Konflikten zwischen Dateien.

Für die Versionsverwaltung können neben SmartGit auch folgende Programme verwendet werden:

- Microsoft Visual Studio
- GitKraken
- Git

Texmaker

Die Masterarbeit wurde in TeX geschrieben. Dabei wurde dafür das Programm Texmaker verwendet. Dieses ist wie auch Dia quelloffen und steht für mehrere Plattformen zur Verfügung.

Als Alternative zu Texmaker können auch mit folgenden Programmen Latex Dokumente erstellt und bearbeitet werden:

- Lyx
- TeXstudio
- TeXworks

Android SDK

Zum Testen und Debuggen während der Entwicklung wird auf dem Android Emulator aus dem Android SDK zugegriffen. Dieser steht kostenfrei bei Google zur Verfügung und kann auf jeder Plattform verwendet werden.

Da das Android SDK eine direkte Abhängigkeit für die Entwicklung der App ist, kann auf keine alternative Software zugegriffen werden.

2.3. Datenschutz

Das Ziel der App ist es, vom Anwender sowohl Mess- als auch Daten zu erfassen und zu verarbeiten. Dementsprechend ist es zwingend notwendig, der Europäischen Datenschutzverordnung (EU-DSGVO) nachzukommen. Aufgrund der Erfassung von Sensordaten, Lokalisations- und Zeitparametern ist es möglich Anwender zu identifizieren, was sie zu personenbezogenen Daten nach Art. 4 Abs. 1 EU-DSGVO, macht. Da die erfassten Daten weiterverarbeitet werden, tritt §6 Abs. 1 und 1a EU-DSGVO in Kraft, besagen, dass die Verarbeitung von Daten nur rechtmäßig ist, wenn der Anwender seine Einwilligung dazu gegeben hat, diese Daten zu verarbeiten. Entsprechend muss die App über eine Datenschutzerklärung verfügen, in welcher der Anwender dieser Zustimmen oder auch ablehnen kann. Der Betreiber der App muss nach Art. 7 Abs. 1 EU-DSGVO nachweisen können, dass der Anwender der Datenschutzerklärung zugestimmt hat.

In Art. 13 Abs. 1 und 2 EU-DSGVO wird der Inhalt der Datenschutzerklärung definiert. So muss der Betreiber der App, dem Anwender seinen Namen und die Kontaktdaten sowie gegebenenfalls die seines Vertreters mitteilen. Auch kann es möglich sein, die Kontaktdaten des Datenschutzbeauftragten zu nennen. Neben den Kontaktdaten muss auch der Zweck für die Erfassung und Verarbeitung dargelegt werden. Der zweite Absatz geht darauf ein, die Dauer der Speicherung anzugeben. Sollte dies nicht gegeben sein, so sollten die Kriterien für die Festlegung der Dauer genannt werden. Der Anwender hat zudem das Recht, Einblick in die gespeicherten Daten zu nehmen sowie die Daten löschen zu lassen.

Da der Anwendungsbereich zurzeit hauptsächlich in Deutschland ist, ist neben der EU-DSGVO zudem das Bundesdatenschutzgesetz (BDSG) zu beachten. Dabei ist nach §45 BDSG eine Weitergabe von Daten an öffentliche Stellen notwendig, wenn dadurch Straftaten vereitelt oder aufgeklärt werden können. Die nachfolgenden Paragraphen §46, §47 und §48 BDSG decken sich in den Inhalten mit der EU-DSGVO bezüglich der Begriffsklärung und Verarbeitung der erfassten Daten.

In dem Anhang A ist die komplette Datenschutzerklärung, wie sie auch in dem Prototypen implementiert wurde, zu finden. Für die Erstellung der Datenschutzerklärung wurde die offizielle Datenschutzerklärung des DLR in deutsch (26) und englischen (25) teilweise übernommen und auf den Anwendungsfall angepasst.

2.4. Frameworks

In diesem Kapitel wird beschrieben, welche Frameworks für die Entwicklung der App verwendet werden können. Es wird auf die Besonderheiten eingegangen sowie darauf, welches Framework sich am besten für die Entwicklung eignet.

Bei der Entscheidung für ein Framework spielen die Anforderungen aus dem Anforderungsdokument eine entscheidende Rolle. Unter den Anforderungen für die Entwicklung einer App stehen die Forderungen: plattformübergreifend, funktional ohne Internetzugang und keine Notwendigkeiten für das Einbinden von Drittanbieterprogrammen, welche Nutzerdaten sammeln.

2.4.1. Native

Jedes Smartphone verfügt über ein Betriebssystem, welches eine eigene Umgebung für die Entwicklung von Apps bereitstellt. Zu den drei am meisten verwendeten Betriebssystemen gehören Googles Android, Apples iOS und Microsofts Windows Phone.

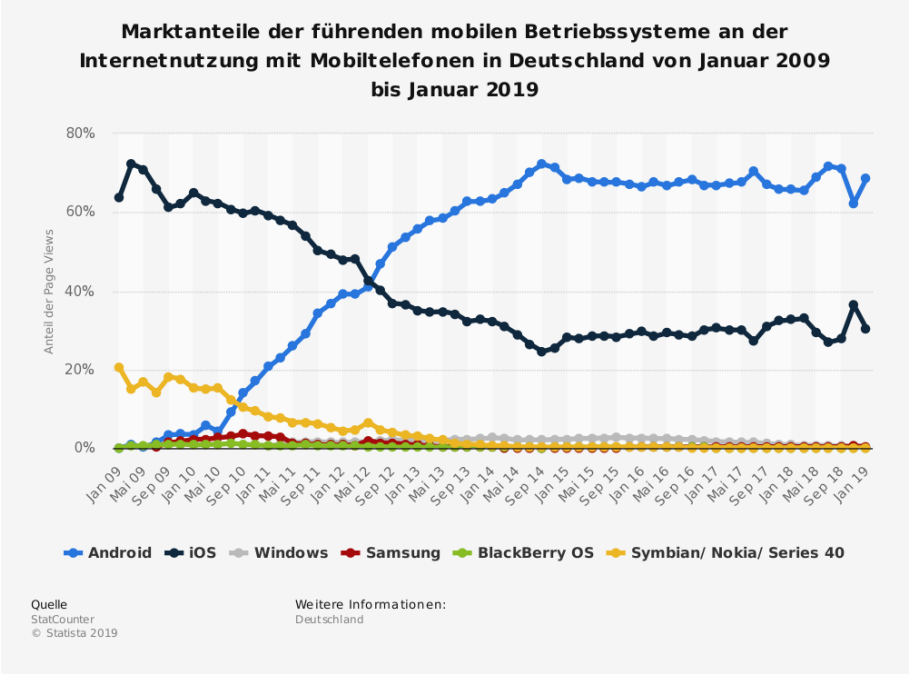

Abbildung 2.1.: Statistik Verbreitung mobiler Betriebssysteme (66)

In der Abbildung 2.1 wird die Marktentwicklung der 6 dominierenden mobilen Betriebssysteme miteinander verglichen. Dabei stechen die beiden Betriebssysteme von Google und Apple hervor. Zudem hat Microsoft angekündigt, den Support für sein Betriebssystem Windows Phone zum Dezember 2019 einzustellen (52).

Android

Das Betriebssystem Android wurde 2003 von Andi Rubin in der gleichnamigen Firma Android entwickelt. Diese wurde 2005 von Google aufgekauft, welche Android weiterentwickelt. Wie man bereits in der Grafik im Kapitel 2.4.1 gezeigt, ist es das derzeit verbreiteste Betriebssystem.

Apps für Android wurden hauptsächlich in Java entwickelt. Für rechenintensive Anwendungen kann C++ verwendet werden, welche über die JNI Schnittstelle mit Java kommuniziert. Für die Entwicklung stellt Google das Android Studio(30) kostenlos zur Verfügung. Dies ist eine abgewandelte integrierte Entwicklungsumgebung (IDE) von IntelliJ IDEA der Firma Jetbrains s.r.o.(45) aus Tschechien. Es ist spezialisiert auf die Entwicklung von Android Apps. Derzeit findet eine Transformation bei der Entwicklung von Android Apps von Java zu Kotlin statt. Diese Programmiersprache wurde 2011 von Jetbrains entwickelt (43).

iOS

Die Entwicklung von iOS begann 2005 bei Apple. Bis 2010 wurde das Betriebssystem auf dem Smartphone iPhone OS und auf dem Tablet iPad OS genannt. Danach wurden beide Betriebssysteme zu iOS umbenannt und zusammengefasst. (72)

Unter iOS wurden Apps bis 2014 hauptsächlich in Objective-C geschrieben. Doch bereits 2010 begann die Entwicklung der Sprache Swift bei Apple, welche dann 2014 erstmals den Entwicklern auf iOS vorgestellt wurde.(19) Für die Entwicklung von Apps für das Betriebssystem iOS ist ein Apple Computer mit macOS Voraussetzung, da der benötigte Compiler nur unter macOS verwendet werden kann. Man kann Apps in Xcode, der IDE von Apple, entwickeln. Diese enthält den benötigten Compiler. Um eigene Apps auf einem physischen Endgerät testen zu können, muss dieses freigeschaltet werden.

2.4.2. Web-App

Eine Web App sind Webseiten, die auf einem Smartphone eine Funktionalität anbieten, die dem Erscheinungsbild nach einer installierten App entsprechen. Sie beruhen auf Webtechnologien. Dabei können Webframeworks wie AngularJS, Angular CLI oder auch native Programmierung verwendet werden. Mittels JavaScript können Berechnungen auf den Clienten ausgeführt werden. Es ist zudem möglich, eine Webapp offline auf dem Smartphone verwenden zu können (11). Nach derzeitigen Stand gibt es aber noch Schwierigkeiten mit Sensoren und Schnittstellen innerhalb des Smartphones (13).

Damit eine mobile App für ein Smartphone entsteht, welche auch offline verwendet werden kann, muss die Webseite als Progressive Web-App (PWA) entwickelt sein. Dieser Standard wurde bei Google von Alex Russell 2015 vorgestellt (63) und beschreibt, wie mit Webtechnologien Apps entwickelt werden können, welche Zugriff auf Funktionen des Betriebssystems haben. Dabei kann die Webseite auf dem Smartphone sowie auch auf dem Desktop Computer verwendet werden.

AngularJS

AngularJS ist ein Webframework zur modularen Entwicklung von Webseiten mit den Webtechnologien HTML5, JavaScript und CSS3. Es wurde 2009 als OpenSource Framework von Google entwickelt. Die Entwicklung erfolgt nach dem Model-View-Controller-Prinzip (MVC). Dabei wird das Programm in Modell (Model), Ansicht (View) und Steuerungselement (Controller) aufgeteilt. Die Logik der Webseite wird in *Controllern* programmiert. Welche Objekte verwenden, die im *Model* beschrieben sind. Die Oberfläche, welche für den Anwender sichtbar ist, wird in der *View* definiert. Dieses Zusammenspiel wird MVC-Prinzip genannt. Als Weiterentwicklung von AngularJS ist Angular CLI entwickelt worden. Aus diesem Grund wird auch von Angular 1 und Angular 2 gesprochen.

Angular CLI

Als Nachfolger von AngularJS wurde Angular CLI entwickelt. Dieses wurde 2016 ebenfalls von Google entwickelt und nutzt im Gegensatz zu AngularJS TypeScript, anstatt von JavaScript. TypeScript ist kompatibel zu JavaScript ES6 und unterstützt zusätzlich statische Typisierungen und klassenbasierte objektorientierte Programmierung. Das Framework ist erweiterbar und unterstützt Pakete zur Entwicklung von Progressive Web-Apps.

Core

Bei der nativen Entwicklung wird auf ein Framework verzichtet und bei der Entwicklung der App nur auf JavaScript, HTML5 und CSS3 gesetzt.

2.4.3. Hybrid-App

Eine Kombination von Web-App und Nativer App stellt die Hybrid-App dar. Dabei werden unterschiedliche Ansätze verfolgt. So können wie bei Ionic, Webtechnologien auf einem lokalen Server auf dem Smartphone verwendet werden und im integrierten Browser dargestellt werden. Es ist aber auch möglich andere Programmiersprachen zu verwenden und die Programme so zu verpacken, dass diese eine App, für die jeweilige Plattform generieren. Alle Hybrid-Apps haben gemein, dass man einen gemeinsamen Quellcode für alle Plattformen hat und nur einzelne Anpassungen für das jeweilige System machen muss.

Xamarin

Das Xamarin Framework bietet die Möglichkeit in C# und XAML für Android und iOS zu entwickeln. Zusätzlich war es auch möglich für Windows Mobile zu entwickeln, welches jedoch aufgrund des Supportendes für Windows Mobile eingestellt wurde. Stattdessen kann das Programm nun als Universal Windows Plattform App auch auf allen kompatiblen Windows Plattformen betrieben werden. Es basiert auf der freien Mono-Plattform, welche als Open Source Variante von Microsofts .Net-Plattform 2001 entwickelt wurde. (54) Die Firma Xamarin wurde 2011 gegründet (37) mit dem Ziel ein plattformübergreifendes Framework auf Basis der Mono-Plattform zu entwickeln. Diese wurde 2016 von Microsoft übernommen (47). Die Entwicklung für Xamarin wurde direkt in Visual Studio, der IDE von Microsoft integriert und ist in allen Ausführungen der IDE vorhanden.

Bei der Entwicklung der Oberfläche werden native Elemente der jeweiligen Plattform verwendet, wodurch sich die entwickelten Apps nahtlos in das Designschema des mobilen Betriebssystems integrieren. Es ist aber auch möglich eigene Elemente zu entwickeln und somit der App ein eigenständiges Design zu geben.

Flutter

Im Gegensatz zu Xamarin wird Flutter in der Programmiersprache Dart geschrieben. Das Framework wurde 2015 das erste Mal der Öffentlichkeit präsentiert (9). Es wurde, wie auch die Programmiersprache, von Google entwickelt. Ziel war es, in kurzer Zeit graphisch ansprechende Apps zu entwickeln, welche in nativer Geschwindigkeit auf dem Smartphone laufen und wenig Speicher benötigen.

Ionic

Mit dem Ionic Framework wird im Gegensatz zu Xamarin und Flutter ein anderes Konzept verfolgt. So wird eine entwickelte App nicht wie eine native App auf dem Smartphone verwendet, sondern wie eine Webseite. Daher wird beim Starten einer Ionic App ein Browserfenster geöffnet, welches das Programm startet. Dieses wird wie eine Webseite mit Webtechnologien in Javascript entwickelt. Das Framework entstand 2013 von der Firma Drifty Co. und basierte auf AngularJS, sowie Apache Cordova. Mit fortschreitender Entwicklung konzentrierten sich die Entwickler darauf, das Framework auf Angular umzustellen.

React Native

Dieses Framework wurde 2015 von Facebook entwickelt (8) und unterstützt, wie auch Xamarin, die Plattformen Android, iOS und UWP. Die Sprache verwendet für die Entwicklung hauptsächlich JavaScript und ermöglicht den Entwicklern nativen Quellcode zusätzlich auszuführen. Das Framework wird bereits von Unternehmen wie Facebook und Instagram verwendet, jedoch existiert bisher noch keine stabile Version, sodass Updates am Framework zu großem Aufwand in der Wartung führen können.

2.4.4. Vergleich der Frameworks

Für die Entwicklung stehen vielfältige Frameworks zur Verfügung. Diese sind anhand ihrer Eigenschaften für unterschiedliche Einsatzzwecke geeignet. Die App soll den Anforderungen im Anforderungskatalog entsprechen. Dementsprechend wurden die Vergleichskriterien gewählt. Verglichen wird die native Entwicklung unter Android und iOS, Webapps wie Angular, AngularJS oder native Entwicklung und die Hybrid-Apps Xamarin, Flutter, Ionic und React. Dabei werden diese Frameworks wie folgt verglichen. Sie müssen auf Android und iOS Smartphones funktionsfähig sein. Dies schließt die nativen Frameworks aus, da diese nur auf dem jeweiligen System lauffähig sind. Man müsste somit die App zweimal entwickeln und hätte den doppelten Aufwand. Dies spiegelt sich auch im Wartungsaufwand wieder, während die Webapps und Hybrid-Apps auf beiden Plattformen mit der selben Codebasis laufen. Die Web-Apps haben eine schnelle Entwicklungszeit, wodurch mehr Anpassungen an die Neuerungen, zu einer intensivere Wartung führen. Jedoch bieten sie auch die Möglichkeit, eine App über einen Browser auf einem beliebigen Gerät ausführen zu können. Im Bereich Datenschutz wird berücksichtigt, inwieweit Daten an Dritte weitergegeben werden. Dabei fällt negativ auf, dass für die Einbindung von Plugins bei den Hybriden Frameworks wie Ionic meist externe Plugins notwendig sind. Diese Abhängigkeit stellt eine Gefahr für Datenmissbrauch dar. Entsprechend sind die nativen Sprachen sowie Xamarin und Flutter bevorzugt. Bezüglich der Sensorverfügbarkeit werden Frameworks mit integrierter Sensorunterstützung bevorzugt. Ein weiteres wichtiges Kriterium ist die Funktionalität der App ohne Internet. Alle Frameworks bieten die Möglichkeit, auch Offline verwendet zu werden. Im Bereich Zukunftssicherheit wird analysiert, inwieweit sich das Framework für eine weitere Entwicklung eignet. AngularJS wird als Vorgänger von Angular CLI weiterhin gepflegt, jedoch nur als Legacy Projekt und auch die Webseite von AngularJS verlinkt auf den Nachfolger Angular CLI. Durch die Entwicklung von Progressive Web-Apps wird Ionic nicht mehr benötigt. Dieses bringt seinen eigenen Browser mit, welches nicht mehr notwendig ist. React Native wird seit mehreren Jahren entwickelt, jedoch wurde noch keine stabile Version veröffentlicht. Dadurch ist die Entwicklung in Zukunft unsicher.

Tabelle 2.1.: Vergleich der Frameworks Nativ und Web-Apps

	Nativ		Web-Apps		
	Android	iOS	AngularJS	Angular CLI	Nativ
Android-kompatibilität	+	-	+	+	+
iOS-kompatibilität	-	+	+	+	+
Wartungsaufwand	-	-	-	+	+
Datenschutz	+	+	-	+	+
Sensorverfügbarkeit	+	+	-	+	+
Offline	+	+	+	+	+
Zukunftssicherheit	+	+	-	+	+

Tabelle 2.2.: Vergleich der Frameworks Hybrid-Apps

	Hybrid-Apps			
	Xamarin	Flutter	Ionic	React Native
Android-kompatibilität	+	-	+	+
iOS-kompatibilität	-	+	+	+
Wartungsaufwand	-	-	-	+
Datenschutz	+	+	-	+
Sensorverfügbarkeit	+	+	-	+
Offline	+	+	+	+
Zukunftssicherheit	+	+	-	-

Nach hinreichender Auseinandersetzung mit den unterschiedlichen Frameworks wurde bei der Entwicklung auf Xamarin gesetzt. Dabei spielt die Verfügbarkeit der Sensoren eine entscheidende Rolle bei der Auswahl. Diese ist bei Web-Apps derzeit nur teilweise gegeben, wodurch eine einheitliche Verwendung der App auf mehreren Geräten nicht gewährleistet werden kann. Zudem werden Web-

Apps, welche zusätzlich als Progressive Web-App entwickelt werden, noch nicht vollständig unter Apple iOS unterstützt, was sich zudem negativ auswirkt. Dies wird sich jedoch mit kommenden iOS Versionen verbessern.

2.5. Anwendungsfälle

Die entwickelte App dient der Unterstützung von Projekten, die auf Bürgerbeteiligung angewiesen sind. Dabei ist es notwendig, die App möglichst nutzerfreundlich und modular zu entwickeln, sodass die Anwender der App diese als Hilfsmittel, statt als Hindernis ansehen. Die Modularität der App dient der einfachen Erweiterbar- und Wartbarkeit der einzelnen Projekte. In diesem Abschnitt werden mögliche Anwendungsfälle betrachtet, in denen die App eingesetzt werden kann.

2.5.1. Forschungsprojekt DEMMIN

Das Forschungsprojekt der Durable Environmental Multidisciplinary Monitoring Information Network, kurz *DEMMIN*, dient als Testfeld Kalibration und Validation von Satelliten. (17) Es wird vom Deutschen Zentrum für Luft- und Raumfahrt (DLR) in Neustrelitz in Kooperation mit Landwirten aus dem Raum Demmin betrieben. (16)

Abbildung 2.2.: Logo Forschungsprojekt DEMMIN ®

Standort

Das Testfeld liegt ungefähr 220 km nördlich von Berlin nahe der Stadt Demmin in Mecklenburg-Vorpommern. Dieser landwirtschaftlich intensiv genutzte Standort erstreckt sich von 54°2'54.29" Nord, 12°52'17.98" Ost bis 53°45'40.42" Nord, 13°27'49.45" Ost. (16)

Das von den mit dem DLR kooperierenden Landwirten bewirtschaftete Gebiet (ca. 30.000 ha) ist für wissenschaftliche Anwendungen der Fernerkundung besonders geeignet, da die Standortheterogenität (z.B. Landschaft, Bodendecke, Hydrologie) bei einer durchschnittlichen Schlaggröße von ca. 80 ha sehr hoch ist. Die angebauten Hauptfruchtarten sind Wintergetreide (z.B. Winterweizen, -gerste, -roggen), die fast 60% der Felder bedecken. Die Fläche für Mais, Zuckerrübe und Kartoffeln beträgt etwa 13%. (16)

Auf den Feldern des Testgeländes befinden sich fest installierte stationäre Sensoren, welche Umgebungsdaten erfassen und an eine Empfangsstation weiterleiten.

Feldkampagne

Bei einer Feldkampagne vom 16.04.2019 bis zum 18.04.2019 kamen mehrere studentische Hilfskräfte, sowie Mitarbeiter und Doktoranden des DLR und der Universität Jena in Kruckow zu einer gemeinsamen Erfassung von Messdaten zusammen. Bei dieser Feldkampagne wurden auf zwei Feldern, auf denen Getreide angebaut war, Messstellen eingerichtet und die Pflanzen auf ihr Wachstum untersucht. Durch diese Feldkampagne konnten die Anforderungen an die App konkretisiert und weitere Entwicklungsschritte diskutiert werden.

Abbildung 2.3.: Feldkampagne (Quelle: Google Maps, verändert am 23.04.2019)

Verwendung der App

Die App wird auf einem freien Feld, wie in Abbildung 2.3 gezeigt, verwendet. Dadurch ist es bei starker Sonneneinstrahlung notwendig, die App so zu gestalten, dass sie gut lesbar ist. Dies hat Auswirkungen auf das Design der App, wie z.B. die Farbwahl, wie in Kapitel 5.4.1 beschrieben.

Bei der Datenerfassung werden Fotos, sowie Standortdaten mit Uhrzeit aufgezeichnet. Zu den jeweiligen Bildern werden unterschiedliche Formulare ausgefüllt, welche dann dem Bild zugeordnet werden müssen. Zudem werden unterschiedliche Geräte zur Erfassung von Messwerten verwendet. Die App bietet durch passende Formulare die Möglichkeit, diese Daten zu speichern und zu verwalten.

2.5.2. Erfassung von Stadtverbesserungen

Ein weiterer möglicher Anwendungsfall kann die öffentliche Erfassung von Daten sein. Dies kann auf verschiedene Art hilfreich für die zuständigen Ämter erfolgen. Dabei könnte jeder Bürger, mittels Foto und erfasster Standortdaten interessierende Aspekte, wie z.B. Schlaglöcher erfassen und an das zuständige Amt senden. Ein weiterer Anwendungsfälle könnten das Melden von illegalen Mülldeponien, Verunreinigungen oder fehlenden Bushaltestellen sein. Dabei kann der Bürger aktiv an der Mitgestaltung seiner Stadt beteiligt werden.

3. Stand der Forschung

Dieses Kapitel befasst sich mit derzeit verfügbaren Entwicklungen bei der Erfassung von Daten mit dem Smartphone. Dabei stellen die meisten Projekte einen Server, sowie eine App für das Smartphone zur Verfügung.

3.1. Spezifische Frameworks zur Methodenentwicklung

Neben der Entwicklung eigener Apps gibt es auch Frameworks, bei denen Apps generiert werden können und kaum eigene Entwicklung notwendig ist. Dieser Abschnitt bezieht sich auf das Paper *Software solutions for form-based collection of data and the semantic enrichment of form data* von Markus Steinberg (67).

Open Data Kit

Das Open Data Kit (ODK) ist ein Framework, welches quelloffen von der gleichnamigen Gemeinschaft entwickelt wird. Diese stellen zwei Versionen des Frameworks Entwicklern zur Verfügung. Dabei richtet sich die Version ODK allgemein an alle Anwender mit vereinfachten Dialogen. Es enthält auch ein Großteil aller notwendigen Programme. Die Version ODK-X (früher ODK 2) richtet sich dagegen eher an Entwickler, welche das Framework stark anpassen möchten. (59)

In der Version ODK sind folgende Programme enthalten:

- *ODK Collect*
  Dies ist eine Android App, welche die Anwender auf ihrem Smartphone ausführen zur Datenerfassung. (60)
- *ODK Aggregate*
  Server auf dem die Daten gespeichert und analysiert werden. (60)
- *ODK XLSForm*
  Microsoft Excel basierter Formulardesigner . (60)
- *ODK Build*
  XForm basierter Formulardesigner. (60)
- *ODK Briefcase*
  Programm zum übertragen der Daten von ODK Collect zu ODK Aggregate. (60)
- *ODK Central*
  Server mit REST-Schnittstelle. (60)

Die erweiterbare Version ODK-X enthält eigene Anwendungen und erweitert das ODK nicht. :

- *ODK-X Application Designer*
  Erstellung einer eigenen App zur Datenerfassung mit Formularen. (61)
- *ODK-X Services*
  Programm zur Verwaltung und Verwendung einer Datenbank. Synchronisiert alle Services zwischen den ODK-X Anwendungen und einem Cloud Endpunkt. (61)

- *ODK-X Survey*
  Datenerfassungs-App bei der die Oberfläche mit HTML, Javascript und CSS entwickelt wird. (61)
- *ODK-X Tables*
  App zur Visualisierung und Überprüfung erfasster Daten. (61)
- *ODK-X Suitcase*
  Programm zur Synchronisierung der Daten mit einem Cloud Endpunkt. (61)
- *ODK Cloud Endpoints*
  Cloud Endpunkt Server um Daten zu verwalten und mit mobilen Endgeräten zu synchronisieren. (61)

EpiCollect5

Dieses Framework wurde am Imperial College London entwickelt. (40) Es besteht aus der Webseite, auf der Umfragen erstellt werden können und generiert Webapplikationen für Android Smartphones, welche die Daten online wie auch offline erfassen können. Die gesammelten Daten werden, nachdem sie erfasst wurden, auf den Server von EpiCollect5 übertragen und dort ausgewertet. Das Framework ist ein Nachfolger von EpiCollect und EpiCollect+, welche auch an dem Imperial College London entwickelt wurden. (39)

KoBo Toolbox

Die KoBo Toolbox (35) ist ein OpenSource Framework, welches EpiCollect stark ähnelt. Mittels eines Servers werden Umfragen generiert. Über die App KoBoCollect aus dem Google Play Store werden die Daten erfasst. Entwickelt wurde dieses Framework von der Harvard Humanitarian Initiative (HHI).

Ohmage

Ein weiteres OpenSource Framework, neben ODK ist Ohmage (4). Es ermöglicht zusätzlich zur Erfassung von Messdaten, diese auch zu analysieren und visualisieren. Entwickelt wird es von der University of California, Los Angeles und der Cornell Tech School. Dabei wird dem Anwender kein eigener Server zur Verfügung gestellt. Die Anwender müssen statt dessen, einen von Server, der Cornell Tech School verwenden.

SurveyCTO

Das SurveyCTO ist eine kommerziell verfügbares Datenerfassungsplattform, welche von Dobility, Inc. entwickelt wurde (27). Die Plattform wird für ein monatliches Abo dem Kunden zur Verfügung gestellt. Zur Erfassung wird auch wie bei den vorherigen Plattformen eine App für Android, sowie eine Webseite zur Verfügung gestellt, über die die Anwender ihre Messdaten eingeben können. Die erfassten Daten können in Echtzeit, sofern Internet vorhanden ist, überwacht werden.

Magpi

Eine weitere kommerzielle Lösung stellt Magpi dar, welches von der gleichnamigen Firma entwickelt und bereitgestellt wird (3). Es bietet neben einem kostenpflichtigen Dienst auch einen im Funktionsumfang eingeschränkten kostenlosen Dienst an.

COBWEB

Das Citizen Observatory Web (COBWEB) war ein europäisches Projekt zur Datengewinnung. Es wurde von der University of Edinburgh geleitet und war auf 4 Jahre ausgelegt. Anwender konnten vom 1. November 2012 bis zum 31. Oktober 2016 teilnehmen, wobei das Ziel war, möglichst viele Umweltdaten zu erfassen. (36)

Vergleich der genannten Frameworks

Bei den genannten Frameworks stechen das Open Data Kit, KoBo, sowie Ohmage positiv hervor. Es wurde sich bei der Entwicklung auf das Open Data Kit geeinigt, da dieses bereits am Institut für Bürgerwissenschaften am DLR Jena in vorherige Projekte eingesetzt wurde.

Tabelle 3.1.: Vergleich der Frameworks zur Datenerfassung

	ODK	EpiCollect5	KoBo	Ohmage	SurveyCTO	MagPi	COBWEB
OpenSource	+	-	+	+	-	-	+
Kosten	+	+	+	+	+[1]/-	+[2]/-	+
Unabhängigkeit	+	-	+	+	-	-	+
Verfügbarkeit	+	+	+	+	+	+	-
Dokumentation	+	+	+	+	+	+	-

3.2. Sensorerfassung

phyphox

Im Rahmen der Aufzeichnung von Sensordaten aus dem Smartphone wurde an dem 2. Physikalischen Institut der RWTH Aachen University die App phyphox entwickelt(6). Ziel der App ist es, mit Hilfe der Sensoren im Smartphone Experimente durchzuführen und zu dokumentieren.

Dabei bietet die App die Möglichkeit, eigene Experimente mittels eines XML-Schemas zu definieren und aufzuzeichnen.

Sensors Multitool

Ein weiteres bereits vorhandenes Tool zum Auslesen von Sensordaten ist die App Sensors Multitool von Wered Software. Es lassen sich bei der App keine Experimente definieren, jedoch bietet sie die Möglichkeit an, die gesammelten Daten aufzuzeichnen. (71)

3.3. Studien zur Datenerfassung via Smartphone

Es gibt bereits mehrere Studien zum Thema Datenerfassung via Smartphone. Dabei hat das Smartphone die Erfassung mit Stift und Papier ersetzt. Bereits 2012 wurde in China ein Versuch mit Müttern durchgeführt. Dabei wurden 120 Müttern zu ihren Kindern von Interviewern im Rahmen der Maternal, Newborn, and Child Health (MNCH) Household Study befragt. Nach dem Zufallsprinzip wurden diese in zwei Gruppen eingeteilt, wobei eine Gruppe Fragebögen mit Stift und Papier ausfüllten und die andere Gruppe eine App auf dem Smartphone verwendete. Das Ergebnis der Studie ist eine Reduzierung von Aufzeichnungs- und Erfassungsfehlern durch das Smartphone. Zudem benötigte die Erfassung mit dem Smartphone keine zusätzliche Zeit. Die Kosten waren für den geringen Umfang der Studienteilnehmer höher, als mit Stift und Papier. Die Autoren der Studie gehen aber davon aus, dass diese mit zunehmender Teilnehmerzahl reduzieren werden. (73)

[1] Community Edition
[2] siehe Fußnote 1

3.4. Usability und User Experience (UX)

Neben den Programmen, welche es bereits gibt, so existieren auch viele wesentliche Studien im Bereich Usabitilty und User Experience. Eine wichtige Erkenntnis ist, dass sich zwar die Oberflächen und Menüführung stetig für den Anwender ändert, es aber Erkenntnisse aus verschiedenen Bereichen gibt, die beständig gelten. Die Grafik 3.1 verdeutlicht welche Quellen für den Erkenntnisgewinn im Bereich User Experience zum Tragen kommen.

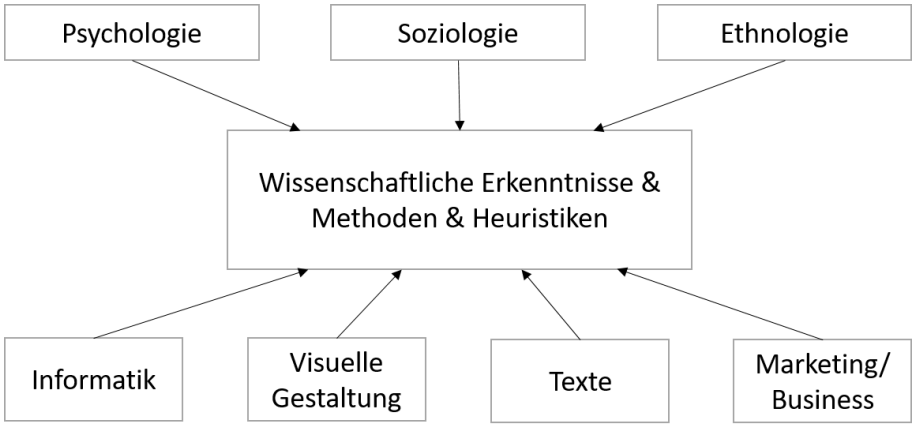

Abbildung 3.1.: Quellen für Erkenntnisse im Bereich UX. Nach (44) S.38

Die Nielsen Norman Group erstellt Studien zum Thema User Experience. Unter anderem werden Trainings und Consultings angeboten, aber auch wöchentliche Reports in denen Artikel zu jeweiligen Themen im Bereich User Experience.

Mobile Navigation

In der Studie *Mobile First Is NOT Mobile Only* von der Nielsen Norman Group (18) wird darauf eingegangen, dass bei der Entwicklung von plattformunabhängigen Anwendungen eine dem Gerät angepasste Darstellung entwickelt werden sollte. Eine für Smartphones optimierte Navigation befindet sich in vielen Fällen hinter einem verdeckten Button. Dadurch ist es möglich, mehr Inhalte auf dem Smartphone darzustellen. Auf größeren Displays, wie Tablets und Laptops, ist dieser Vorteil nicht mehr gegeben. Es sind zusätzliche Wege mit der Maus notwendig und die Übersicht der Webseite ist durch die größere Darstellungsfläche verschlechtert.

4. Konzept

Dieses Kapitel befasst sich mit der spezifischen Entwicklung der App. Dies umfasst die Architektur, Entwicklung einzelner Prototypen, sowie den Test der App.

4.1. Projektaufbau

Das Projekt wurde mit Microsoft Visual Studio 2017 erstellt und besteht aus 4 Unterprojekten. Dabei handelt es sich um einen gemeinsamen Quellcode, Anpassungen an Android und iOS, sowie ein Unterprojekt für Unit Tests.

4.1.1. DLR_Data_App

In diesem Unterprojekt wird der gemeinsam verwendete Quellcode für Android und iOS entwickelt. Dabei handelt es sich um die Geschäftslogik, die Modelle und Services, welche in der App verwendet werden.

- *Data*
  Der Ordner Data umfasst ein Beispielprojekt. Dabei handelt es sich um die Feldkampagne mit mehreren Formularen und einer Projektbeschreibung.
- *Documentation*
  Für die Dokumentation werden zusätzlich zu den Kommentaren im Quellcode, einzelne Seiten für Beschreibungen und Erklärungen geschrieben. Aus den in Markdown geschriebenen Seiten erzeugt Doxygen in der Onlinedokumentation zusätzliche Seiten.
- *Localizations*
  Die Sprachen werden in XML-Dateien mit Schlüssel-Werte-Paaren abgespeichert. Für jede Sprache wird dabei eine eigene Datei angelegt.
- *Models*
  Die Models umfassen Definitionen von Objekten, welche dann im Quellcode verwendet werden können. Es ist Bestandteil des MVVM-Prinzips.
- *Services*
  Zu den Services gehören wiederkehrende Methoden, wie z.B. Methoden für die Verwaltung der lokalen Datenbank.
- *ViewModels*
  Die ViewModels sind Teil des MVVM-Prinzips und umfassen die Logik für die jeweilige Oberfläche.
- *Views*
  Die Definitionen für die Darstellung der Oberflächen wird in den Views gespeichert.

4.1.2. DLR_Data_App.Android

Dieses Unterprojekt enthält alle Android-spezifischen Änderungen und Einstellungen, welche bei der Entwicklung notwendig sind. Dabei handelt es sich unter anderem um Dateipfade, Rechtemanagement und Bilder.

- *Assets*
- *Resources*

4.1.3. DLR_Data_App.iOS

Analog zu dem Unterprojekt für Android, wurde dieses Unterprojekt für iOS angelegt. Es beinhaltet spezifische Einstellungen für iOS und jeweilige Anpassungen von Pfaden, Rechten und Bildern.

- *Resources*

4.1.4. DLR_Data_App_Test

Für die Unit Tests, welche automatisch durchgeführt werden können, wurde ein weiteres Unterprojekt angelegt. Dies greift auf den gemeinsamen Quellcode von dem Unterprojekt *DLR_Data_App* zu und testet die dortigen Funktionen.

- *Login*
- *Services*

4.2. Anforderungen

Dieses Kapitel beschreibt alle Anforderungen, die an die App gestellt werden. Im Rahmen der Masterarbeit wird, aufgrund des Umfangs und Zeiteinteilung, eine Untermenge an Anforderungen bearbeitet. Dieses Kapitel bezieht sich auf das Dokument (69). Damit eine einheitliche Übersicht über die Anforderungen erhalten bleibt, wurde die Nummerierung aus dem Dokument (69) übernommen.

4.2.1. Nicht funktionale Anforderungen

4.2.1.1. Generelle Anforderungen

Zu den grundlegenden Anforderungen gehört die Definition, auf welchen Endgeräten mit welchen Betriebssystemen die App laufen soll. Da es hauptsächlich auf mobilen Endgeräten lauffähig sein soll, wurde sich auf Smartphones und Tablets reduziert. Laut der Statistik aus Kapitel 2 sind auf einem Großteil der Geräte Android und iOS installiert. Die Zuverlässigkeit der App muss durch korrekte Eingaben gewährleistet sein, auch, wenn kein Internet vorhanden ist. Dabei spielt die Zuverlässigkeit vor allem bei der Profilerstellung und -verwaltung eine übergeordnete Rolle. Eine einfache Erweiterung der App um z.B. weitere Sensoren muss zudem bereits im Konzept angedacht werden, da so der Entwicklungs- und Wartungsaufwand reduziert werden kann. Zudem ist die Verwendung von Standards bei der Entwicklung in Teams sinnvoll. Daher muss die App die folgenden Anforderungen erfüllen:

- *URQ-GEN-IT-010* Endgeräte
- *URQ-GEN-IT-020* Betriebssystem
- *URQ-GEN-IT-030* Internetzugriff
- *URQ-GEN-IT-040* Zuverlässigkeit
- *URQ-GEN-IT-050* Erweiterbarkeit
- *URQ-GEN-IT-070* Sensorikzugriff
- *URQ-GEN-IT-080* Einsatz von Standards

Es wurde die Entwicklung einer Webapplikation in Kapitel 2.4.4 ausgeschlossen. Zudem wurde die Anforderung an den Serverkontakt nicht realisiert, da diese nicht für die grundlegende Lauffähigkeit der App notwendig ist:

- *URQ-GEN-IT-060* Serverkontakt

4.2.1.2. Anforderungen an die Nutzerfreundlichkeit

Die Nutzerfreundlichkeit ist ein wichtiger Aspekt bei der Entwicklung der App. Entsprechend befasst sich das Kapitel 5 intensiv mit den unterschiedlichen Faktoren in der Nutzerfreundlichkeit. Dabei wurde bei der Entwicklung konsequent auf die folgenden Eigenschaften geachtet:

- *URQ-GEN-NFR-010* Allgemein
- *URQ-GEN-NFR-020* intuitiv
- *URQ-GEN-NFR-030* effektiv
- *URQ-GEN-NFR-040* Antwortverhalten
- *URQ-GEN-NFR-050* heart-beat

- *URQ-GEN-NFR-060* Bildschirm
- *URQ-GEN-NFR-070* Datenverwaltung

Diese werden durch Heuristiken des User Interfaces von Jacob Nielsen in dem Kapitel 8 überprüft. Aufgrund der noch ausstehenden Implementation der Datenübertragung, wurde der folgende Punkt nicht implementiert:

- *URQ-GEN-NFR-080* Datenübertragung

4.2.1.3. Anforderungen an die Daten-Sicherheit

Bei der Entwicklung der Anwendung ist die Datensicherheit von großer Bedeutung. Es muss bereits im Konzept berücksichtigt werden, nur die Module von externen Anbietern einzubinden, welche keine Nutzerdaten bei der Verwendung analysieren. Dies lässt sich durch die Lizenzen der Plugins überprüfen, sowie auch über die Datenschutzerklärungen der verwendeten Software. Da dies im Nachhinein nur sehr schwer verändert werden kann, ist bereits beim Konzept darauf zu achten. Aus diesem Grund wurden die folgenden Anforderungen berücksichtigt:

- *URQ-GEN-DAS-010* Zugriff
- *URQ-GEN-DAS-030* Externe Analyse

Eine weitere wichtige Anforderung ist die sichere Kommunikation mit dem Server. Dies erfolgt durch eine verschlüsselte Verbindung. Es handelt sich dabei jedoch um keine Anforderung, welche die grundlegende Funktionalität der App beeinflusst. Daher wurde diese Anforderung nicht berücksichtigt:

- *URQ-GEN-DAS-020* Verschlüsselung

4.2.2. Funktionale Anforderungen

4.2.2.1. Authentifizierung

Aus der Entwicklung der Benutzerprofile ergibt sich die Notwendigkeit der Authentifizierung. Da ein Benutzerprofil nur verwendet werden kann, wenn man es erstellen und sich damit anmelden kann, war die Implementierung dieser Anforderung notwendig. Daher wurden die folgenden Anforderungen implementiert:

- *URQ-FUN-AUT-010* Registrierung
- *URQ-FUN-AUT-020* Anmeldung

4.2.2.2. Benutzerkonto

Bei der Verwendung der App ist ein Profil zwingend notwendig zur Identifikation und entsprechenden Einschätzung des Anwenders durch Profiling. Daher wurde eine Profilverwaltung implementiert. Damit wurden die folgenden Anforderungen implementiert:

- *URQ-FUN-BKT-010* Anzeige eigenen BN-profils
- *URQ-FUN-BKT-020* Änderung eigenen BN-profils
- *URQ-FUN-BKT-030* Allg. Einstellungen Zugriffsrechte
- *URQ-FUN-BKT-040* Abmeldung

Es wurde bei der Implementation auf die Eingabe einer E-Mail-Adresse, sowie einer Sicherheitsabfrage verzichtet. Dies ist mit der möglicherweise fehlenden Internetverbindung begründet. Der Account kann lokal von einem anderen Account aus verändert werden. Dabei wird das Profil nur lokal erstellt und verwaltet.

4.2.2.3. Individuelle App-Funktionen

Bei den individuellen App-Funktionen, handelt es sich um Eigenschaften, welche in den einzelnen Formularen verwendet werden können. Diese wurden an das Open Data Kit Build ausgelagert. Dort können die Formulare erstellt werden. Der Prototyp verfügt jedoch nicht über den vollen Umfang aller Module. Es fehlen Lineale, welche an der Bildschirmkante dargestellt werden, sowie Foto- und Karten-Funktionen.

- *URQ-FUN-IAF-010* Anzeige Modulkatalog
- *URQ-FUN-IAF-020* Modulauswahl
- *URQ-FUN-IAF-030* Anzeige integrierter Module

4.2.2.4. Prüfmodule

Die App verwendet Sensoren, zu denen auch die Geolokation gehört. Entsprechend wurden Prüfroutinen eingebaut, welche die Verfügbarkeit der Sensoren überprüfen. Entsprechend wurden die folgenden Anforderungen implementiert:

- *URQ-FUN-PRM-020* Erfassen Geolokation
- *URQ-FUN-PRM-030* Verfügbarkeit Sensorik

Die Verfügbarkeit des Internets wird nicht überprüft, da dies beim Downloaden des Projekts Aufgabe des Systembrowsers ist, mit dem das Archiv heruntergeladen wird. Aufgrund der noch offenen Implementation bei der Kommunikation mit ODK Central wurde keine Internetverfügbarkeit implementiert:

- *URQ-FUN-PRM-010* Verfügbarkeit Internet

4.2.2.5. Basismodule

Die Basismodule wurden alle implementiert. Dabei basieren alle Module auf Systemkomponenten, von denen die Daten ausgelesen wurden.

- *URQ-FUN-BAM-010* Zeitzone
- *URQ-FUN-BAM-020* Datensatzart
- *URQ-FUN-BAM-030* Bezugszeit
- *URQ-FUN-BAM-040* Lokalisierung
- *URQ-FUN-BAM-050* Ausrichtung

4.2.2.6. Profiling

Die Anforderungen des Profilings wurden nicht betrachtet, da dies die Aufgabenstellung der Studenten an der Friedrich-Schiller-Universität in Jena war. Aus diesem Grund wurden alle Anforderungen aus diesem Abschnitt nicht betrachtet:

- *URQ-FUN-PRF-010* Allgemein
- *URQ-FUN-PRF-020* Selbstreflexion
- *URQ-FUN-PRF-030* Sorten erkennen
- *URQ-FUN-PRF-040* Sorten & Entwicklungsstadien erkennen
- *URQ-FUN-PRF-050* Bedeckungsgrad & Anteil grüner Pflanzenbestandteile schätzen

4.2.2.7. Feldkampagne

Da die App so aufgebaut ist, dass mehrere Projekte hinzugefügt werden können, wurde dieser Teil nicht direkt in die App integriert. Stattdessen wurde das Projekt teilweise als Beispielprojekt implementiert. Dabei enthält das Beispielprojekt die folgenden Module:

- *URQ-FUN-FKP-010* Feldbezeichnung
- *URQ-FUN-FKP-020* Vorfrucht
- *URQ-FUN-FKP-030* Anbaufrucht
- *URQ-FUN-FKP-040* Aussaat
- *URQ-FUN-FKP-050* Pflanzenreihenausrichtung
- *URQ-FUN-FKP-060* Pflanzenreihenabstand
- *URQ-FUN-FKP-070* Pflanzendichte
- *URQ-FUN-FKP-080* Phänologische Entwicklung
- *URQ-FUN-FKP-090* Schäden am Bestand
- *URQ-FUN-FKP-100* Ernte
- *URQ-FUN-FKP-110* Bewässerung
- *URQ-FUN-FKP-120* Düngung

Die Module wurden in Absprache mit dem DLR Jena ausgewählt und implementiert. Dabei wurde die ODK Build Webapp zur Erstellung der Formulare verwendet. Die folgenden Module sind als Formular noch nicht erstellt worden:

- *URQ-FUN-FKP-130* Pestizide/Herbizide
- *URQ-FUN-FKP-910* Feststellung Bodenvegetation
- *URQ-FUN-FKP-920* Beschreibung Bodenvegetation
- *URQ-FUN-FKP-930* Messung der Höhe der Vegetation ausgehend vom Bodenniveau
- *URQ-FUN-FKP-940* Phänologische Entwicklung Farne

4.3. Übersicht der Funktionalität

Aus den Anforderungen in Kapitel 4.2 ergeben sich die Funktionalitäten, die die App erfüllen muss. Neben den Anforderungen an die Plattform und das Framework gibt es auch Anforderungen an die Funktionalität. So muss die App die folgenden Funktionen enthalten:

- Profilverwaltung
- Projektverwaltung
- Sensorverwaltung
- Einschätzung von Anwendermessungen
- Messungen erfassen

Aus den Funktionen lassen sich die notwendigen Oberflächen für den Anwender ableiten. Zudem müssen Oberflächen für Lizenzen, Datenschutz und ein Impressum festgelegt werden. Die App wird aus folgenden Seiten bestehen:

- Login
- Neues Profil
- Einstellungen
- Datenschutzerklärung
- Sensortest
- Impressum
- Liste aller Projekte
- Aktuelles Projekt
- Fragebogen

4.3.1. Login

Damit der Anwender das die App verwenden kann, muss er sich authentifizieren. Bei dem ersten Start der App wird er nach seinen Anmeldeinformationen gefragt. Hat der Anwender noch kein Profil angelegt, so hat er die Möglichkeit dies über das Login durchzuführen.

4.3.2. Menü

Die App verfügt über ein zentrales Menü, welches man über einen Button an der oberen linken Ecke aufrufen kann. Das Icon der drei übereinanderliegenden Striche, sowie die Menüform, wir als Burgermenü bezeichnet. Dieses verlinkt zu den jeweiligen zentralen Seiten in der App und sorgt für eine übersichtliche und schnelle Navigation. Zudem kann sich der Anwender über das Burgermenü abmelden.

Es besteht aus den folgenden Punkten:

- Alle Projekte
- Aktuelles Projekt
- Einstellungen
- Sensortest
- Impressum
- Abmelden

4.3.3. Einstellungen

In den Einstellungen kann der Anwender seine App konfigurieren und die folgenden Einstellungen vornehmen:

- Verwaltung der lokalen Profile in der Datenbank
- Verwaltung über die Nutzung von Sensoren
- Datenschutzerklärung anzeigen lassen

4.4. Entwurf

4.4.1. Vorgehen beim Entwurf

Der Entwurf der Anwendung wird über das Top-Down Verfahren realisiert. (65)(S. 35) Es sind bereits die Anforderungen gegeben. Diese abstrakten Ziele werden dann immer detaillierter definiert. Alternative Vorgehensweisen wären das Button-Up Verfahren, bei dem Randbedingungen und Stakeholder gegeben sind und sich aus den detaillierten Vorgaben abstrahiert. Als dritte Vorgehensweise dient das Outside-In Verfahren zur Entwicklung des Inneren Systems ausgehend von externen Schnittstellen.

4.4.2. Entwurfsmuster

Ein Entwurfsmuster ist eine Schablone aus dem Bereich der Softwarearchitektur und beschreibt den Aufbau einer Software. Dabei gibt es je nach Einsatzzweck unterschiedliche Muster, welche bei der Entwicklung eingesetzt werden können.

- Factory Pattern
- Template Pattern
- Model-View-Controller (MVC)
- Model-View-ViewModel (MVVM)

In der zu entwickelnden App wird das Entwurfsmuster Model-View-ViewModel (MVVM) verwendet, welches von Xamarin vorgegeben wird. Dabei handelt es sich um eine Abwandlung des Model-View-Controller Musters und dient der Trennung von Darstellung und Funktionalität. Es bietet die Möglichkeit die Entwicklung besser in Entwicklung der Oberfläche und der Funktionalität aufzuteilen, wodurch die Wartung des Quellcodes erleichtert wird.

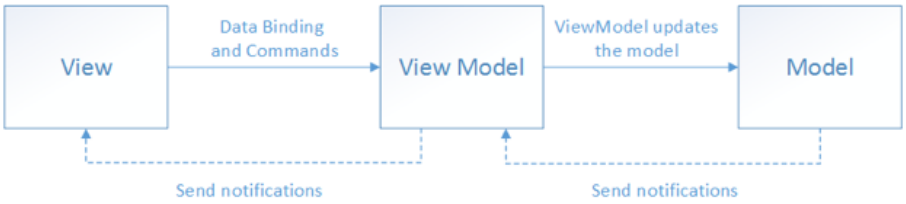

Abbildung 4.1.: MVVM Muster (50)

View

In der Methode Ansicht (View) wird die Anwendungsoberfläche definiert. Bei Xamarin ist dies eine Xaml-Datei, welche auf XML basiert. Diese beschreibt die verwendeten Elemente, sowie deren Anordnung. Zudem wurde eine C# Datei erstellt, in der die Elemente initialisiert werden können.

Model

Das Modell (Model) ist ein Datenmodell, welches vom Anwender eingesehen, sowie bearbeitet werden kann. Es wird in der App als Grundlage für den Aufbau der Datenbank verwendet. Dabei spiegelt eine Klasse, eine Tabelle in der Datenbank wider. Innerhalb des Modells befindet sich die Geschäftslogik. Diese kann mit Unit Tests getestet werden.

ViewModel

Die für die Oberfläche benötigte Logik befindet sich in den ViewModels. Dabei greift View auf öffentliche Funktionen dieser zurück. Die Daten bekommt das ViewModel von den Model, welche die Datenbank repräsentiert.

4.5. Open Data Kit

Die App verfügt über eine Anbindung an das Datenerhebungsframework ODK, siehe dazu Kapitel 3.1. Entsprechend können neue Projekte über ODK Build erstellt werden und in ODK Central ausgewertet werden. Die Anbindung erfolgt über eine Schnittstelle im Programm, welche dazu dient, die erstellten Formulare zu parsen, zu einem Projekt zusammenzufassen und die gesammelten Daten zurück an das Framework zu senden.

Open Data Kit Build

Projekte bestehen aus mehreren Formularen. Diese werden mit dem Open Data Kit Build erstellt.

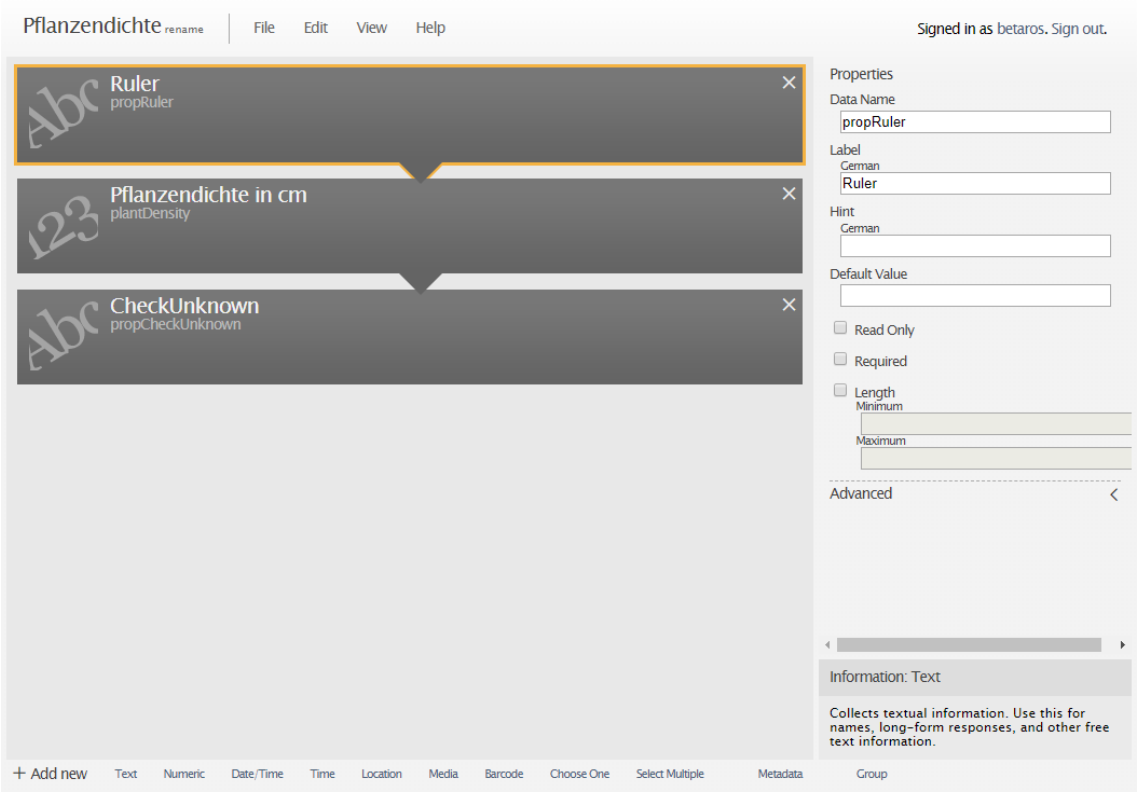

Abbildung 4.2.: Open Data Kit Build Webapp

Erstellung von Projekten

Für die Erstellung der Formulare wird ODK Build verwendet. Eine Webseite, welche es dem Anwender ermöglicht über eine übersichtliche Oberfläche ein Formular zu erstellen und zu exportieren. Dabei kann das Formular als JSON Datei mit der Dateiendung *.odkbuild* heruntergeladen werden, sowie auch als *.xml*. Dabei ist zu beachten, dass man in jedem Projekt mehrere Formulare integrieren kann und man somit auch je nach Anwendungsfall mehrere Formulare erstellen muss.

Die App verfügt über einen Parser für das jeweilige Dateiformat. Dabei werden die Formulare im System wie in Abbildung 4.11 verwaltet. Die Entwicklung basiert dabei auf dem vorliegenden JSON-Format.

4.6. Kommunikationsprotokolle

Zwischen dem Backend Server und der App wird mit unterschiedlichen Protokollen kommuniziert. Die Definition von Formularen wird mit der Markdownsprache JSON definiert. Unter anderem werden Formulare, welche in der App als eigenständige Tabs definiert sind mit der Open Data Kit Build Webapp erstellt. Zudem gibt es weitere Kommunikationsprotokolle, welche für den Austausch von Daten zwischen den App und Backend verwendet werden.

4.6.1. Übertragungsformate

JSON

Bei der Erstellung von Formularen wird die Open Data Kit Build Webapp verwendet. Dabei werden die Formulare als *.odkbuild*-Datei im JSON-Format gespeichert. Die Webapp ist Teil des Open Data Kit Frameworks. Als Beispiel ist im Anhang B.1 die Darstellung des *Phänologischen Stadiums* als Formular im JSON Formular angefügt.

XForms

Der XForms Standard wurde 2009 von der W3C unter (46) veröffentlicht. In dem Standard wird das Format und die Sprache beschrieben, in welcher die Daten ausgetauscht werden. Dabei handelt es sich um eine Markup Sprache, welche plattformunabhängig verwendet werden kann. Da die Erstellung der Formulare in ODK Build in JSON Format ausgegeben wird, wurde XForms nicht verwendet.

XLSForm

Während der XForm Standard eine eigenständige Markup Sprache besitzt und auf jeder Plattform verwendet werden kann, richtet sich dieser Standard mehr an Anwender, welche es bevorzugen mit Microsoft Excel aus dem Officepaket von Microsoft zu arbeiten. Das Format wird unter (5) beschrieben. Auch XLSForm wurde, wie XForms, aufgrund des Exportformats in ODK Build nicht verwendet. Zudem ist das programmatische Einlesen von Daten aus JSON wesentlich einfacher zu implementieren.

4.6.2. Verschlüsselung

Es gibt Bereiche der Anwendung, welche besonderen Schutz durch Verschlüsselung benötigen. So darf das Passwort des Anwenders nicht im Klartext gespeichert werden, da dieses sonst einfach ausgelesen werden kann. Diese Gefahr muss auch bei der Kommunikation mit dem Server zur Übertragung der Messdaten beachtet werden.

Profildaten

Wie bereits erwähnt wurde, wird das Passwort des Anwenders verschlüsselt in der Datenbank gespeichert. Dabei wird bei der Profilerstellung das eingegebene Passwort mit dem SHA512 Algorithmus (34) verschlüsselt. Dieses wird zur Zeit vom Bundesamt für Sicherheit in der Informationstechnik (BSI), als kryptographisch stark eingeschätzt. (2)

Projekte

Der Inhalt von Projektdaten zur Erstellung neuer Projekte bedarf keiner gesonderten Verschlüsselung. Jedoch wurde eine Begrenzung für den Zugriff auf ein Projekt gefordert. Dabei sollte es dadurch möglich sein, die Gruppe an Anwendern einzuschränken. Dadurch soll sichergestellt werden, dass die Teilnehmer an einem Projekt bekannt sind und keine Daten von Unbekannten erhoben werden können. Dementsprechend ist es möglich, ein Projekt mit einem Passwort zu sichern. Der Anwender verifiziert durch die Eingabe des Passwortes, dass er berechtigt ist, an dem Projekt teilzunehmen. Wie auch bei den Profildaten, wird auf den SHA512 Algorithmus gesetzt. Der Anwender gibt das Projektpasswort in der App ein. Dies wird dann mit SHA512 verschlüsselt und abgeglichen. Bei der Kommunikation mit dem Server wird das Passwort wieder verglichen. Dadurch ist eine lokale Manipulation des Passworts im Projektarchiv, ohne Folgen für die Erfassung der Daten auf dem Server.

4.7. Szenarien im Konzept

In diesem Abschnitt wird das Konzept auf die unterschiedlichen Szenarien abgebildet. Jedes Szenario beschreibt dabei den konzeptionellen Aufbau der jeweiligen Module, die in dem Szenario verwendet werden. Die Bedienung der einzelnen Szenarien wird in Kapitel 5.5 beschrieben. Wenn nicht anders angegeben, befinden sich alle Klassen in dem Unterprojekt DLR_Data_App. Die verwendeten Tabellen in der Datenbank werden in Kapitel 4.8 beschrieben.

4.7.1. Neues Profil anmelden

Dieses Szenario beschreibt die Anmeldung, sowie die Erstellung eines neuen Profils. Dabei wird auf die Datenbank zugegriffen, in der alle Profile gespeichert sind. Zudem werden die Profile in einem User-Modell abgebildet.

Modell

Die Userklasse beinhaltet die grundlegenden Informationen über den Anwender. Jeder Anwender erhält eine einzigartige lokale Id, seinen Nutzernamen, sowie ein in SHA-512 verschlüsseltes Passwort.

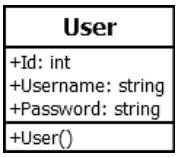

Abbildung 4.3.: Modell User

Verwendete Klassen

Nachfolgend ist die Verzeichnisstruktur abgebildet, mit den verwendeten Dateien für dieses Szenario.

- Models
  - User.cs
- Services
  - Database.cs
  - Helper.cs
- ViewModels
  - Login
    - LoginViewModel.cs
    - NewProfileViewModel.cs
- Views
  - Login
    - LoginPage.xaml
    - LoginPage.cs
    - NewProfilePage.xaml
    - NewProfilePage.cs

Programmablaufpläne

Login Das Login Fenster dient zur Identifikation eines Anwenders. Dieser muss seinen Nutzernamen, sowie sein Passwort zur Verifikation angeben. Das Passwort intern, wie in Kapitel 4.6.2 beschrieben, direkt nach dem Tippen auf den Bestätigungsbutton mit SHA-512 verschlüsselt. Wenn die Anmeldung erfolgreich war, wird dem Anwender die Datenschutzerklärung angezeigt. Nach der EU-DSGVO (siehe Kapitel A) muss nachgewiesen werden, dass der Anwender der Datenschutzerklärung zugestimmt hat. Mit der Bestätigung der Datenschutzerklärung wird die EU-DSGVO eingehalten. Sollte der Anwender dem widersprechen, so wird er zurück auf das Loginfenster geleitet. Der Anwender hat, sofern er keinen Account besitzt, die Möglichkeit einen neuen Account anzulegen.

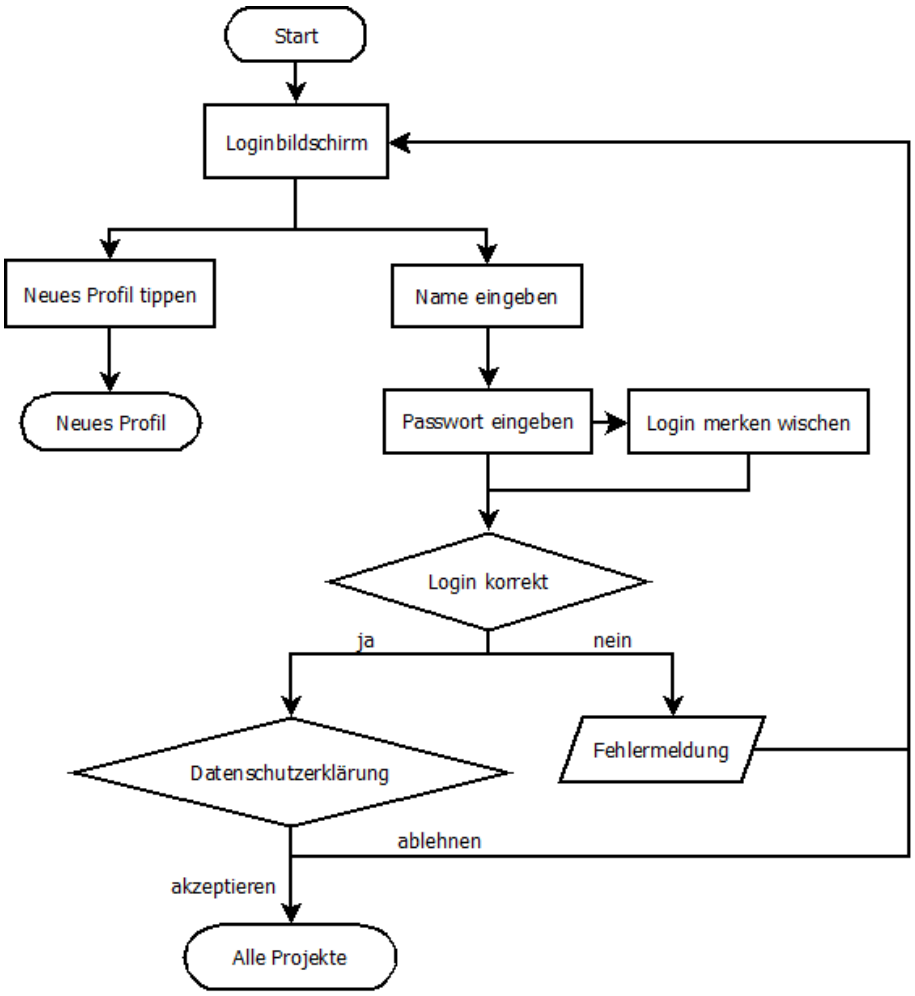

Abbildung 4.4.: Programmablaufplan Login

Neues Profil Über den Login Bildschirm kann man zu dem Fenster zum Erstellen eines neuen Profils gelangen. Dort gibt man einen Wunschnamen und -passwort an. Man hat jederzeit die Möglichkeit den Vorgang abzubrechen und zurück zum Login zu gelangen.

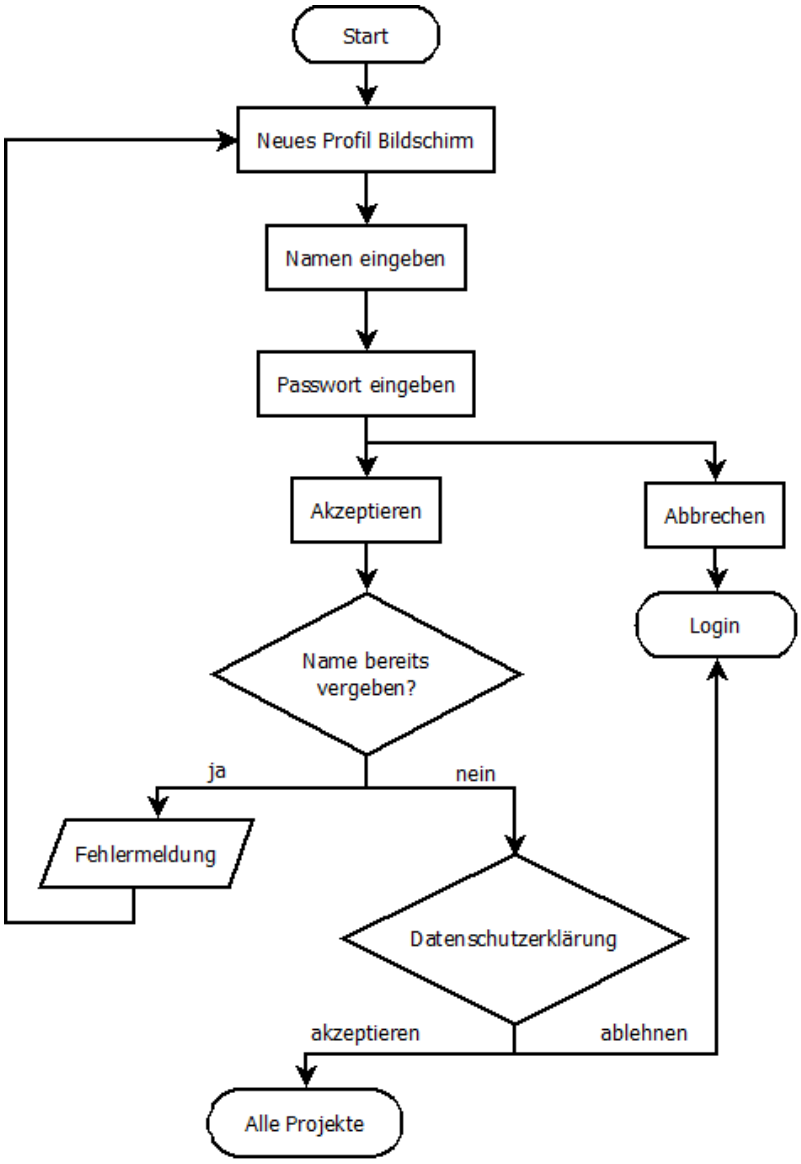

Abbildung 4.5.: Programmablaufplan Neues Profil

4.7.2. Hauptmenü

Das Hauptmenü wird beim Antippen des Burger-Icons aufgerufen. Es verweist auf die einzelnen Abschnitte der App. Zusätzlich hat der Anwender die Möglichkeit, sich über das Menü abzumelden. Der Anwender kann zu den folgenden Fenstern navigieren:

- Aktuelles Projekt
- Alle Projekte
- Sensor Test
- Einstellungen
- Impressum
- Abmelden

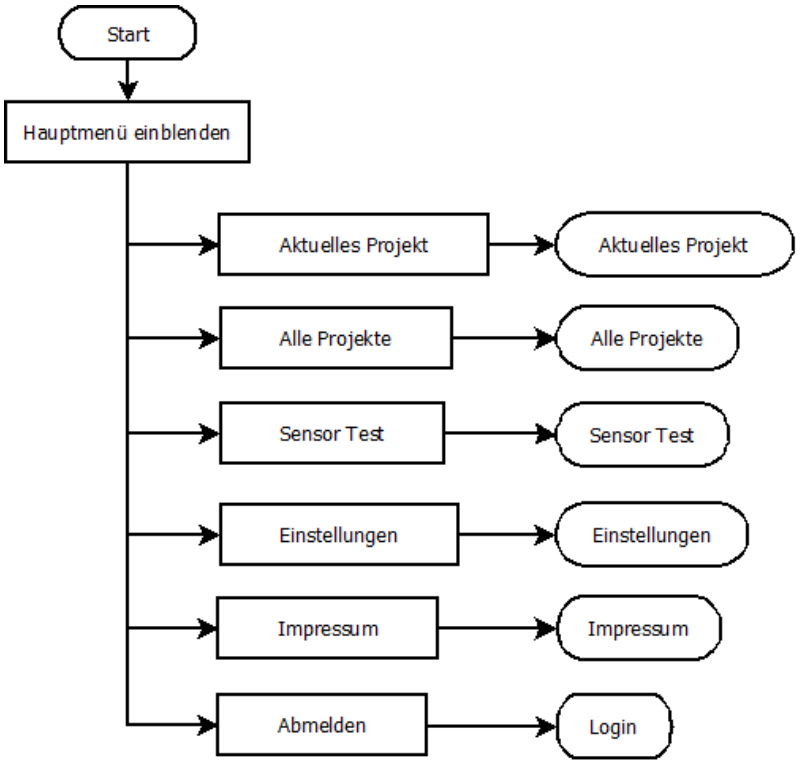

Abbildung 4.6.: Programmablaufplan Hauptmenü

4.7.3. Projekt auswählen

Der Anwender kann von Projekten das gewünschte Projekt auswählen. Befindet sich noch kein Projekt in der Liste, oder möchte der Anwender ein neues Projekt hinzufügen, kann er dies über das Szenario *Projekt erstellen*, in Abschnitt 4.7.5 beschrieben, durchführen.

Modell

Bei der Modellierung eines Projektobjekts wurde sich an der Ausgabe aus dem Open Data Kit Build erstellten JSON Datei orientiert. So besteht ein Projekt aus mehreren Formularen, welche wiederum aus einzeln definierten Elementen , sowie Metadata bestehen. Die für die Erstellung der Formulare notwendigen Informationen sind in den *ProjectFormElements* definiert. Die Klasse *Project* beinhaltet die Informationen über den Ersteller, eine Projektbeschreibung, optionales Passwort, welches in SHA-512 verschlüsselt ist, und integrierte Sprachen.

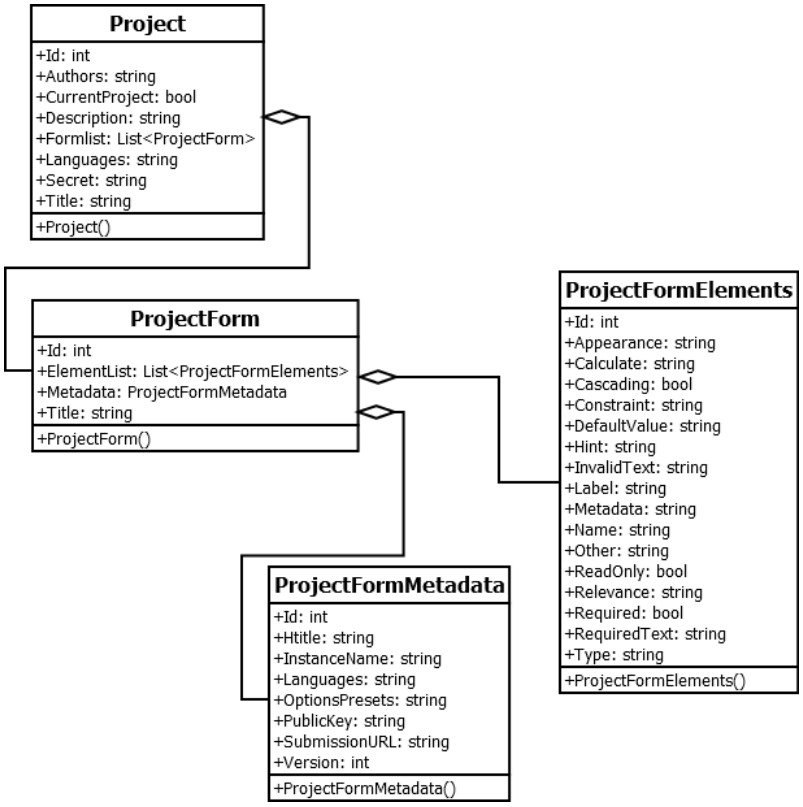

Abbildung 4.7.: Modell Projekt

Verwendete Klassen

- Models
  - ProjectModel
    - Project.cs
    - ProjectForm.cs
    - ProjectFormElements.cs
    - ProjectFormMetadata.cs

Programmablaufplan

Auf der Seite *Alle Projekte* werden in einer Liste alle bisher hinzugefügten Projekte dargestellt. Dabei können in einem neuen Fenster weitere Informationen zu jedem Projekt aufgerufen werden. Zudem lässt sich das ausgewählte Projekt als Aktuelles Projekt auswählen, oder aber löschen.

Über den Button *Hinzufügen* an der oberen rechten Ecke, können weitere Projekte geladen werden. Es öffnet sich ein neues Fenster, in dem über einen Dateibrowser ein ZIP-Archiv ausgewählt werden kann. Die App erkennt, ob es sich um eine Datei im richtigen Format handelt. Danach kann der Anwender den Importvorgang bestätigen oder abbrechen. Mit der Bestätigung beginnt die App mit dem auslesen des Archivs. Über eine Statusnachricht wird der Anwender darüber informiert, ob das Laden erfolgreich war.

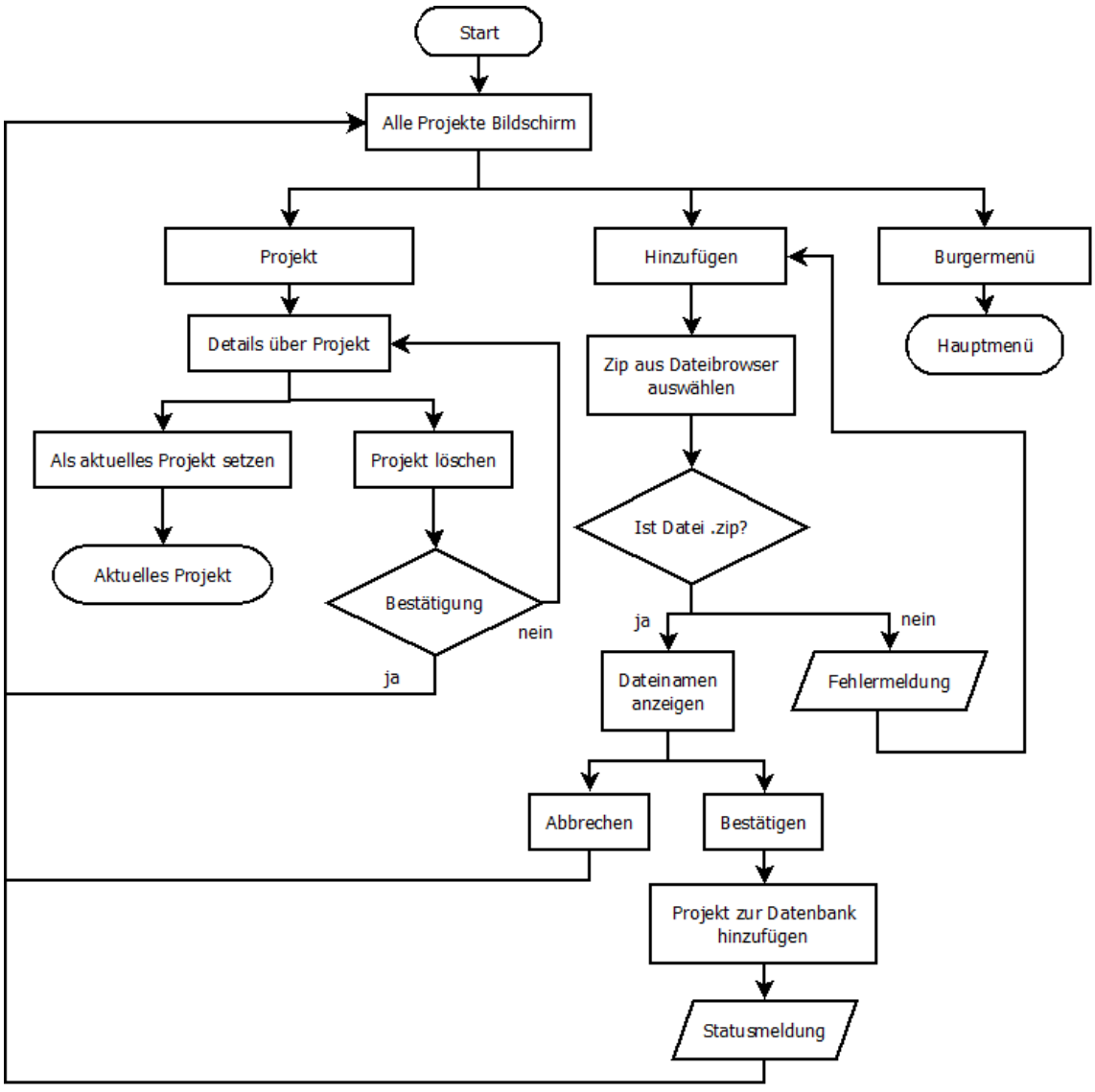

Abbildung 4.8.: Programmablaufplan Alle Projekte

4.7.4. Projekt erstellen

Für die Erstellung eines neuen Projekts wird die Open Data Kit Build Webapp verwendet. Dieses Szenario ist in Kapitel 5.5 beschrieben.

4.7.5. Projekt hinzufügen

Das Hinzufügen eines neuen Projekts erfolgt in der Klasse *ProjectGenerator.cs*. Dort werden alle benötigten Schritte durchgeführt. Die Schritte müssen in der folgenden Reihenfolge nacheinander folgen:

1. Archiv entpacken
2. Dateien parsen
3. Daten in Datenbank schreiben
4. Liste aller Projekte aktualisieren

Modell

Bei dem verwendeten Modell handelt es sich um das Projektmodell, welches bereits in Abschnitt 4.7.3 dargestellt wurde.

Verwendete Klassen

- Models
  - ProjectModel
    - Project.cs
    - ProjectForm.cs
    - ProjectFormElements.cs
    - ProjectFormMetadata.cs
- Services
  - Database.cs
  - Helper.cs
  - Parser.cs
  - ProjectGenerator.cs
- ViewModels
  - ProjectList
    - NewProjectViewModel.cs
- Views
  - ProjectList
    - NewProjectPage.cs

Programmablaufplan

Generierung neuer Projekte in der Datenbank Die in der Einleitung definierte These umfasst die dynamische Generierung von Formularen. Dabei müssen die in Open Data Kit Build erstellten Formulare geparsed und in der Datenbank gespeichert werden.

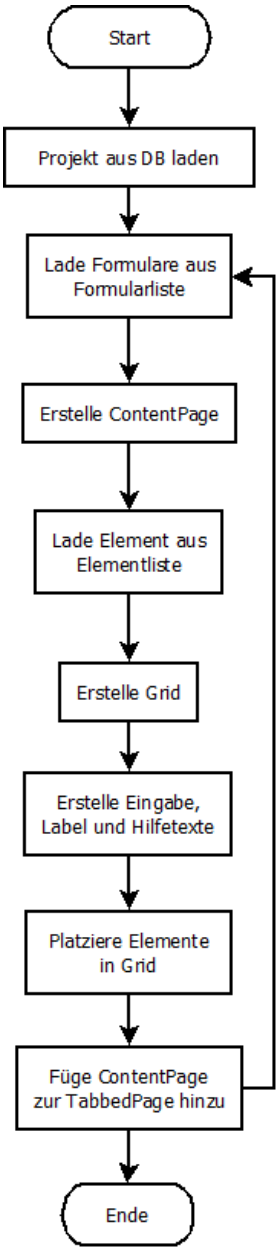

Abbildung 4.9.: Programmablaufplan Projekt laden

4.7.6. Daten erfassen

Verwendete Klassen

- Models
  - ProjectModel
    - Project.cs
    - ProjectForm.cs
    - ProjectFormElements.cs
    - ProjectFormMetadata.cs
- Services
  - Database.cs
  - Helper.cs
- ViewModels
  - CurrentProject
    - ProjectViewModel.cs
- Views
  - CurrentProject
    - ProjectPage.xaml
    - ProjectPage.cs

Programmablaufplan

Erstellung einer UI aus der Datenbank Bei der Darstellung des aktuellen Projekts werden die Informationen über die UI-Elemente aus der Datenbank geladen. Diese werden in ein Projekt Modell überführt, aus dem das Fenster (ContentPage) generiert wird. Für jedes Formular wird ein eigener Tab generiert. Aus der Elementliste wird so lange ein Element entnommen, bis die Liste leer ist. Dabei wird ein Raster erstellt und mit dem Titel des Elements, einen Button für die Hilfetexte und der passenden Eingabemethode generiert. Das Raster wird dann, samt Inhalt, zum Formular Tab hinzugefügt.

Wenn die Seite angezeigt wird, kann der Anwender seine Eingabe tätigen. Dabei kann er auf jeden Hilfebutton tippen und sich einen Hilfstext anzeigen lassen. Über die Buttons an der oberen rechten Ecke, kann dieser auch das Projekt bearbeiten oder aber speichern.

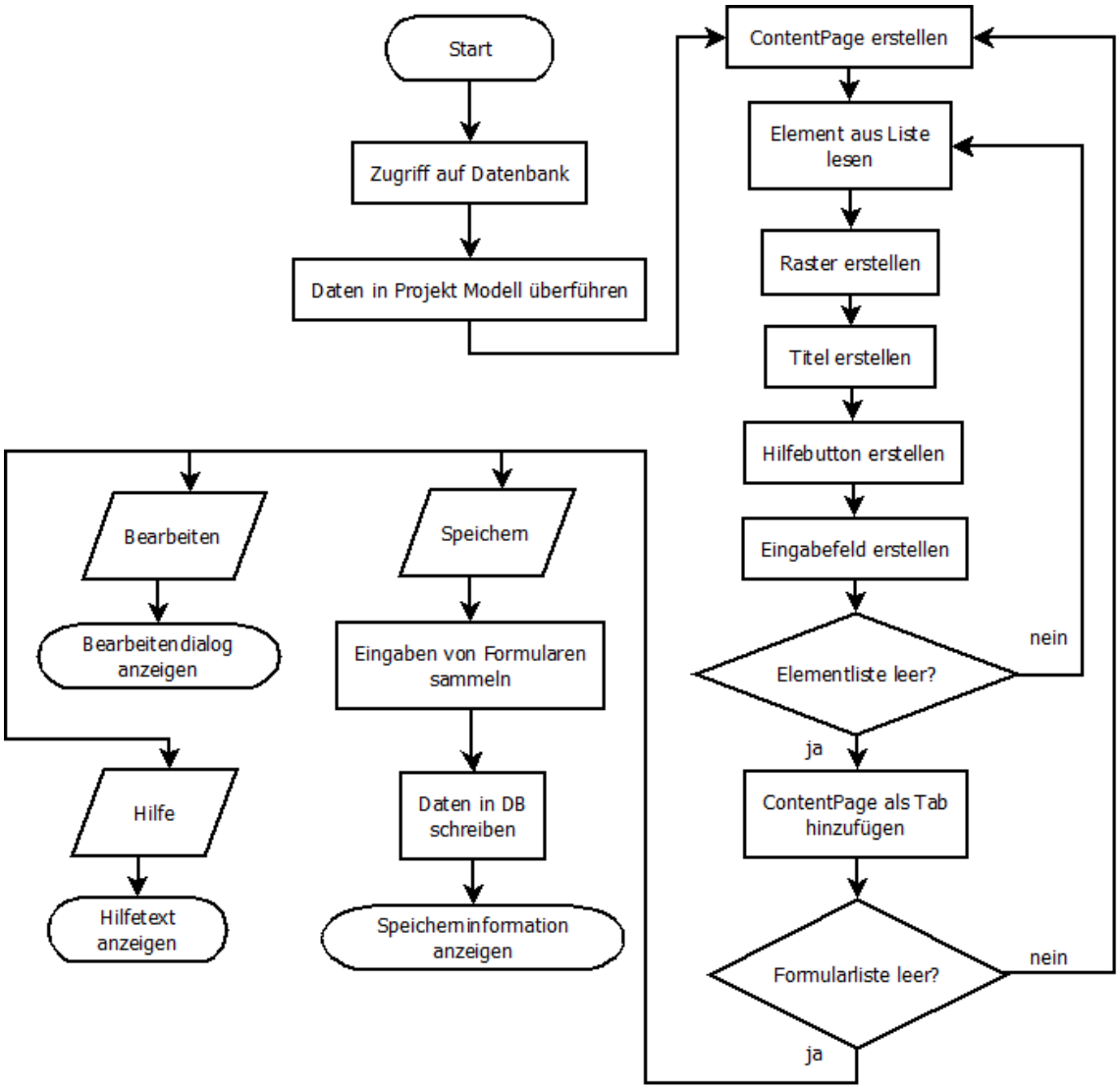

Abbildung 4.10.: Programmablaufplan UI generieren

Aktuelles Projekt Beim Fenster für das aktuelle Projekt wird aus der Liste der vorhandenen Projekte in der App ein bereits angelegtes Projekt verwendet. Dabei ist es entscheidend, ob der Anwender den Fragebogen bereits beantwortet hat. Ist dies nicht der Fall, so muss der Anwender einen Fragebogen ausfüllen, um ein Profiling des Anwenders durchführen zu können. Danach wird er zum Projekt weitergeleitet. Das Projekt umfasst mehrere Fragebögen, welche über die Tabs verteilt werden und ihrerseits über Wischbewegungen mit der Hand gesteuert werden. Über das Burger-Menü kann man jederzeit das Hauptmenü öffnen und zu einem anderen Fenster wechseln. Das Icon mit den drei Punkten gibt dem Anwender die Möglichkeit, Beschreibungen zum aktuellen Formular zu lesen oder auch den Fragebogen zu wiederholen.

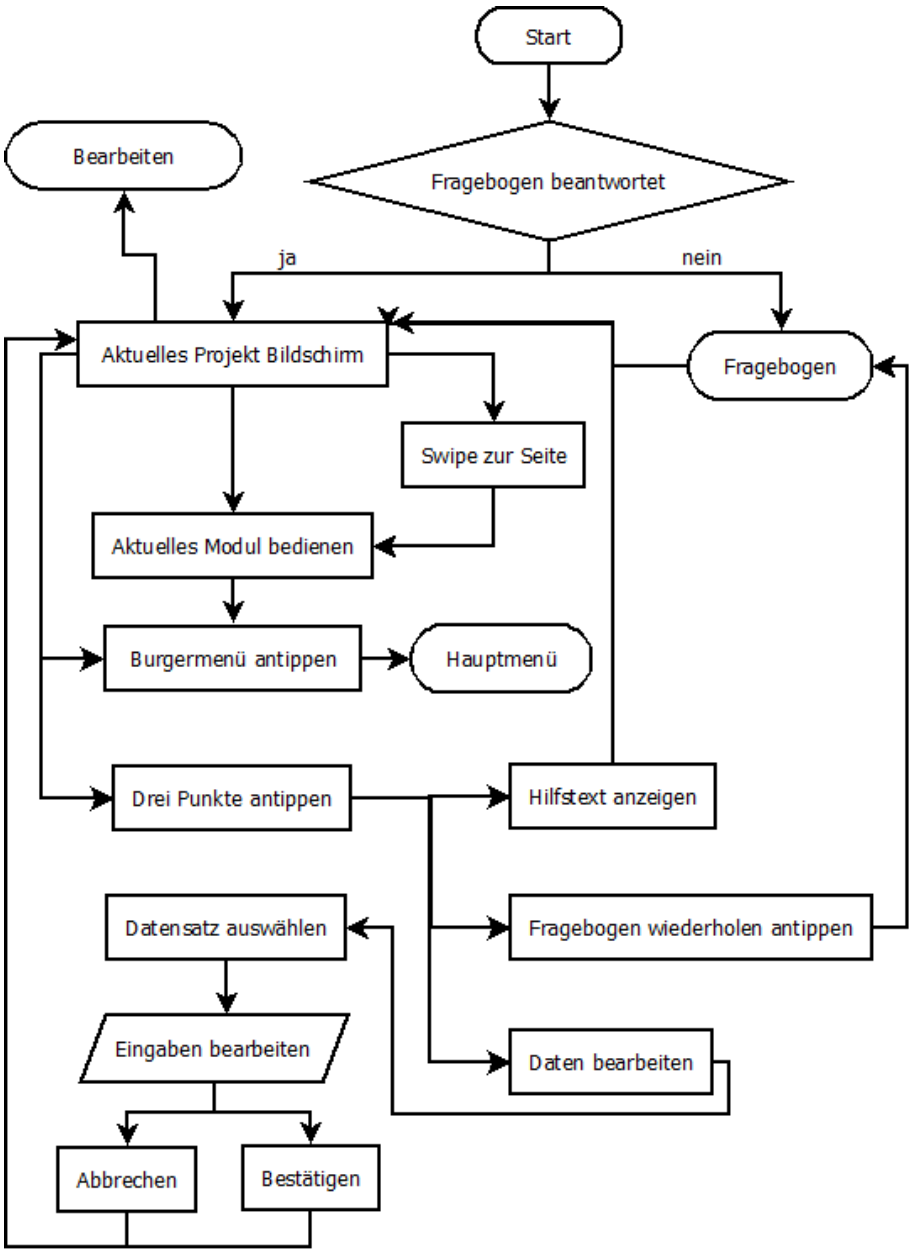

Abbildung 4.11.: Programmablaufplan Aktuelles Projekt

4.7.7. Daten bearbeiten

Neben der Datenerfassung, müssen die Daten innerhalb der Anwendung auch veränderbar sein. Dies ist wichtig bei der Korrektur von Fehleingaben. Diese Funktion wurde in dem Prototypen nur zum Teil implementiert. Der Anwender kann sich im Prototypen alle getätigten Eingaben anzeigen lassen, jedoch nicht verändern.

Modell

Bei dem verwendeten Modell handelt es sich um das Projektmodell, welches bereits in Abschnitt 4.7.3 dargestellt wurde.

Verwendete Klassen

- Models
  - ProjectModel
    - Project.cs
    - ProjectForm.cs
    - ProjectFormElements.cs
    - ProjectFormMetadata.cs
- Services
  - Database.cs
- ViewModels
  - CurrentProject
    - EditDataDetailViewModel.cs
    - EditDataViewModel.cs
- Views
  - CurrentProject
    - EditDataDetailPage.xaml
    - EditDataDetailPage.cs
    - EditDataPage.xaml
    - EditDataPage.cs

Programmablaufplan

Der Anwender kann auf der Seite zum Bearbeiten der Daten in einer Liste alle eingegebenen Daten in einer Liste sehen. Über die Auswahl des Tages und der Zeit kann er in der Liste passende Elemente sich anzeigen lassen. Dabei bleiben alle Elemente in der Liste vorhanden, es wird jedoch ein Element, passend zum Zeitstempel, markiert. Der Anwender kann dann auf ein Element tippen und sich dieses anzeigen lassen.

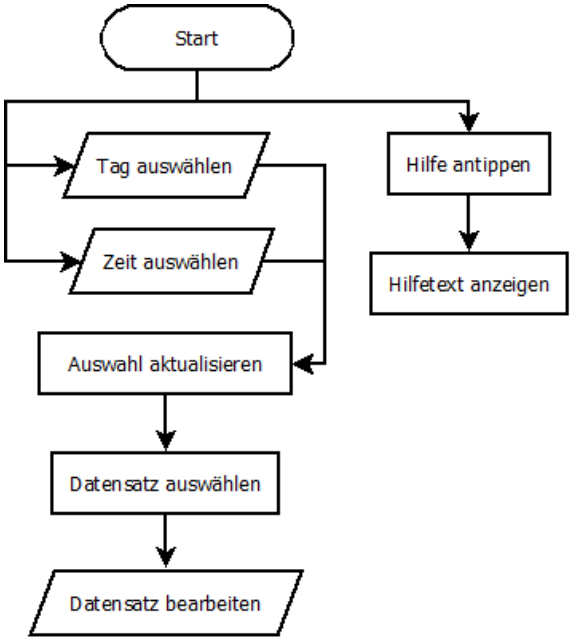

Abbildung 4.12.: Programmablaufplan Daten bearbeiten

4.8. Datenbank

Für die lokale Speicherung von Daten wird eine Datenbank auf dem Smartphone erstellt. Dabei handelt es sich um SQLite, welche alle Daten innerhalb einer Datei verwaltet. Der Vorteil gegenüber anderen Datenbanksystemen wie MySQL oder CouchDB ist der geringe Ressourcenverbrauch und einfache Verwendung innerhalb der App.

Die Anbindung der Datenbank wird über das NuGet-Paket *sqlite-net-pcl* ermöglicht. Weitere Details über das Paket ist in der Tabelle 6.1 aufgeführt. Das Paket ermöglicht die vorhandenen Daten in den Datenmodellen direkt in der Datenbank abzubilden. Dabei werden die Datenstrukturen übernommen. Dies wird dadurch limitiert, dass keine Fremdschlüssel und Listen übernommen werden, sowie eigene Datentypen selbstständig in die Datenbank übernommen werden müssen. (22)

Tabellen

Die Datenbank umfasst die folgenden Tabellen, welche sich aus den Modellen ergeben. Diese sind in Kapitel 4.7 definiert. Neben den statisch angelegten Tabellen, gibt es zudem generierte Tabellen, dabei wird für jedes Projekt eine eigene Tabelle generiert. Alle Elemente der Formulare werden dort gemeinsam in einer Tabelle abgebildet.

Nachfolgend wird die Datenbankstruktur der statischen Modelle dargestellt.

Tabelle 4.1.: Tabelle User

Name	Datentyp
Id	integer
Username	varchar
Password	varchar

Tabelle 4.2.: Tabelle Projekt User Connection

Name	Datentyp
Id	integer
ProjectId	integer
UserId	integer
Weight	float

Tabelle 4.3.: Tabelle Projekt Formular Elemente

Name	Datentyp
Id	integer
FormId	integer
ElementId	integer
MetadataId	integer

Tabelle 4.4.: Tabelle Projekt Form

Name	Datentyp
Id	integer
Title	varchar
ProjectId	integer

Tabelle 4.5.: Tabelle Projekt Formular Metadata

Name	Datentyp
Id	integer
Version	integer
Languages	varchar
OptionsPresets	varchar
Htitle	varchar
InstanceName	varchar
PublicKey	varchar
SubmissionUrl	varchar

Tabelle 4.6.: Tabelle Projekt

Name	Datentyp
Id	integer
Title	varchar
Description	varchar
Authors	varchar
Secret	varchar
Languages	varchar
CurrentProject	integer

Tabelle 4.7.: Tabelle Projekt Formular Elementliste

Name	Datentyp
Id	integer
Name	varchar
Label	varchar
Hint	varchar
DefaultValue	varchar
ReadOnly	integer
Required	integer
RequiredText	varchar
Relevance	varchar
Constraint	varchar
InvalidText	varchar
Calculate	varchar
Options	varchar
Cascading	integer
Other	varchar

4.9. Elemente

4.9.1. Profil

Zentraler Bestandteil der App ist das Profil. Der Anwender kann auf seinem Smartphone mehrere Profile verwenden. Bei Projekten die mehrere Anwender haben, soll es möglich sein, dass wenn ein Teilnehmer während der Datenerhebung bei nachlassender Kapazität seines Smartphone-Akkus auf ein anderes Gerät auszuweichen und sich so, zwei Anwender ein Smartphone teilen können.

Die Erstellung des Profils erfolgt lokal auf dem Smartphone. Es ist dabei keine Kommunikation mit dem Server notwendig. Dadurch ist neben dem Datenschutz personenbezogener Daten, auch die Offline-Nutzung der App sichergestellt.

Neben der Erstellung und Verwaltung des Profils werden auch Projekte mit dem Account verknüpft. Dies dient dazu, Einstellungen des Anwenders zu den jeweiligen Projekten zu verwalten. Zudem kann das Ergebnis des Fragebogens für jedes Projekt dort hinterlegt, bzw. zurückgesetzt werden.

4.9.2. Projekt

Ein Projekt wird als eine Sammlung von Formularen bezeichnet, welche zusätzlich die Möglichkeit haben, den Anwender über Fragebögen einschätzen zu können. Die Formulare, sowie die Fragebögen und eine beschreibende README-Datei[1] werden in einem ZIP-Archiv[2] zusammengefasst.
Jedes Formular wird dabei in drei Unterabschnitte eingeteilt:

- Titel
- Content
- Metadata

Neben dem Titel des Formulars wird im Abschnitt *Content* jedes verwendete Element im Formular definiert. Im Abschnitt Metadata werden weitere Details gespeichert. Dabei wird jedes Formular über das Programm ODK Build aus dem Open Data Kit generiert und als JSON im *.odkbuild* Dateiformat exportiert.

4.9.3. Fragebogen

Die Fragebögen, welche zur Einschätzung des Anwenders verwendet werden, sind Teil der Projektarbeit von Studenten der Friedrich-Schiller-Universität in Jena. Die Implementierung und Beschreibung der Fragebögen wird aus diesem Grund nicht in der Masterarbeit behandelt.

[1] Eine Textdatei mit generellen Informationen über das Projekt
[2] Eine verlustfreie Komprimierung von Dateien, welche 1989 von Phil Katz entwickelt wurde.

4.10. Module

Eine der wichtigsten Anforderungen an die App ist die Einteilung in Module 4.2. Dadurch wird die Wartung der App und das Hinzufügen von weiteren Programmteilen vereinfacht. Aus der Bausteinansicht aus Kapitel 4.11 ergibt sich eine Übersicht über die Einteilung der Module. Diese werden weiter zusammengefasst und in diesem Kapitel näher beschrieben.

4.10.1. Allgemein

Als allgemeine Module werden Module bezeichnet, welche ohne zutun anderer Module, Funktionen übernehmen und nicht genauer kategorisiert werden. Es handelt sich dabei um Menüs und die Anzeige einzelner statischer Daten. Zu den allgemeinen Modulen gehören die folgenden Funktionen:

- Hauptseite
- Impressum
- Menü
- Sensortest

4.10.2. Login

Dieses Modul beinhaltet die Funktionen zur Anmeldung und Erstellung eines neuen Profiles.

Anmeldung

Bei der Anmeldung wird der Nutzername und das Passwort eingegeben. Dabei wird bei der Eingabe jedes Zeichen durch ein Sternchen dargestellt. Der Anwender kann nur das letzte eingegebene Zeichen für eine Sekunde sehen, bevor es in ein Sternchen umgewandelt wird.

Die Funktion hinter dem Login Button läd eine Liste aller gespeicherten Anwender. Dabei wird der Nutzername in Klartext mit der Eingabe verglichen. Das Passwort wird in der Datenbank nur verschlüsselt in SHA512 abgespeichert, daher muss das eingegebene Passwort auch verschlüsselt werden. Danach können beide Hashwerte der Passwörter miteinander verglichen werden. Dies dient der Datensicherheit und schützt den Anwender vor Passwortdiebstahl. Wenn die Eingabe mit einem Anwender aus der Liste übereinstimmt, sowie auch das Passwort zu dem dazugehörigen Anwender stimmt, wird die Datenschutzerklärung angezeigt. Es steht dem Anwender frei, sich für oder dagegen zu entscheiden. Nur wenn er sich dafür entscheidet, wird er zur App und den Projekten weitergeleitet.

Neues Profil

Wenn der Anwender ein neues Profil anlegen möchte, kann er im Loginfenster auf den Button *Neues Profil* tippen. Er wird dann weitergeleitet an das Fenster zur Erstellung eines neuen Profils. Dort können der gewünschte Name, sowie das gewünschte Passwort gesetzt werden. Auch hier kann der Anwender vom Passwort nur das letzte Zeichen für eine Sekunde sehen, bevor es von Sternchen ersetzt wird.

Bei dem Erstellen eines neuen Profils wird aus der Datenbank die Liste aller bestehenden Profile geladen. Danach wird abgeglichen, ob der Nutzername bereits verwendet wurde. Ist dies nicht der Fall wird das eingegebene Passwort mit SHA512 verschlüsselt und der Nutzername, sowie das Passwort in der Datenbank gespeichert.

4.10.3. Einstellungen

Die App kann über die Einstellungen den Bedürfnissen des Anwenders angepasst werden.

Allgemeine Einstellungen

Zu den allgemeinen Einstellungen gehören die folgenden Optionen:

- Datenschutzerklärung anzeigen
- Datenbank löschen

Profilverwaltung

In der Profilverwaltung ist es möglich, die einzelnen Profile sich auflisten zu lassen und das Passwort, sowie den Namen ändern, oder aber das Profil löschen.

Sensorverwaltung

Es können für alle Anwender Sensoren aktiviert und deaktiviert werden.

4.10.4. Projekte

Die App soll variabel für mehrere Forschungsprojekte verwendet werden können. Dabei wird jedes Forschungsprojekt als Projekt definiert. Dies kann über die App geladen und verwaltet werden. Die Erhebung von Daten erfolgt über die in der Datenbank abgelegten Informationen zu den Projekten. Daraus wird dynamisch eine Oberfläche generiert, welche über Hilfetexte, sowie Eingabefelder verfügt.

4.10.5. Services

Häufig verwendete Funktionen werden zu Services zusammengefasst. Die Implementation dieser wird in Kapitel 6.2 genauer beschrieben.

Sensoren

Die Verwendung der Sensoren erfolgt über die in den Services hinterlegten Sensorliste. Diese beinhalten alle für die Entwicklung notwendigen Eigenschaften des jeweiligen Sensors und bieten ein Modell mit allen Informationen, die der Sensor beinhaltet.

Datenbank

Ein zentraler Service ist die Schnittstelle zur Datenbank. Die Klasse dient der Ausführung der grundlegenden CRUD Befehle. Zudem beinhaltet es die Logik für das Auslesen und Abspeichern der erfassten Daten.

Helferklasse

Die Helferklasse dient bei der Entwicklung für häufig verwendete Funktionen, welche kleinere Aufgaben aus unterschiedlichen Bereichen erfüllt. Dabei handelt es sich unter anderem um die Verschlüsselungsalgorithmen und das übersetzen von Elementen aus der Datenbank.

4.11. Sichten

Das entwickelte System kann in unterschiedliche Sichten unterteilt werden. Dabei zeigt jede Sicht eine eigene Facette der App auf. Dabei richtet sich sich Arbeit nach dem Buch *Effektive Softwarearchitekturen: Ein praktischer Leitfaden* von Gernot Starke. (65)

4.11.1. Kontextabgrenzung

In der Kontextabgrenzungssicht wird das System als Blackbox dargestellt. Es soll die Schnittstellen zu den Nutzern, Betreibern und Fremdsystemen darstellen, inklusive der transportierten Daten oder Ressourcen. Zudem wird die technische Systemumgebung, Prozessoren und Kommunikationskanäle dargestellt. (65)(S. 161)

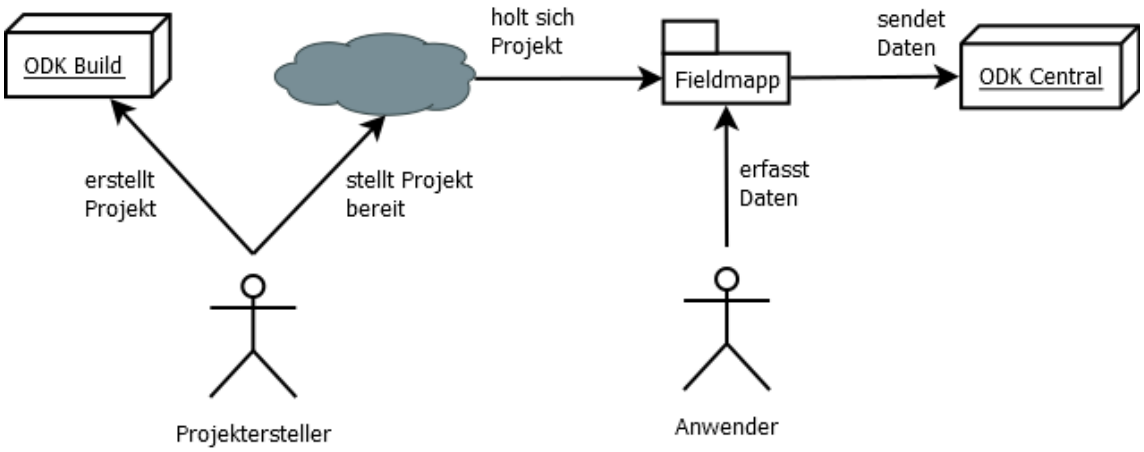

Abbildung 4.13.: Kontextabgrenzung

Bei der App gibt es zwei Nutzergruppen. Da sind zum einen die Projektersteller, welche Projekte mit dem Open Data Kit Build erstellen. Die erstellten Projekte werden dann über das Internet für die Anwender bereitgestellt. Sie laden die erstellten Projekte in ihre App und können nach einem Profiling Daten in der App erfassen. Die gesammelten Daten werden danach an das Open Data Kit Central übertragen.

4.11.2. Bausteinansicht

Die Bausteinansicht gibt Aufschluss über die Aufgaben des Systems und stellt die benötigten Teile als Baustein dar. Dabei kann zwischen Bausteinen unterschieden werden, welche als Blackbox dargestellt werden oder Bausteinen, die die Arbeitsweise und Struktur darstellen. In der Bausteinansicht werden die Komponenten, Pakete, Klassen, Subsysteme und Partitionen, sowie Abhängigkeiten zwischen den Bausteinen dargestellt. Zudem wird dargestellt, welche Bausteine, zur Erfüllung der Anforderungen benötigt werden. (65)(S. 164)

Bausteinübersicht

Bei der Aufteilung der App in Bausteine wird diese in die folgenden Abschnitte unterteilt:

- Login
- Neues Profil
- Einstellungen
- Projektliste
- Aktuelles Projekt
- Impressum
- Sensortest

Die jeweiligen Bausteine werden daraufhin weiter unterteilt in die jeweiligen Funktionsabschnitte. Jeder Baustein, welche an der DLR Data Collection angrenzt, kann als eigenständiges Modul angesehen werden.

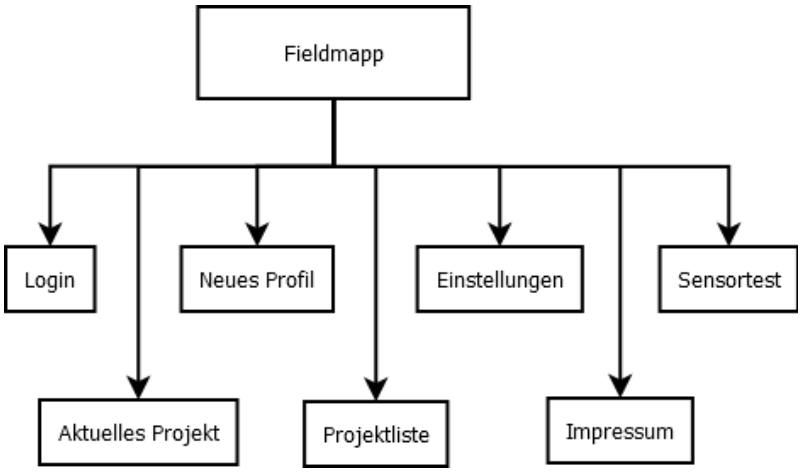

Abbildung 4.14.: Bausteinansicht Gesamt

Login

Die Anmeldung dient der Identifikation des Anwenders. Dabei umfasst die Anmeldung eine Verifikation, welche auf die Datenbank zugreift und überprüft, ob das gegebene Profil vorhanden ist. Ist die Identifikation erfolgreich, so wird die Datenschutzerklärung angezeigt, welches ein weiterer Unterbaustein ist.

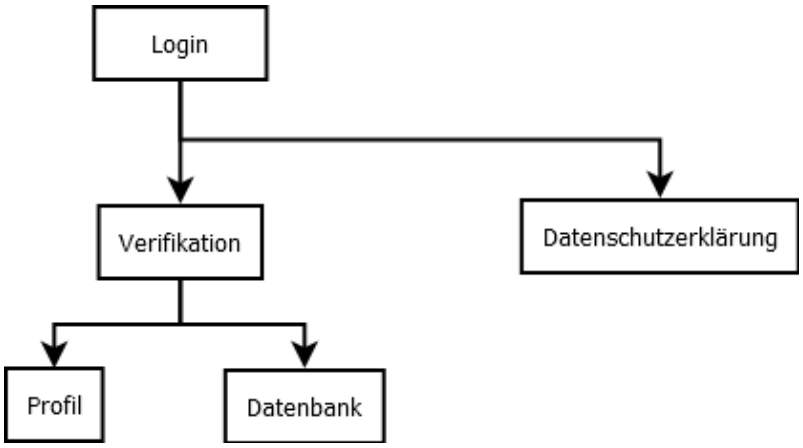

Abbildung 4.15.: Bausteinansicht Login

Neues Profil

Ein neues Profil enthält viele Bausteine, die bereits im Login verwendet wurden und auch die Oberfläche ähnelt dem Login. Beide unterscheiden sich darin, dass bei der Erstellung des neuen Profils überprüft wird, ob es bereits ein Profil mit dem Namen gibt. Ist dies nicht der Fall, wird ein neuer Nutzer zur Datenbank hinzugefügt. Anschließend muss dieser die Datenschutzerklärung bestätigen.

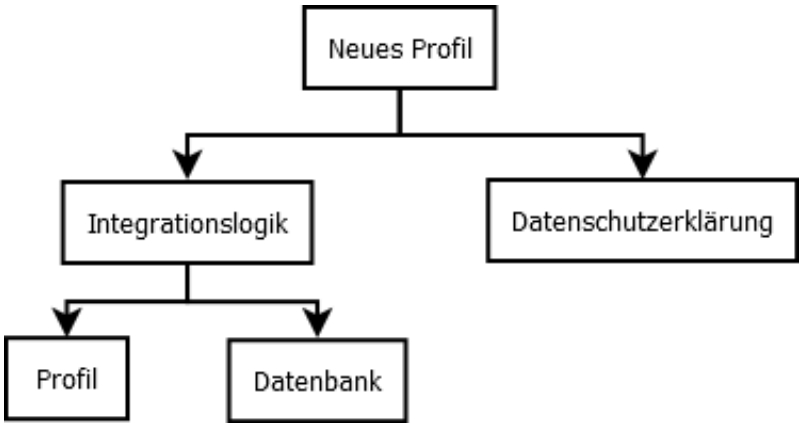

Abbildung 4.16.: Bausteinansicht Neues Profil

Einstellungen

Die Einstellungen umfassen allgemeine Konfigurationen, die Profilverwaltung, sowie die Aktivierung und Deaktivierung von Sensoren. Zu den allgemeinen Einstellungen gehört, das Löschen der gesamten Datenbank, sowie der Export dieser und die Anzeige der Datenschutzerklärung. In der

Profilverwaltung kann man von jedem Profil aus das Passwort ändern, sowie Profile löschen. Dies führt zu einem Rechteproblem, welches in Kapitel 8.3.2 beschrieben wurden.

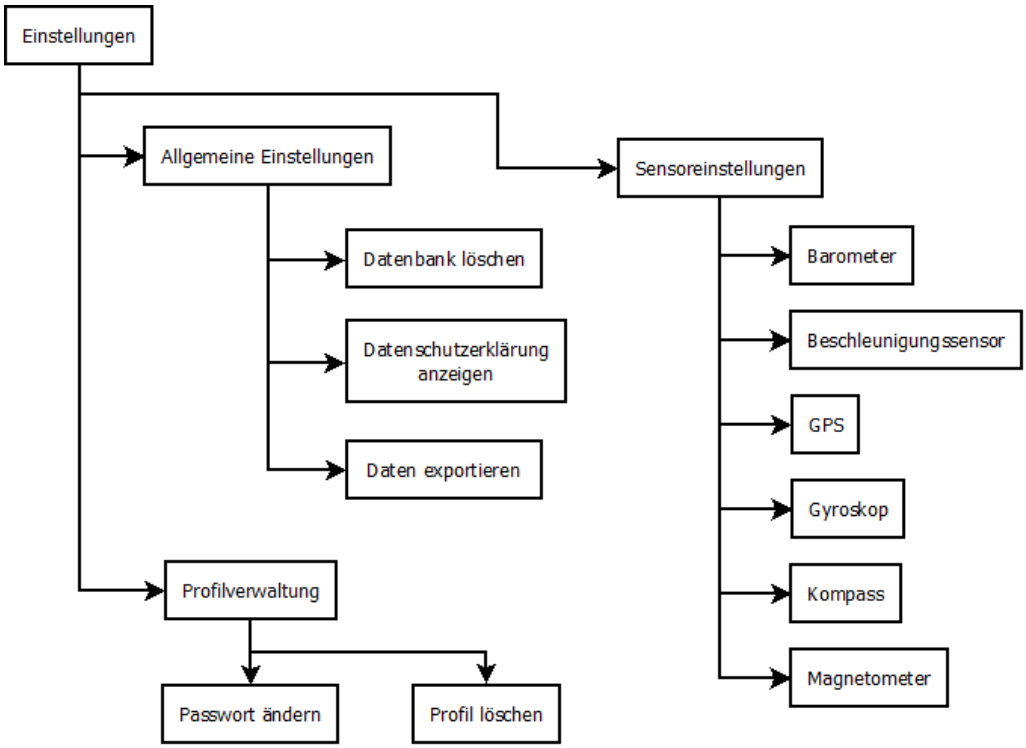

Abbildung 4.17.: Bausteinansicht Einstellungen

Projektverwaltung

Nach dem erfolgreichen Anmelden des Users mit seinem Profil, wird die Projektverwaltung über den Hauptbildschirm aufgerufen. Dort können neue Profile hinzugefügt werden und bereits existierende verwaltet. Somit ist es möglich, mehrere Projekte zu verwalten und zu verwenden.

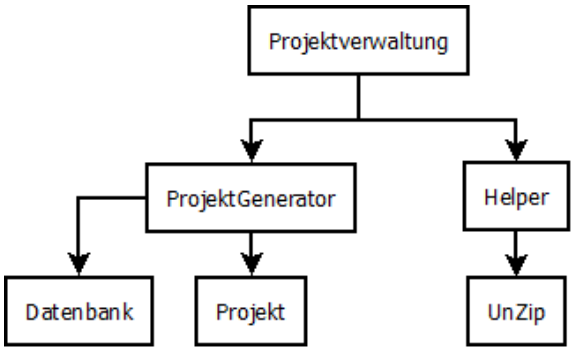

Abbildung 4.18.: Bausteinansicht Projektverwaltung

Aktuelles Projekt

Bei der Erfassung der Daten können Projekte nicht parallel verwendet werden. Aus diesem Grund wird ein Projekt, welches bereits zur App hinzugefügt wurde, als aktuelles Projekt markiert. Beim Laden des Fensters werden alle Informationen über das Projekt, wie Formulare und Informationen über die Projektersteller aus der Datenbank geladen. Aus den Einstellungen werden die Konfigurationen über die Aktivität der Sensoren geladen.

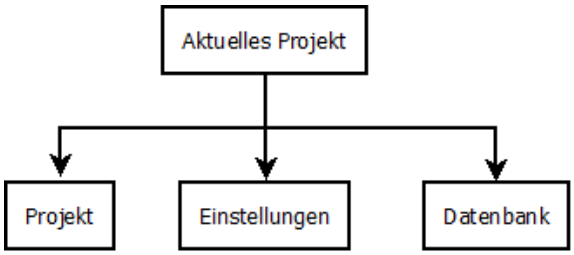

Abbildung 4.19.: Bausteinansicht Aktuelles Projekt

Sensortest

Zur Überprüfung der Sensoren wurde der Sensortest hinzugefügt. Dieser kann hilfreich bei der Bestimmung des Messfeldes sein, und die Teilnehmer bei der Erfassung von Daten helfen. Er besteht aus einer Seite mit Zugriff auf die einzelnen Sensoren, deren Werte dargestellt werden.

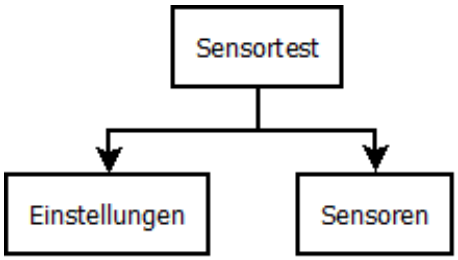

Abbildung 4.20.: Bausteinansicht Sensortest

Impressum

Das Impressum besteht nur aus einer Seite. Dort wird auf das DLR, die Hochschule Stralsund und die Friedrich-Schiller-Universität Jena hingewiesen, sowie allen beteiligten Personen.

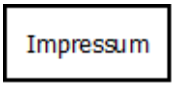

Abbildung 4.21.: Bausteinansicht Impressum

4.11.3. Laufzeitsicht

Bei der Laufzeitsicht wird dargestellt, welche Bestandteile des Programms zur Laufzeit verwendet werden und wie sie gemeinsam interagieren. Dabei dient diese Ansicht hauptsächlich dazu, die wesentlichen Aufgaben darzustellen und so Use-Cases abzudecken. (65)(S. 170)

In der Abbildung 4.22 wird der gesamte Ablauf für die Verwendung der App dargestellt. Dabei wird in den folgenden Abbildungen auf die einzelnen Abschnitte der App eingegangen.

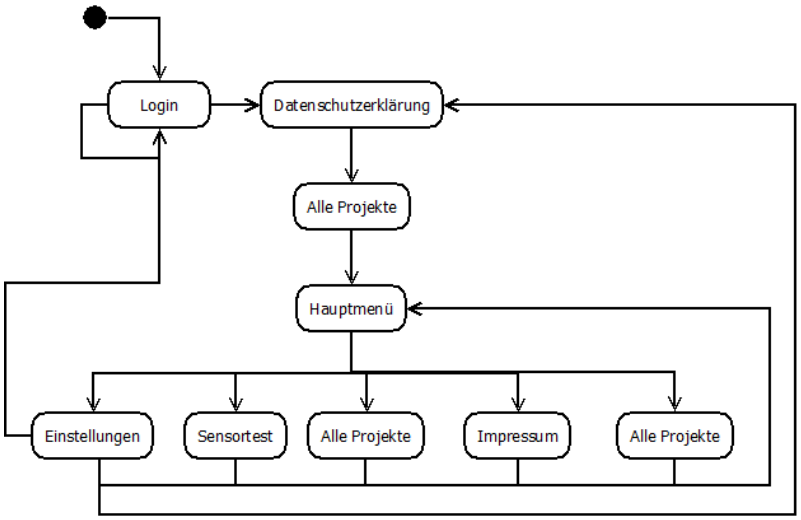

Abbildung 4.22.: Gesamte Laufzeitansicht

5. Usability und User Experience (UX)

Dieses Kapitel befasst sich mit der Entwicklung der Nutzeroberfläche und der Programmführung.

5.1. Standards

Es gibt im Bereich von Usability und UX mehrere Standards, welche von unterschiedlichen Organisationen entwickelt wurden. Sie dienen zur Vereinheitlichungen in der Entwicklung mit dem Ziel, Schnittstellen zueinander zu schaffen und den Anwender zu unterstützen. Als größte und bekannteste Organisation erstellt die Internationale Organisation für Normung (ISO) mit Sitz in Genf (Schweiz) internationale Standards. Neben den internationalen Normen der ISO werden bei der Entwicklung auch die Richtlinien der Plattformhersteller berücksichtigt. Da sich die Entwicklung auf Android und iOS beschränkt, werden Richtlinien der jeweiligen Plattform eingehalten.

5.1.1. ISO 9241-11

Dieser Standard beschreibt Schlüsselwörter und Konzepte, sowie die fundamentalen Grundlagen der Usability. Zudem wird beschrieben, wie die Konzepte angewendet werden können. Die ISO 9241-11 wurde 1998 festgelegt und 2018 überarbeitet. (41)

5.1.2. ISO 9241-210

Dieser Standard beschreibt die Anforderungen und Empfehlungen für anwenderorientierte Entwicklungsprinzipien für die Lebensdauer computerbasierter interaktiver Systemen. (42)

Er ist ein Teilbereich der ISO 9241-11 und beschreibt, was vor, während und nach der Benutzung einer interaktiven Anwendung

5.1.3. Android Design Guidelines

Die Android Design Guidelines wurden von Google festgelegt und sind unter (31) erreichbar. Sie beschreiben den grundsätzlichen Aufbau einer App, damit der Anwender diese der Android Plattform zuordnen kann und ein einheitliches Design über alle Apps erhält. Im Gegensatz zu Apple sind die Designrichtlinien nicht zwingend zu beachten, damit die App im Google Playstore verfügbar ist. Dies führt zu einer größeren Vielfalt an Apps, jedoch auch zu einen weniger einheitlichen App Design.

5.1.4. iOS Human Interface Guidelines

Die von Apple vorgegebenen Design Richtlinien (12) werden vor der Veröffentlichung einer App überprüft. Dabei werden Apps, welche gegen diese Richtlinien verstoßen, nicht im Apple Appstore zugelassen. Dadurch ist das Design des UI plattformweit einheitlich und übersichtlich.

5.2. Heuristiken

Bei der Entwicklung einer Anwendung sollte diese nach Jakob Nielsen bestimmten Heuristiken folgen, um für Anwender möglichst nutzfreundlich zu sein. Dies beschreibt er in seinem Paper aus dem Jahre 1994 (57).

- Sichtbarkeit des Systemstatus
- Übereinstimmung von System und Wirklichkeit
- Nutzerkontrolle und Freiheit
- Beständigkeit und Standards
- Fehlervermeidung
- Wiedererkennung statt Erinnerung
- Flexibilität und Effizienz
- Ästhetisches und minimalistisches Design
- Hilfestellung beim Erkennen, Bewerten und Beheben von Fehlern
- Hilfe und Dokumentation

Die App wird im Kapitel 8.2 anhand dieser Heuristiken auf ihre Usability untersucht.

5.3. Personas

Um eine erfolgreiche Anwendung zu entwickeln ist es notwendig sich an die künftigen Anwender und ihre Bedürfnisse zu orientieren. So werden Nutzerstudien durchgeführt und aus den Daten fiktive Personen erstellt, welche die unterschiedlichen Anwendergruppen repräsentieren.

5.3.1. Datenerhebung

Für die Erhebung von Daten zur Erstellung von Messdaten wurden unterschiedliche Wege gewählt. Es wurden drei Methoden zur Erfassung angewandt.

Vor-Ort-Beobachtung
Bei den Vor-Ort-Beobachtungen werden Anwender dabei beobachtet, wie sie unterschiedliche Aufgabe angehen, welche Probleme auftreten und welche Anforderungen der Anwender an die App stellt. Es ist dabei notwendig, dass die beobachtende Person möglichst geringe Interaktion hat, um die Messungen nicht zu verfälschen. Ein Effekt dieser Methode ist die Erfassung unbewusster Arbeitsabläufe und Nutzungsmustern. (44)

Diese Methode wurde bei einer Feldkampagne für das Projekt DEMMIN angewandt, siehe Kapitel 2.5.1. Dabei wurden die Studenten dabei beobachtet, wie sie die Messungen durchführten und welche Probleme auftraten.

Befragung
Zusätzlich zu den Vor-Ort-Beobachtungen wurden Befragungen durchgeführt. Dies ist bei der Entwicklung von Personas für Projekterstelle hilfreich, da so Informationen zu den Anforderungen an die Anwendung ersichtlich werden.

Studien
Um weitere Einsatzzwecke abdecken zu können, werden unterschiedliche Studien ausgewertet. Ziel ist es eine möglichst genaue und realistische Einschätzung von Personengruppen zu erzeugen, welche dann als einzelne Persona abgebildet wird.

5.3.2. Inhalt Persona

Eine Persona dient als Verallgemeinerung einer Anwendergruppe zu einer fiktiven Person. Diese dient als Referenz für die Anforderungen der Anwendung. Entsprechend werden die erfassten Parameter der Persona spezifisch für die Anwendung angepasst.

Name
Zur Personifikation wird ein neutraler Name gewählt, welcher keine persönlichen Eigenschaften beinhaltet.

Foto
Ein Foto dient den Entwicklern zur besseren Visualisierung des Anwenders.

Alter
Das Alter dient zur Einschätzung der Person. So können dadurch die unterschiedlichen Personengruppen zugeordnet werden.

Brillenträger
Dies dient zu Einschätzung, welche Farben, Kontraste und gestalterischen Elemente in der App verwendet werden. Zudem werden weitere Besonderheiten auf die visuelle Erfassung des Anwenders

erfasst, wie z.B. ob der Anwender Farbenblind ist.

Medienaffinität
Über die Medienaffinität kann eingeschätzt werden, in wie weit sich der Anwender mit dem Smartphone auskennt und dieses Bedienen kann. Entsprechend kann das Nutzerkonzept angepasst werden.

Familienstand
Der Familienstand dient allein zur Personifikation der Persona und wird nicht in der Entwicklung berücksichtigt.

Wohnort
Dies spielt, wie auch der Familienstand in der Entwicklung keine Rolle. Sie dient zur Personifikation für die Entwickler.

Beruf
Anhand des Berufs lassen sich Merkmale über Medienkompetenzen herleiten.

Hobbies
Das Hobby hilft dabei die Person zu identifizieren. Aber auch dabei welche Themen der Person wichtig sind.

Motivation
Die Motivation hilft dabei die Bereitschaft zum lernen zu erkennen, wie auch die Intentionen der Person.

Herausforderungen für UX-Designer
Jede Person hat gewissen Ansprüche an eine App, welche als Herausforderung an die UX-Designer gestellt wird.

Botanische Kenntnisse
Im Bezug auf die Feldkampagne des Projekts DEMMIN spielen die vorhandenen Kenntnisse über Pflanzen eine entscheidende Rolle.

5.3.3. Entwickelte Personas

Projektersteller Der Projektersteller ist der zentrale Ansprechpartner für das Projekt. Er betreut das Projekt und legt fest, welche Daten erhoben werden und wie die Fragestellungen für die Einstufung der Teilnehmer des Projekts gestellt werden. Damit nimmt er die Position des Projektadministrators ein und verfügt zudem über alle erfassten Daten.

Tabelle 5.1.: Fiktive Personas Projektersteller

	Peter Meier	Tina Schmidt
Alter	56	29
Brillenträger	ja	nein
Medienaffinität	eigene Webseite	soziale Netzwerke
Familienstand	verheiratet	ledig
Wohnort	Neubrandenburg	Jena
Beruf	Professor	Doktorandin
Hobbies	Reisen	Schwimmen
	Kino	Violine spielen
Motivation	möchte genaue Messergebnisse haben	möchte Promotion schreiben
Herausforderungen für UX-Designer	kurze Einarbeitungszeit	einfacher Export von Daten
Botanische Kenntnisse	Botaniker	besitzt Zimmerpflanzen

Aus den gegebenen Personas lässt sich ableiten, die Nutzung einer Webapp zur Generierung der Formulare und einem anschließenden erstellen eines ZIP-Archivs, eine einfache Lösung zur Projekterstellung haben. Der Export der gesammelten Messdaten ist über einen Button in den Allgemeinen Einstellungen implementiert. Dies vereinfacht die Verwendung der Messdaten bei der Auswertung.

Teilnehmer Forschungsprojekt
Bei den Anwendern wird zwischen Teilnehmern von Forschungsprojekten und öffentlichen Projekten unterschieden. Der Unterschied besteht darin, dass die Teilnehmer eines Forschungsprojekts bekannt sind und durch einen Betreuer geschult werden.

Tabelle 5.2.: Fiktive Personas Teilnehmer Forschungsprojekt

	Tobias Müller	Stefanie Peters
Alter	20	24
Brillenträger	ja, farbenblind	nein
Medienaffinität	soziale Netzwerke	soziale Netzwerke
Familienstand	ledig	ledig
Wohnort	Burg Stargard	Neustrelitz
Beruf	Student	Studentin
Hobbies	Freunde treffen	Freunde treffen
	Programmieren	Volleyball spielen
Motivation	möchte Forschung voran bringen	möchte Kenntnisse erweitern
Herausforderungen für UX-Designer	einfache Bedienung auch für Farbenblinde	modernes, einfaches Design
Botanische Kenntnisse	keine	besitzt Zimmerpflanzen

Bei den beiden Personas handelt es sich um Studenten, die durch die Verwendung von sozialen Medien, bereits mit digitalen Medien vertraut sind. Als Herausforderung für den UX-Designer muss darauf geachtet werden, dass Farben keinen Einfluss auf die Verwendung der App haben. So

werden in der App zur Differenzierung von Elementen Kontraste und Helligkeitsstufen verwendet. Dadurch wird auch auf Personen mit Rot-Grün-Sehschwäche eingegangen, welche die Farben Rot und Grün nicht differenzieren können.

Anwender öffentliches Projekt
Anwender von öffentlichen Projekten sind meist anonym und haben keinen direkten Ansprechpartner bei der Erfassung der Messdaten. Zudem sind die Altersgruppen und sozialen Schichten diverser als in den meisten Forschungsprojekten.

Tabelle 5.3.: Fiktive Personas Anwender öffentliches Projekt

	Jens Schneider	Helga Pfaff
Alter	43	74
Brillenträger	nein	ja
Medienaffinität	eigener YouTube Kanal	nein
Familienstand	verheiratet	ledig
Wohnort	Rostock	Stralsund
Beruf	Kaufmann	Rentnerin
Hobbies	Heimwerken	Gärtnern
Motivation	möchte Stadt mitgestalten	möchte auf Probleme hinweisen
Herausforderungen für UX-Designer		hoher Kontrast einfache Bedienung
Botanische Kenntnisse	besitzt Zimmerpflanzen	eigener Garten

Die Personas für öffentliche Projekte sind aus unterschiedlichen Altersschichten und technischen Kompetenzen. So muss bei Helga Pfaff auf eine möglichste einfache Bedienung geachtet werden, um so die Eingabefehler zu minimieren. Auch muss auf einen hohen Kontrast geachtet werden. Dabei ähneln sich die Anforderungen an die Lesbarkeit der App, mit denen der Anwender in Forschungsprojekten.

5.4. Design

Unter Berücksichtigung der Standards und Richtlinien der iOS und Android Plattformen wurde ein Designkonzept für die App entwickelt. Bei der Erstellung der Designkonzepte wurde Adobe XD verwendet, welches in Kapitel 2.2 beschrieben wurde.

5.4.1. Farben

Bei der Wahl der Farben wurden mehrere Aspekte berücksichtigt. Ein wichtiger Aspekt ist zum einen die Corporate Identity, welche als Hauptfarben Grau, Weiß und Gelb haben mit schwarzen Text. Dies musste darauf abgestimmt werden mit einem hohen Kontrast. Dies ist notwendig bei der Nutzung der App unter direktem Sonneneinfluss.

5.4.2. Navigation

Die App verfügt über mehrere Menüs. Dabei wurde das Master/Detail Pattern angewandt. Dies bietet die Möglichkeit gleichzeitig für Smartphones, wie auch Tablets optimiert zu sein. Es verfügt über ein Hauptmenü, welche von jeder Stelle, außer dem Login, erreichbar ist. Dieses erreicht man über ein Tippen auf das Burgericon. Da diese Menüführung bereits in mehreren Systemapps auf dem Smartphone bekannt ist, muss sich der Anwender nicht auf eine neue Menüführung einstellen.

Innerhalb der einzelnen Seiten navigiert der Anwender in verschiedenen Arten. In der Übersicht über allen Projekten wird jedes verfügbare Projekt aufgelistet, mit der zusätzlichen Information über den Ersteller. Interessiert sich der Anwender für ein Projekt, so kommt er mit einem Tippen auf das Projekt zu weiteren Projektdetails und kann sich für das Projekt anmelden. Je nach Projekt kann es sein, dass der Anwender ein Passwort eingeben muss.

Über die Auswahl des Projekts gelangt der Anwender zu einem Fragebogen, welcher dazu dient, seine Kenntnisse zu erfassen und einzuordnen. Die Navigation erfolgt dabei jeweils von Frage zu Frage. Nachdem die letzte Frage beantwortet wurde, gelangt man zur Projektseite, über die die Datenerfassung erfolgt.

5.4.3. Header

Eine App-übergreifende Designentscheidung ist der Header, welcher an der oberen Leiste der App den Anwender zusätzliche Funktionen bietet. Über den Header erfährt der Anwender, in welchen Fenster er sich befindet. An der linken oberen Ecke befindet sich das Symbol für das Hauptmenü, welches als Burgermenü aufgebaut ist. Das Hauptmenü wird von der linken Seite aus eingeblendet und überdeckt die Fenster. Auf der rechten Seite des Headers befinden sich Button, welche in den jeweiligen Fenstern Funktionen, wie das Hinzufügen von Projekten oder Bearbeiten von Daten hinzufügt.

5.4.4. Formulare

Die Formulare werden dynamisch generiert. Dabei besteht jedes Eingabefeld aus drei Elementen, welche tabellarisch angeordnet sind. Der Aufbau ist wie folgt:

Tabelle 5.4.: Aufbau Eingabefelder

Name	Hilfe
Eingabefeld	

5.5. Szenarien

Dieser Abschnitt beschreibt unterschiedliche Anwendungsszenarien, in denen die App verwendet wird. Dabei wird auf die deutsche Sprachausgabe eingegangen. Bei anderen Sprachen ist die Positionierung der Texte gleich.

5.5.1. Neues Profil anmelden

Im Loginfenster kann man auswählen, ein neues Profil anzulegen. Tippt der Anwender auf den *Neues Profil* Button wird dieser weitergeleitet zu einer neuen Seite. Dort muss der Profilname und das neue Passwort eingegeben werden. Der Anwender kann sich jederzeit dazu entscheiden, den Prozess abzubrechen. Möchte dieser das Profil erstellen, tippt er auf den *Erstellen* Button. Ist in der lokalen Datenbank kein Profil vorhanden, wird das Profil angelegt. Andernfalls erscheint eine Fehlermeldung und der Anwender kommt zurück auf die *Neues Profil* Seite.

5.5.2. Projekt erstellen

Die App startet mit einer leeren Liste an Projekten. Damit ein neues Projekt hinzugefügt werden kann, muss es zunächst vom Projektleiter erstellt werden. Dies kann man über die Webapp ODK Build vom Open Data Kit durchführen. (58)

Nachdem man sich auf der Webseite registriert hat, kann man Formulare erstellen, bearbeiten, speichern und gespeicherte Formulare löschen. Nachdem ein Formular erstellt wurde, muss man es über *File - Save Form to File* exportieren. Jedes Formular stellt dabei einen Tab in der Projektseite der App dar.

Zur Beschreibung des Projekts muss eine JSON-Datei erstellt werden, welche die Dateiendung *.json* aufweisen muss und das folgende Format hat:

Listing 5.1: Projektbeschreibung

```
{
    "Project": {
      "0": "Feldkampagne",
        "1": "Fieldcampagne"
    },
    "Author": {
        "0": "Deutsches Zentrum fuer Luft- und Raumfahrt e.V. (DLR) Jena und
            Friedrich Schiller Universitaet Jena",
        "1": "German Aerospace Center (DLR e.V.) Jena and Friedrich Schiller
            University Jena"
    },
    "Description": {
        "0": "Messungen des Wachstums von Pfanzen zur Kalibrierung von Satelliten"
            ,
        "1": "Measurements of the growth of plants for the calibration of
            satellites"
    },
    "Secret": null,
    "Languages": {
        "0": "German",
        "1": "English"
    }
}
```

Es ist möglich, das Projekt mehrsprachig zu definieren. Entsprechend muss unter dem Punkt *Languages* die Sprache nummeriert und in englisch benannt werden. In jedem Unterpunkt kann nun die Sprache definiert werden und der dazugehörige Text.

Die exportierten Dateien im *odkbuild*-Format werden danach über ein Archivierungsprogramm zu einem Zip-Archiv zusammengefasst. Dabei bietet sich z.B. 7zip an. Das Zip-Archiv sollte dann auf einem für die Anwender zugreifbaren Server liegen.

5.5.3. Projekt hinzufügen

Die erstellten Projekte werden auf dem Smartphone über den Webbrowser heruntergeladen. Dazu öffnet der Anwender einen beliebigen Webbrowser, navigiert zum Server, auf dem sich das Zip-Archiv befindet, und lädt diese Datei herunter. In der App kann man über den Menüpunkt *Alle Projekte* neue Projekte durch das Antippen auf *Hinzufügen* importieren.

Es öffnet sich ein weiteres Fenster, in dem das Archiv ausgewählt wird. Ist das ausgewählte Archiv kein Zip Archiv, erhält der Anwender eine Fehlermeldung. Es ist jederzeit in dem Fenster möglich, den Prozess abzubrechen. Mit dem Tippen auf *Hinzufügen* wird der Inhalt des Archivs geparst, und wenn es erfolgreich war, zur Liste aller Projekte hinzugefügt. Sollte das Parsen fehlschlagen, erhält der Anwender eine Fehlermeldung.

5.5.4. Projekt auswählen

Der Anwender kann ein neues Projekt über die Projektliste auswählen. Dort wählt er ein gewünschtes Profil aus. Daraufhin wird er zur Projektdetailseite weitergeleitet. Auf dieser kann das Projekt als *Aktuelles Projekt* ausgewählt werden. Er wird dann zu der Projektseite weitergeleitet. Der Anwender kann jederzeit über das Hauptmenü auf das aktuelle Projekt zugreifen.

5.5.5. Daten erfassen

Die Daten werden über das Fenster *Aktuelles Projekt* erfasst. Dort sind alle Formulare in jeweiligen Tabs aufgeführt. Der Anwender kann die Daten eingeben und dann mit dem Tippen auf den *Speichern* Button die eingegebenen Daten speichern. Zu jedem Eingabefeld kann ein Hinweistext aufgerufen werden, welcher den Anwender unterstützt bei der Eingabe.

5.5.6. Daten bearbeiten

Es können vom jeweiligen aktuellen Projekt die Daten bearbeitet werden. Dies ist über das Fenster des aktuellen Projektes mit dem Tippen auf die Schaltfläche *Bearbeiten*. Im weiterführenden Fenster kann ein Zeitpunkt ausgewählt werden, in dem die Daten verändert werden sollen. Nachfolgend werden die Daten in einer Liste dargestellt und der Datensatz mit dem Zeitstempel markiert. Dadurch ist es für den Anwender einfacher, einen Datensatz zu finden und zu bearbeiten.

6. Prototyp

Im Rahmen der Masterarbeit wurden Teile das Konzepts in einem Prototypen umgesetzt.

6.1. Verwendete Technologien

Für die App wurde das Xamarin Framework ausgewählt. Dies wurde in Kapitel 2.4.4 beschrieben. Dementsprechend wurde die Programmiersprache C# unter dem Mono-Framework verwendet. Mono ist eine offene Implementierung des .NET-Frameworks von Microsoft, welche ursprünglich unter für Linux entwickelt wurde.

Für die Entwicklung der App sind mehrere zusätzliche Pakete aus dem NuGet-Paketmanager verwendet worden. Diese erweitern das Xamarin-Framework um Funktionen, welche für die Entwicklung notwendig sind.

Tabelle 6.1.: Verwendete NuGet Pakete

Paketname	Autor	Beschreibung	Lizenz	Verweis
sharpziplib	ICSharpCode	Verwaltung von ZIP Archiven	MIT	(38)
sqlite-net-pcl	Frank A. Krueger	Datenbankfunktionen mit SQLite	MIT	(49)
Newtonsoft.Json	James Newton-King	Parsen von JSON	MIT	(56)
xunit	James Newkirk Brad Wilson	Unittests unter Xamarin	Apache 2.0 und MIT	(55)
Xamarin.Essentials	Microsoft	Einstellungen zwischenspeichern	MIT	(53)
Xamarin.Plugin. FilePicker	Gerald Versluis rafaelrmou vividos	Dateiauswahl	MIT	(70)

6.2. Services

Serviceklassen bieten bei der Entwicklung die Möglichkeit, häufig verwendete Funktionen an einer übersichtlichen Stelle zu verwalten. Dabei werden die Funktionen statisch erstellt. Dadurch muss von der verwendeten Klasse kein Objekt erstellt werden, um die Funktion nutzen zu können. Dies reduziert zudem den belegten Platz auf dem Arbeitsspeicher, da weniger Objekte zwischengespeichert werden müssen. Die zentrale Verwaltung der Funktionen erleichtert zudem die Wartung des Codes.

6.2.1. Sensoren

Ein Vorteil der App ist die Bestimmung von begleitenden Daten (Metadaten) zu den Messdaten. Dies erfolgt durch unterschiedliche Sensoren, welche im Smartphone verbaut sind. In der App werden die folgenden sechs Sensoren unterstützt.

Barometer

Das Barometer misst den Luftdruck in der Umgebung und kann verwendet werden für die Bestimmung von Wetterereignissen, sowie indirekt zur Höhenbestimmung des Standorts über dem Meeresspiegel.

Beschleunigungssensor

Der Beschleunigungssensor misst die Trägheitskraft einer Masse und berechnet daraus die Beschleunigung, mit der sich das Smartphone bewegt.

GPS

Die Lokalisation des Smartphones erfolgt über die Positionierung und Kommunikation mit Satelliten. Dabei gibt es unterschiedliche Systeme von mehreren Staaten, wobei der Begriff GPS nur ein einzelnes System beschreibt, sich der Begriff aber als Synonym verbreitet hat für Satellitennavigation. Das am weitesten verbreitete Positionsbestimmungssystem ist GPS, welches vom US-amerikanischen Militär entwickelt und bereitgestellt wird. Aus Russland stammt das GLONASS System, welches vom russischen Verteidigungsministerium entwickelt wurde. Das chinesische Satellitensystem nennt sich Beidou und befindet sich derzeit noch im Aufbau. Ein weiteres noch im Aufbau befindliches System und als einziges nicht militärisches System wird GALILEO von der Europäischen Union betrieben. Je nach verwendetem Smartphone können unterschiedliche Systeme verwendet werden. Das Betriebssystem auf dem Smartphone liefert der Anwendung Positionsdaten unabhängig von dem verwendeten Satellitensystem.

Gyroskop

Das Gyroskop wird auch Kreiselinstrument genannt und dient zur Bestimmung der Ausrichtung des Smartphones.

Kompass

Der Kompass bestimmt die Richtung in der sich der magnetische Norden der Erde befindet. Dies ist kein eigenständiger Sensor im Smartphone, sondern ergibt sich aus den Werten des Magnetometers. Zur genaueren Bestimmung werden Daten des Gyroskops und Beschleunigungssensors verwendet.

Magnetometer

Mit dem Magnetometer wird das Magnetfeld der Erde gemessen. Es dient dazu, die Ausrichtung des Smartphones zu erfassen.

6.2.2. Helfer

Die Helferklasse enthält Funktionen, die nicht kategorisiert sind oder zu klein sind, um sie mit weiteren Funktionen zu kategorisieren.

Verschlüsselung von Strings

Wie bereits im Kapitel 4.6.2 beschrieben, wird in einigen Bereichen der App Verschlüsselung eingesetzt. Dabei wird ein Text, mit einem Verschlüsselungsalgorithmus in eine Zeichenkette umgewandelt, welche einen Rückschluss auf den ursprünglichen Text verhindert. Es wird hierbei der SHA-512 Algorithmus verwendet. Dieser wurde 2002, als Teil vom SHA-2 Algorithmus, vom amerikanischen National Institute of Standards and Technology (NIST) als Nachfolger für SHA-1 entwickelt. (28)

Die Funktion nimmt einen String entgegen und wandelt diesen mit einer von Microsoft entwickelten Funktion in ein Bytearray aus 128 Hexadezimalzeichen um. Diese wird in ein String umgewandelt und zurückgegeben.

Entpacken von Zip Archiven

Für das Entpacken von Zip Archiven und den Zugriff auf das lokale Dateisystem wurde die Helfer Klasse um diese Funktion erweitert. Damit die App auf das lokale Dateisystem zugreifen kann, muss diese um die Rechte für den Zugriff auf das lokale Datensystem erweitert werden.

6.2.3. Datenbank

Um die Daten des Anwenders zu speichern und Projekte zu verwalten, wird eine lokale Datenbank verwendet. Dabei handelt es sich um eine SQLite Datenbank. Der Aufbau der Datenbank wird in Kapitel 4.8 beschrieben.

CRUD Prinzip

Für den Zugriff und die Verwaltung der Datenbank wurde das CRUD Prinzip angewandt. Die Abkürzung CRUD steht für Create (Erstellen), Read (Zugreifen), Update (Aktualisieren), Delete (Löschen) und beinhaltet die grundlegenden Funktionen, welche im Datenmanagement notwendig sind.

Modelle verwalten

Das verarbeiten von Modellen in der Datenbank erfolgt über definierte Funktionen aus dem SQLite-Net-PCL Paket. Dadurch ist es möglich, Klassen an eine Funktion zu übergeben, welche dann in die Datenbank übertragen werden. Dabei gibt es die Einschränkung, dass nur primitive Datentypen verwendet werden können und keine selbst erstellten Datentypen, sowie keine Listen. Wenn Listen verwendet werden, sollten diese in der Definition mit *[Ignore]* definiert werden und als eigene Klasse für die Datenbank übergeben werden. Daraus ergibt sich, dass es Klassen geben muss, welche die IDs aus beiden Elementen verbinden und selbst auch in die Datenbank übertragen werden müssen.

Dynamische Daten verwalten

Wenn Datensätze dynamisch definiert sind, kann man kein festes Modell dafür vorher definieren. Daher ist es nicht möglich Klassen zu erstellen und diese zu übergeben. Um dem entgegenzuwirken, kann man eigene Queries für die Datenbank definieren, welche die Erstellung von Tabellen, das Einfügen von Daten, sowie das auslesen von Daten definieren.

6.3. Implementierte Funktionen

In dem Prototypen wurden nicht alle im Konzept definierten Funktionen implementiert.

6.3.1. Login

Das Login ist die erste Seite, die der Anwender von der App sieht. Er hat hier die Möglichkeit, sich anzumelden oder zu der Seite zum Erstellen eines neuen Profils weitergeleitet zu werden. Es wird von der Datenbank die Tabelle der vorhandenen Profile geladen und abgeglichen, ob die eingegebenen Daten korrekt sind. Dabei wurde auch die Verschlüsselung bereits implementiert.

6.3.2. Profilerstellung

Wenn der Anwender noch über kein Profil verfügt, kann er ein neues Profil anlegen. Ähnlich wie beim Login wird auch hier die Tabelle der verfügbaren Profile geladen und abgeglichen, ob der Profilname bereits verwendet wurde. Damit wird eine Doppelbelegung der Profilnamen verhindert. Ist der Nutzername nicht vorhanden, wird das neue Profil in die Datenbank eingetragen.

6.3.3. Einstellungen

Die Einstellungen wurden umfassend, wie in Kapitel 4.11 beschrieben, implementiert.

Profilverwaltung

Bei der Profilverwaltung wird in der Datenbank auf die Profiltabelle zugegriffen. Es werden die Informationen zu den einzelnen Profilen dargestellt, sowie die Änderung des Passwortes, sowie das Löschen von Profilen implementiert. Die erfassten Daten von gelöschten Profilen werden in der lokalen Datenbank weiterhin gespeichert.

Sensorverwaltung

Die Aktivierung der Sensoren erfolgt für alle gemeinsam. Das bedeutet, dass der Status des Sensors unabhängig vom Profil erfolgt. Dies wird mit *Preferences* zwischengespeichert, welches Teil des Pakets *Xamarin.Essentials* (siehe Kapitel 6.1) ist. Die Einstellungen sind innerhalb des gesamten Projektes verfügbar.

6.3.4. Sensortest

Bei dem Sensortest werden die implementierten Sensoren ausgelesen und die erfassten Daten dargestellt. Je nachdem welche Sensoren in den Einstellungen aktiviert wurden, werden die Sensoren angesprochen. Ist ein Sensor deaktiviert, so erscheint in dem Fenster bei dem jeweiligen Sensor der Text, dass der Sensor deaktiviert wurde.

6.3.5. Projektverwaltung

In die Projektverwaltung ist die meiste Entwicklungsarbeit geflossen. Dies begründet sich zum einen durch die dynamische Generierung der Nutzeroberfläche, aber auch durch das auftretende Problem in der Datenbank, welches in Kapitel 8.3.2 beschrieben wird.

Projektliste

Bei der Projektliste werden alle vorhandenen Projekte aus der Datenbank abgefragt. Diese werden in einer Liste dargestellt. Über ein Unterfenster können, weitere Details zum ausgewählten Projekt angezeigt werden.

Projekt hinzufügen

Die Projektliste ist zu Beginn noch leer und muss gefüllt werden. Diese Funktion wurde implementiert. Dabei wird ein Zip-Archiv entpackt, die einzelnen Formulare geparst und die gesammelten Daten in eine Datenbank geschrieben.

Formulare generieren

Aus den geparsten Daten, welche in der Datenbank abgelegt werden, werden Formulare generiert und mehrere Formulare in einem Projekt als einzelne Tabs dargestellt. Dabei werden einzelne Formelemente generiert und beim speichern wieder ausgelesen.

Daten bearbeiten

Die erfassten Daten aus der Datenbank können für das einzelne Projekt ausgelesen werden.

6.3.6. Impressum

Im Impressum werden alle an dem Projekt beteiligten Personen aufgelistet.

6.3.7. Unterstützung mehrerer Sprachen

Die App unterstützt mehrere Sprachen. Dabei wird die Systemsprache als zu verwendende Sprache standardmäßig verwendet. Innerhalb der App werden die Texte per Schlüsselwort definiert und sind über die *AppResources* verfügbar. Im Ordner *Localisation* befinden sich Ressource Dateien im XML-Format. Diese beinhalten das Schlüsselwort und den dazugehörigen Text in der jeweiligen Sprache der XML-Datei.

Die Definition der Sprache in den Projekten erfolgt über die Definition in der Webapp Open Data Kit Build. Dort können mehrere Sprachen angegeben werden, welche über unterschiedliche Nummern von als Identifikator voneinander getrennt werden.

Sollte keine Sprache der Systemsprache entsprechen, wird innerhalb der App standardmäßig auf die englische Sprache zurückgegriffen. Bei den Projekten wird standardmäßig auf die Sprache mit der Id 0 zugegriffen.

6.4. Tests

Zur Reduzierung von Bugs im Quellcode wird der Quellcode getestet. Dabei werden unterschiedliche Arten von Tests durchgeführt, bei denen die Funktionalität und Fehlertoleranz überprüft werden.

Unit Test

In Unit Tests werden einzelne Funktionen des Quellcodes auf ihre Funktionalität geprüft. Dabei wird eine Funktion mit unterschiedlichen Übergabeparametern aufgerufen und überprüft, ob der Rückgabewert dem entspricht, was vorher definiert wurde. Für die Tests wurde ein eigenständiges Projekt erstellt. Im Kapitel 8.3.2 wurde darauf eingegangen.

Usability Test

Die Anwendung wurde mehreren Personen, weiterhin als Tester bezeichnet, zum Ausprobieren zur Verfügung gestellt. Dabei handelte es sich um verschiedene Altersgruppen mit unterschiedlichen technischen Erfahrungen. Dadurch wurden Bedienelemente verbessert dargestellt und Fehler in der Programmierung aufgedeckt und anschließend behoben.

7. Validierung und Verifikation

Dieses Kapitel befasst sich mit der Validierung der Anforderungen, welche in Kapitel 4.2 beschrieben wurden. Dabei wird sich nach dem Standard IEEE 610.12 gerichtet (1), welcher die Standards zur Definition von Anforderungen im Bereich des Software Engineerings beschreibt.

7.1. Nicht-funktionale Anforderungen

7.1.1. Generelle Anforderungen

Zu den generellen Anforderungen zählen die Verwendung der App auf mehreren Plattformen, sowie Eigenschaften die für die Messungen und den Anwender relevant sind. In der folgenden Tabelle werden diese Anforderungen aufgelistet und die Eigenschaft verifiziert.

Tabelle 7.1.: Verifikation genereller Anforderungen

Anforderung	Erfüllt	Beschreibung
Endgeräte	ja	Android App vorhanden, iOS Code vorhanden
Betriebssystem	ja	lauffähig unter Android, iOS Code vorhanden
Internetzugriff	nein	App benötigt in derzeitiger Version kein Internet Projekte werden über Browser heruntergeladen
Zuverlässigkeit	ja	
Erweiterbarkeit	ja	weitere Projekte können hinzugefügt werden
Serverkontakt	nein	nicht implementiert
Sensorikzugriff	ja	Sensortest zum Überprüfen der Sensoren
Einsatz von Standards	ja	Standards bei der Gestaltung der App und Standardtemplates für Programmierung verwendet

7.1.2. Anforderungen an die Nutzerfreundlichkeit

Eine einfache Bedienung der App ist die Voraussetzung für die Verwendung dieser. Um dies zu überprüfen, wird die Heuristik nach Nielsen angewandt, was in Kapitel 8 durchgeführt wird.

7.1.3. Anforderungen an die Daten-Sicherheit

Die App soll die Eingaben des Anwenders vor dem Zugriff dritter Parteien schützen. Um dies zu verifizieren, werden bei den verwendeten Erweiterungen, welche in Kapitel 6.1 beschrieben wurden, die Datenschutzbestimmungen überprüft. Dort wird angegeben, ob und in welcher Weise Daten die die Erweiterung erfasst, weitergegeben werden. Ist keine Datenschutzerklärung angegeben, wird die Dokumentation überprüft.

Tabelle 7.2.: Datensicherheit von NuGet Paketen

Paketname	Datenschutzerklärung vorhanden	Weitergabe von Daten
sharpziplib	nein	nein
sqlite-net-pcl	nein	nein
Newtonsoft.Json	nein	nein
xunit	ja	nein
Xamarin.Essentials	ja	nein
Xamarin.Plugin. FilePicker	ja	nein

7.2. Funktionale Anforderungen

7.2.1. Authentifizierung

Der Anwender muss sich authentifizieren, wenn er Messungen durchführen möchte. Dies dient dazu, ein Profil über den Anwender erstellen zu können und verknüpft die erfassten Daten mit dem Profil. Zur Verifikation muss überprüft werden, ob der Anwender ein Profil anlegen und sich mit dem erstellten Profil anmelden kann.

7.2.2. Benutzerkonto

Sollte der Anwender sein eigenes Profil bearbeiten wollen, so kann er dies über die Einstellungen durchführen. Dabei ist ein bereits angelegtes Profil notwendig. In den Einstellungen kann der Anwender seinen Nutzernamen einsehen, sowie sein Passwort ändern. Die Zugriffsrechte beschränken sich auf die Formulare, welche in dem Projekt vorhanden sind und die Sensoren, welche der Anwender ein- und ausschalten kann. Die Abmeldung kann nur bei erfolgreicher Anmeldung über das Hauptmenü ausgeführt werden.

7.2.3. Individuelle App-Funktionen

Die individuellen App-Funktionen wurden zum Teil in die App integriert. So sind Texte, Eingabe- und Auswahlfelder integriert, sowie auch die Erfassung der Position per GPS.

7.2.4. Prüfmodule

Zur Verifikation der Prüfmodule und als zusätzliche Funktion für den Anwender, wurde die Seite *Sensortest* für das Auslesen der Sensoren entwickelt. Dort können die Werte aller aktivierten Sensoren ausgelesen und dargestellt werden. Dabei wird die Geolokation als Sensor aufgefasst.

7.2.5. Basismodule

Die Elemente der Basismodule basieren auf Systemkomponenten. Dabei wird die Systemzeit verwendet, welche über eine Information der Zeitzone verfügt. Auch die Lokalisierung und Ausrichtung basiert auf die im System verbaute Sensorik. Dabei lässt sich diese über das Fenster *Sensortest* verifizieren.

7.2.6. Profiling

Da diese Arbeit das Profiling nicht abdeckt, kann diese Anforderungskategorie nicht validiert und verifiziert werden.

7.2.7. Feldkampagne

Das als Beispielprojekt hinzugefügte Projekt *Feldkampagne* befindet sich auf dem beigelegten Medium zu Masterarbeit. Es handelt sich dabei um die Datei **Feldkampagne.zip**. Das Archiv umfasst die in den Anforderungen beschriebenen Formulare, sowie eine Projektbeschreibung. Dieses Projekt dient auch zur Validierung, indem es von der App geladen wird und als *Aktuelles Projekt* definiert wird. Dadurch werden alle beschriebenen Formulare generiert und können unter dem Menüpunkt *Aktuelles Projekt* aufgerufen und mit Daten gefüllt werden. Die gespeicherten Daten können dann über das Menü *Einstellungen* in eine JSON-Datei überführt und ausgegeben werden.

8. Diskussion

Dieses Kapitel befasst sich mit der Diskussion zur Umsetzung der in der Einleitung aufgestellten These. Es wird eingegangen auf das entwickelte Konzept, den entwickelten Prototypen, die Usability und damit einhergehend die Userexperience, sowie auf aufgetretene Probleme.

8.1. Hypothesen

In der Einleitung wurden mehrere Hypothesen aufgestellt, welche die Entwicklung der App beantworten soll. Dabei ging es unter anderem um die Verbesserung der Messergebnisse, eine modulare Erweiterbarkeit, die Generierung einer modularen Oberfläche, sowie der Erstellung von Profilen und der Einschätzung der Anwender. Dabei ist zu beachten, dass das Profiling von Studenten der Friedrich-Schiller-Universität Jena durchgeführt wurde und daher nicht näher darauf eingegangen wird.

8.1.1. Verbesserung der Qualität der Messwerteingabe

Bei der Entwicklung der App wurden für die Verbesserung der Qualität der Messwerteingabe unterschiedliche Maßnahmen ergriffen. Dazu zählen eine feste Definition von Eingabefeldern und Wertebereichen, d.h. dass der Anwender nur Werte in einem vorher definierten Bereich eingeben kann und z.B. in einem Feld für Zahlen auch nur Zahlen eingeben kann. Um den Anwender bei der Erfassung der Daten auch weiterhin zu Unterstützen gibt es für jedes Eingabefeld einen Hilfstext in dem Beschrieben der Projektersteller beschreiben kann, wie die Daten erfasst werden müssen. Weiterhin unterstützt das Smartphone den Anwender durch die Verwendung der integrierten Sensoren im Gerät.

8.1.2. Modulare Erweiterbarkeit

Die App ist durch weitere Projekte erweiterbar. Diese können mit der Open Data Kit Build Webapp erstellt werden und zusammen mit einer Projektbeschreibung zu einem Zip Archiv zusammengefasst werden. Zudem ist es möglich, weitere Sensoren künftiger Smartphones einfach zu implementieren.

8.1.3. Generierung einer dynamischen Oberfläche

Alle Projekte werden durch Formulare im JSON Format definiert. Die App ist in der Lage diese Dateien auszuwerten, sie in einer internen Datenbank abzulegen und daraus Oberflächen zu generieren. Dabei werden keine neuen Klassen für die Anwendung generiert.

8.2. Usability und User Experience mittels Heuristiken

Zur Einschätzung der Usabilitiy der App werden Heuristiken angewandt, welche von Jakob Nielsen (57) entwickelt wurden. In dem von Jakob Nielsen verfassten Paper wird eine Strategie zur Erfassung und Beschreibung der Oberfläche mittels einer Heuristik beschrieben. Die Erläuterung der einzelnen Bestandteile wird in Kapitel 5.2 beschrieben. Dieser Abschnitt befasst sich mit der Auswertung der App mit Hilfe der Heuristiken.

Sichtbarkeit des Systemstatus

Der Systemstatus wird auf mehreren Arten den Anwender dargestellt. Bestätigungen, sowie auch Fehlermeldungen erscheinen in einem Pop-Up Fenster über der App. Diese beinhalten einen beschreibenden Text, welcher dem Anwender den aktuellen Stand des Systemstatus vermittelt.

Übereinstimmung von System und Wirklichkeit

Die App ist multilingual und kann einfach über eine XML Datei erweitert werden. Als Standardsprache wird Englisch definiert. Sollte auf dem Smartphone eine Sprache als Systemsprache ausgewählt sein, welche die App nicht unterstützt, so wird Englisch verwendet. Daneben wurde auch eine deutsche Sprachversion implementiert.

Nutzerkontrolle und Freiheit

Der Anwender kann von Hauptfenstern jederzeit über das Hauptmenü zu anderen Menüunterpunkten wechseln. Befindet er sich in detaillierteren Unterfenstern, so hat er jederzeit die Möglichkeit, über einen Zurück-Button in das vorherige Fenster zu wechseln.

Beständigkeit und Standards

Durch das verwenden von Systemelementen, sowie dem Systemdesignes werden die Designrichtlinien des jeweiligen Systems eingehalten. Die Erstellung von Formularen durch die Open Data Kit Build Webapp ermöglicht es dem Anwender beim Erstellen von Projekten bereits bestehendes Wissen vom Open Data Kit wiederverwenden zu können.

Fehlervermeidung

Um Fehler bei der Eingabe des Anwenders zu verhindern, werden Eingaben durch Slider oder Einschränkungen bei der Eingabe vorgenommen. Dadurch werden Minimum- und Maximumwerte vorgegeben, sodass eine fehlerhafte Eingabe, etwa durch Eingabe von Text in einem Zahlenfeld verhindert werden kann.

Bei der Auswahl von Projekten aus Zip Archiven werden Fehler dadurch verhindert, dass bei der Auswahl der Datei darauf geachtet wird, dass es sich um ein Zip Archiv handelt. Sollte ein falsches Zip Archiv geöffnet werden, so wird dem Anwender dies mittels einer Fehlermeldung dargestellt.

Wiedererkennung statt Erinnerung

Der Aufbau der App ist über die gesamte App einheitlich. Zudem lehnt sich das Design an das Systemdesign an, wodurch sich der Anwender leicht zurechtfinden kann.

Flexibilität und Effizienz

Die Bedienung wird nicht unterteilt in Anfänger und Fortgeschrittenen-Modus. Auch gibt es keine Schnellverweise auf andere Fenster. Dies ist auf Grund der Anwendung und das Fehlen einer Tastatur auch nicht notwendig.

Ästhetisches und minimalistisches Design

Das Design verwendet die vom Betriebssystem bereitgestellten Elemente in der Systemfarbe. Dadurch ist der Anwender bereits vertraut mit der Menüführung und den verwendeten Elementen, wie Buttons, Menüs und Untermenüs. Aufgrund der Nähe zum Systemdesign wird der Anwender nicht abgelenkt und

Hilfestellung beim Erkennen, Bewerten und Beheben von Fehlern

Die Anwendung ist so aufgebaut, dass die Eingabe so definiert wird, dass möglichst wenige Fehler auftreten. Dabei wird z.B. für die Eingabe von Zahlenwerten in definierten Bereichen ein Slider verwendet. Dadurch wird vermieden, dass der Anwender eine Fehleingabe macht. Fehler können über einen Editor behoben werden, der in der App integriert ist.

Hilfe und Dokumentation

Zu jeder Eingabe wurde ein Hilfstext definiert. Dadurch wird sichergestellt, dass der Anwender weiß, wie die Messungen durchgeführt werden müssen.

8.3. Aufgetretene Probleme

Während der Entwicklung der App sind unterschiedliche Schwierigkeiten aufgetreten.

8.3.1. Konzept

Bei der Entwicklung eines Konzepts für die Anwendung sind unterschiedliche Herausforderungen aufgetreten. Diese werden in dem Unterkapitel beschrieben.

Zugriffsrechte auf Profile und Datenbank

Auf dem Smartphone können sich mehrere Anwender anmelden. Dabei ist das Konzept, dass auf dem Smartphone jeder Anwender die gleichen Zugriffsrechte innerhalb der App hat. Dies ist kritisch für verschiedene Funktionen. Die kritischen Funktionen sind das Löschen der gesamten Datenbank, sowie die Verwaltung der Profile. In dem Prototypen wurden die Rechte so vergeben, dass jeder Anwender die gleichen Rechte hat und Projekte hinzufügen kann, sowie auch Profile verändern oder die ganze Datenbank löschen. Hintergrund dafür ist die Notwendigkeit, Passwörter von Profilen zu verändern, sollte ein Anwender sein Passwort vergessen haben. Dies könnte auch einem Administratorprofil in der App passieren. In dem Fall gäbe es keine Möglichkeit, für den Anwender Einstellungen an der App und den Profilen vorzunehmen. Es wird davon ausgegangen, dass das Smartphone, schon wegen der Vielzahl an privaten Information auf dem Gerät, nur an Personen weitergegeben wird, dem der Besitzer des Smartphones vertraut. Daher basiert die App mit seinem Mehrnutzersystem auf das Vertrauen unter den Personen, die das Gerät verwenden.

8.3.2. Programmierung

Neben dem Entwurf der App als Konzept wurde die Entwicklung des Prototyps durch unterschiedliche Umstände erschwert. Darunter fallen fehlende Funktionen innerhalb des verwendeten Frameworks, aber auch Einschänkungen bei der Auswahl der verwendeten Entwicklungshardware.

Parsen von Json Dateien

Das Parsen von Json erwies sich aufgrund von mehreren Unterschiedlichen möglichen Datentypen als problematisch. Aus diesem Grund wurde die folgende Frage bei StackOverflow gestellt (14). Bei dem Datentyp handelt es sich um ein Objekt, welches aus Schlüsseln und Werten besteht. Dabei können die folgenden Datentypen verwendet werden:

- KeyValuePair
- Directory
- IEnumeration
- ICollection

Es stellte sich heraus, dass die Verwendung des Datentyps Directory gewählt werden muss.

Unterstützung von SQLite unter Xamarin

Zur Speicherung von Daten wurde die lokale Datenbank SQLite mit implementiert. Diese beinhaltet einen ausreichenden Funktionsumfang zur Verwaltung der Messdaten. Das Problem liegt in der fehlenden Unterstützung von Foreign Keys durch das Xamarin Framework. So ist es bei entwickelten Modellen nicht möglich, Listen und eigene Klassen in der Datenbank durch Systemfunktionen abzubilden.

Das Problem wurde durch eine eigenständige Implementierung in der Datenbank-Klasse der App behoben. Es wurden Funktionen implementiert, welche dazu dienen, Tabellen unabhängig von Modellen zu erstellen und zu verwalten, sowie Abhängigkeiten zwischen Tabellen zu berücksichtigen.

Dynamische Verwaltung von Daten in SQLite

Das verwendete Paket für die lokale Datenbank SQLite Net PCL verfügt über keine Funktionalität zum Auslesen von Datensätzen aus der Datenbank ohne vorherige Definition einer Klasse. Es ist lediglich möglich, einzelne skalare Felder der Tabelle auszulesen, anstatt zeilenweise vorgehen zu können. Daher musste eine Funktion implementiert werden, mit der jedes einzelne Feld ausgelesen wird und in jeweiligen Listen gespeichert wird.

Rechtemanagement unter Android

Das Rechtemanagement unter Android verhindert, dass Dateien aus dem Downloadordner des Nutzers verwendet werden können. Es ist jedoch möglich die Dateien in den Programmordner der App zu kopieren und dort zu verwenden. Dazu müssen im Android Manifest die Rechte *READ_EXTERNAL_STORAGE* und *WRITE_EXTERNAL_STORAGE* gesetzt werden. Dies wird vor allem benötit, wenn ein Projekt als ZIP Archiv auf das Smartphone heruntergeladen wurde und sich im Downloadordner befindet.

Entwicklung für Apple iOS

Die Entwicklung der App konzentriert sich auf das Android Betriebssystem von Google. Der Quellcode ist unter Android, sowie auch unter iOS von Apple ausführbar. Für die Entwicklung für das iOS Betriebssystem wird ein Computer von Apple, sowie ein iPhone von Apple vorausgesetzt, welche zusätzlich über Entwicklerlizenzen verfügen. Da dies nicht gegeben war, wurden keine iOS spezifischen Anpassungen durchgeführt.

Erstellung von Unit Tests

Zur Erstellung von Unit Tests für das Xamarin-Framework gibt es nur sehr wenig Dokumentation. Entsprechend traten bei der Entwicklung von Tests Fehler auf. So war ursprünglich eine Integration der Tests in das Hauptprojekt angedacht. Dies führte zu Fehlern auf Grund des verwendeten .Net-Frameworks, welche als Bibliothek keine Tests ausführbar macht. Aus diesem Grund wurde ein zusätzliches Projekt innerhalb der Projektmappe erstellt. Zudem kann dadurch der Umfang der App verringert werden, da nun keine Tests in der gepackten App vorhanden sind. Bei der Problembehebung fand Unterstützung durch den Anwender *zivkan* auf StackOverflow (15) statt.

9. Fazit

Die Entwicklung eines Prototypen mit dem Xamarin Framework von Microsoft hat gezeigt, dass es möglich ist, eine modulare, plattformunabhängige Anwendung zur Erfassung von Messwerten zu entwickeln. Dabei wurde diese App von Grund auf konzeptioniert und prototypisch umgesetzt. Bei der Entwicklung wurde zudem Wert auf die Usability und die User Experience gelegt. Der Fokus bei der Entwicklung lag auf einer genaueren und schnelleren Datenerfassung und einer anwenderfreundlichen Bedienung.

Zu Beginn der Arbeit wurden im Kapitel 2 Grundlagen, Begriffe und verwendete Programme beschrieben. Zudem wurde auf den Datenschutz eingegangen, welcher durch die Erfassung personenbezogener Daten einen hohen Stellenwert hat. Eine entsprechende Datenschutzerklärung wurde im Prototyp implementiert und befindet sich im Anhang der Masterarbeit. Im Anschluss daran wurden unterschiedliche Frameworks miteinander darauf verglichen, wie sie am effektivsten zur Erfüllung der Anforderungen beitragen können. Dabei erwies sich das Xamarin Framework von Microsoft als das effektivste Werkzeug für die Entwicklung. Es sticht durch Plattformunabhängigkeit, kostenfreie Verwendung, sowie guter Dokumentation hervor.

Der zweite Abschnitt der Arbeit befasst sich im Kapitel 3 mit dem Stand der Forschung und Entwicklung. Es gibt bereits mehrere Frameworks, welche sich auf die Datenerfassung spezialisiert haben. Diese wurden analysiert und miteinander verglichen. Ziel war es ein Framework auf einem Server zu installieren und es mit der Anwendung für Smartphone zu kombinieren, sodass es als Backend zur Auswertung der Daten verwendet werden kann. Dabei waren offener Quellcode, sowie geringe Kosten entscheidende Faktoren. Das Open Data Kit ging aus dem Vergleich als optimales Backend-Framework hervor, da es allen Anforderungen entsprach und Möglichkeiten zur Erweiterbarkeit bot. Zudem wurden zwei Anwendungen für Smartphones betrachtet, welche die Sensoren von Smartphones erfassen und dokumentieren können. Dadurch lassen sich eigene Experimente erstellen. Das Kapitel umfasst zudem Studien, welche die Erfassung von Messdaten mittels Smartphones mit der Erfassung auf Papier gegenüberstellen. Sie kommen zu dem Schluss, dass die Erfassung per Smartphones günstiger, schneller und genauer ist, als mit Papier. Zudem wurden Studien aus dem Bereich Usability und User Experience betrachtet und auch mögliche Anwendungsfälle.

Der Hauptteil der Arbeit befindet sich im Kapitel 4. *Konzept*. Es wird dort auf die wesentlichen Strukturen der App eingegangen und gezeigt, dass es möglich ist eine App zu entwickeln, welche von mehreren Personen auf dem Smartphone verwendet werden kann und auf mehreren Plattformen lauffähig ist. Zudem ist durch die gegebene Modularität der App möglich, dynamisch mehrere Projekte als Modul in die App einzubinden und zu verwalten. Das Kapitel 5 Usability und User Experience (UX) ist ein weiteres wichtiges Kapitel, welches einen größeren Fokus auf den Anwender lenken soll. Es lässt sich aus dem Kapitel herleiten, dass die Entwicklung für den Anwender dafür sorge trägt, dass dieser die App mit größerem Engagement verwendet. Zudem wurde auf die unterschiedlichen Kommunikationsprotokolle eingegangen.

Im folgenden Kapitel folgt die Umsetzung als Prototyp. Es wird auf die Klassenstruktur eingegangen, sowie die Umsetzung von Konzepten beschrieben. Auch werden Implementierungen vom Design, sowie Programmstrukturen beschrieben.

Die Verifikation und Validation des Prototyps gegenüber den Anforderungen wurde in Kapitel 7 durchgeführt. Es wird auf die einzelnen Anforderungen des Anforderungskatalogs eingegangen und

inwieweit diese durch den Prototypen abgedeckt sind.

Es findet im Kapitel 8. *Diskussion* eine Auswertung des Konzept, Designs, sowie des Prototypen statt. Dabei wird auch auf Probleme eingegangen, welche im laufe der Entwicklung aufgetreten sind. Dabei wurde festgestellt, dass unter anderem die Datenbankanbindung zu größeren Schwierigkeiten führte, aufgrund fehlender funktionaler Anbindung durch das verwendete Xamarin Framework.

Die These der Arbeit war die Fragestellung nach der plattformunabhängigen, modularen Anwendung, welche die Erfassung von Messdaten optimieren und beschleunigen soll. Dies ist durch die Verwendung der integrierten Sensoren in Smartphones möglich. Eine zusätzliche Digitalisierung der erhobenen Daten kann durch den Export des Messwerte in der App zu einer zusätzliche Zeitersparnis und besseren Messergebnissen führen. Die Entwicklung der App hat gezeigt, dass es möglich ist für mehrere Plattformen zu entwickeln. Durch die Verwendung von ODK Build ist es möglich, die App in mehreren unterschiedlichen Forschungsprojekten einzusetzen.

Die Masterarbeit hat den Schwerpunkt in der ingenieursmäßigen Entwicklung einer mobilen Anwendung. Dabei liegt der Fokus auf der Erfüllung der gesetzten Anforderungen durch den Anforderungskatalog. Es ist die erste App, welche Daten auf mehreren Plattformen erfassen kann. Zudem ist es mit der App möglich mehrere Projekte auf dem Smartphone oder Tablet zu verwalten.

Da im Rahmen der Masterarbeit der Prototyp nicht alle Anforderungen des Anforderungskatalogs erfüllen kann, kann dieser um weitere Anforderungen erweitert werden. Unter anderem kann der Prototyp um eine Kameraanbindung erweitert werden, aber auch eine Einbindung von freien Karten, welche den Anwender nicht zurückverfolgen, ist denkbar. Dabei ließe sich OpenStreetMap als möglicher freier Kartenanbieter verwenden. Zudem muss des Profiling, welches von Studenten an der Friedrich Schiller Universität Jena entwickelt wurde, noch in den Prototypen integriert werden.

A. Datenschutzerklärung

Die folgende Datenschutzerklärung stützt sich stark auf die Datenschutzerklärung, die das DLR auf seinem Internetauftritt zur Kenntnis gibt. Da diese Erklärung in die vom DLR beauftragte App eingebunden wird, werden in Absprache mit dem Betreuer des DLRs die Formulierungen zur Gewährleistung der Rechtssicherheit stellenweise übernommen. (26).

Informationen zum Datenschutz

Das DLR nimmt den Schutz personenbezogener Daten sehr ernst. Wir möchten, dass Sie wissen, wann wir welche Daten speichern und wie wir sie verwenden. Als eingetragener Verein des deutschen Bürgerlichen Rechts unterliegen wir den Bestimmungen der EU-Datenschutzgrundverordnung (DSGVO) und des Bundesdatenschutzgesetzes (BDSG) sowie des Telemediengesetzes (TMG). Wir haben technische und organisatorische Maßnahmen getroffen, die sicherstellen, dass die Vorschriften über den Datenschutz sowohl von uns als auch von externen Dienstleistern beachtet werden.

I. Name und Anschrift des Verantwortlichen

Der Verantwortliche im Sinne der Datenschutz-Grundverordnung und anderer nationaler Datenschutzgesetze der Mitgliedsstaaten sowie sonstiger datenschutzrechtlicher Bestimmungen ist die:

Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)
Linder Höhe
51147 Köln

Telefon: +49 2203 601-0
E-Mail: datenschutz@dlr.de
WWW: https://www.dlr.de

II. Begriffsbestimmungen

In Übereinstimmung mit der Datenschutzgrundverordnung und dem Bundesdatenschutzgesetz verwenden wir in dieser Datenschutzerklärung unter anderem die folgenden Begriffe:

1. Personenbezogene Daten

Personenbezogene Daten sind alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person (im Folgenden „betroffene Person“) beziehen. Als identifizierbar wird eine natürliche Person angesehen, die direkt oder indirekt, insbesondere mittels Zuordnung zu einer Kennung wie einem Namen, zu einer Kennnummer, zu Standortdaten, zu einer Online-Kennung oder zu einem oder mehreren besonderen Merkmalen, die Ausdruck der physischen, physiologischen, genetischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität dieser natürlichen Person sind, identifiziert werden kann.

2. Betroffene Person

Betroffene Person ist jede identifizierte oder identifizierbare natürliche Person, deren personenbezogene Daten von dem für die Verarbeitung Verantwortlichen verarbeitet werden.

3. Verarbeitung

Verarbeitung ist jeder mit oder ohne Hilfe automatisierter Verfahren ausgeführte Vorgang oder jede solche Vorgangsreihe im Zusammenhang mit personenbezogenen Daten wie das Erheben, das Erfassen, die Organisation, das Ordnen, die Speicherung, die Anpassung oder Veränderung, das Auslesen, das Abfragen, die Verwendung, die Offenlegung durch Übermittlung, Verbreitung oder eine andere Form der Bereitstellung, den Abgleich oder die Verknüpfung, die Einschränkung, das Löschen oder die Vernichtung.

4. Einschränkung der Verarbeitung

Einschränkung der Verarbeitung ist die Markierung gespeicherter personenbezogener Daten mit dem Ziel, ihre künftige Verarbeitung einzuschränken.

5. Profiling

Profiling ist jede Art der automatisierten Verarbeitung personenbezogener Daten, die darin besteht, dass diese personenbezogenen Daten verwendet werden, um bestimmte persönliche Aspekte, die sich auf eine natürliche Person beziehen, zu bewerten, insbesondere, um Aspekte bezüglich Arbeitsleistung, wirtschaftlicher Lage, Gesundheit, persönlicher Vorlieben, Interessen, Zuverlässigkeit, Verhalten, Aufenthaltsort oder Ortswechsel dieser natürlichen Person zu analysieren oder vorherzusagen.

6. Pseudonymisierung

Pseudonymisierung ist die Verarbeitung personenbezogener Daten in einer Weise, auf welche die personenbezogenen Daten ohne Hinzuziehung zusätzlicher Informationen nicht mehr einer spezifischen betroffenen Person zugeordnet werden können, sofern diese zusätzlichen Informationen gesondert aufbewahrt werden und technischen und organisatorischen Maßnahmen unterliegen, die gewährleisten, dass die personenbezogenen Daten nicht einer identifizierten oder identifizierbaren natürlichen Person zugewiesen werden.

7. Verantwortlicher oder für die Verarbeitung Verantwortlicher

Verantwortlicher oder für die Verarbeitung Verantwortlicher ist die natürliche oder juristische Person, Behörde, Einrichtung oder andere Stelle, die allein oder gemeinsam mit anderen über die Zwecke und Mittel der Verarbeitung von personenbezogenen Daten entscheidet. Sind die Zwecke und Mittel dieser Verarbeitung durch das Unionsrecht oder das Recht der Mitgliedstaaten vorgegeben, so kann der Verantwortliche beziehungsweise können die bestimmten Kriterien seiner Benennung nach dem Unionsrecht oder dem Recht der Mitgliedstaaten vorgesehen werden.

8. Auftragsverarbeiter

Auftragsverarbeiter ist eine natürliche oder juristische Person, Behörde, Einrichtung oder andere Stelle, die personenbezogene Daten im Auftrag des Verantwortlichen verarbeitet.

9. Empfänger

Empfänger ist eine natürliche oder juristische Person, Behörde, Einrichtung oder andere Stelle, der personenbezogene Daten offengelegt werden, unabhängig davon, ob es sich bei ihr um einen Dritten handelt oder nicht. Behörden, die im Rahmen eines bestimmten Untersuchungsauftrags nach dem Unionsrecht oder dem Recht der Mitgliedstaaten möglicherweise personenbezogene Daten erhalten, gelten jedoch nicht als Empfänger.

10. Dritter

Dritter ist eine natürliche oder juristische Person, Behörde, Einrichtung oder andere Stelle außer der betroffenen Person, dem Verantwortlichen, dem Auftragsverarbeiter und den Personen, die unter der unmittelbaren Verantwortung des Verantwortlichen oder des Auftragsverarbeiters befugt sind, die personenbezogenen Daten zu verarbeiten.

11. Einwilligung

Einwilligung ist jede von der betroffenen Person freiwillig für den bestimmten Fall in informierter Weise und unmissverständlich abgegebene Willensbekundung in Form einer Erklärung oder einer sonstigen eindeutigen bestätigenden Handlung, mit der die betroffene Person zu verstehen gibt, dass sie mit der Verarbeitung der sie betreffenden personenbezogenen Daten einverstanden ist.

III. Allgemeines zur Datenverarbeitung

1. Umfang der Verarbeitung personenbezogener Daten

Wir verarbeiten personenbezogene Daten unserer Nutzer grundsätzlich nur, soweit dies zur Bereitstellung einer funktionsfähigen Website sowie unserer Inhalte und Leistungen erforderlich ist. Die Verarbeitung personenbezogener Daten unserer Nutzer erfolgt regelmäßig nur nach Einwilligung des Nutzers. Eine Ausnahme gilt in solchen Fällen, in denen eine vorherige Einholung einer Einwilligung aus tatsächlichen Gründen nicht möglich ist und die Verarbeitung der Daten durch gesetzliche Vorschriften gestattet ist.

2. Rechtsgrundlage für die Verarbeitung personenbezogener Daten

Soweit wir für Verarbeitungsvorgänge personenbezogener Daten eine Einwilligung der betroffenen Person einholen, dient Art. 6 Abs. 1 lit. a EU-Datenschutzgrundverordnung (DSGVO) als Rechtsgrundlage.

Bei der Verarbeitung von personenbezogenen Daten, die zur Erfüllung eines Vertrages, dessen Vertragspartei die betroffene Person ist, erforderlich ist, dient Art. 6 Abs. 1 lit. b DSGVO als Rechtsgrundlage. Dies gilt auch für Verarbeitungsvorgänge, die zur Durchführung vorvertraglicher Maßnahmen erforderlich sind.

Soweit eine Verarbeitung personenbezogener Daten zur Erfüllung einer rechtlichen Verpflichtung erforderlich ist, der dem DLR unterliegt, dient Art. 6 Abs. 1 lit. c DSGVO als Rechtsgrundlage.

Für den Fall, dass lebenswichtige Interessen der betroffenen Person oder einer anderen natürlichen Person eine Verarbeitung personenbezogener Daten erforderlich machen, dient Art. 6 Abs. 1 lit. d DSGVO als Rechtsgrundlage.

Ist die Verarbeitung zur Wahrung eines berechtigten Interesses des DLR oder eines Dritten erforderlich und überwiegen die Interessen, Grundrechte und Grundfreiheiten des Betroffenen das erstgenannte Interesse nicht, so dient Art. 6 Abs. 1 lit. f DSGVO als Rechtsgrundlage für die Verarbeitung.

3. Datenlöschung und Speicherdauer

Die personenbezogenen Daten der betroffenen Person werden gelöscht oder gesperrt, sobald der Zweck der Speicherung entfällt. Eine Speicherung kann darüber hinaus erfolgen, wenn dies durch den europäischen oder nationalen Gesetzgeber in unionsrechtlichen Verordnungen, Gesetzen oder sonstigen Vorschriften, denen der Verantwortliche unterliegt, vorgesehen wurde. Eine Sperrung oder Löschung der Daten erfolgt auch dann, wenn eine durch die genannten Normen vorgeschriebene Speicherfrist abläuft, es sei denn, dass eine Erforderlichkeit zur weiteren Speicherung der Daten für einen Vertragsabschluss oder eine Vertragserfüllung besteht.

B. Beispiel JSON

Listing B.1: Phänologisches Stadium Beispiel

```
{"title": "Entwicklungsstadium",
    "controls": [{
            "name": "phenologicalStadium",
            "label": {"0": "Phaenologisches Stadium"},
            "hint": {},
            "defaultValue": "0",
            "readOnly": false,
            "required": false,
            "requiredText": {},
            "relevance": "",
            "constraint": "",
            "invalidText": {},
            "calculate": "",
            "options": [{
                    "text": {"0": "Keimung"},
                    "cascade": [],
                    "val": "1"
                },{
                    "text": {"0": "weitere Stadien"},
                    "cascade": [],
                    "val": "2"
                }
            ],
            "cascading": false,
            "other": false,
            "appearance": "Default",
            "metadata": {},
            "type": "inputSelectOne"
        }
    ],"metadata": {
        "version": 2,
        "activeLanguages": {
            "0": "German",
            "_counter": 0,
            "_display": "0"
        },
        "optionsPresets": [],
        "htitle": null,
        "instance_name": "",
        "public_key": "",
        "submission_url": ""
    }
}
```

Glossar

API Bei einer API handelt es sich um eine Schnittstelle (Application Programming Interface bzw. Schnittstelle zur Anwendungsprogrammierung), die es einem Programm erlaubt mit einem bestehenden Softwaresystem in Verbindung zu treten und dabei dessen Ressourcen mit zu benutzen. (21).

App Die Abkürzung *App* leitet sich aus dem englischen Wort Application her. Dabei handelt es sich um eine Anwendung, welche auf einem mobilen Endgerät ausgeführt wird.

BN Abkürzung für Benutzer.

BSI Bundesamt für Sicherheit in der Informationstechnik.

CRUD Das Akronym CRUD definiert sich aus den vier grundlegenden Anwendungen, welche bei der Verwendung von Datenbanken durchgeführt werden. Diese stehen für **C**reate (Erstellen), **R**ead (Lesen), **U**pdate (Bearbeiten) und **D**elete (Löschen). (29).

DB Abkürzung für Datenbank.

DLR Das Deutsche Zentrum für Luft- und Raumfahrt e.V. ist eine deutsche Forschungseinrichtung. Diese hat folgende Themengebiete: Luft- und Raumfahrt, Energietechnik, Verkehr, Sicherheit, angewandte und Grundlagenforschung, sowie Datenwissenschaften.

IDE Unter einer IDE versteht man eine Entwicklungsumgebung, welche dem Softwareentwickler bei der Programmierung von Software unterstützt. (*engl.: integrated development environment*).

JNI Das Java Native Interface dient als Schnittstelle in Java für den Zugriff auf plattformspezifische Ressourcen..

JSON Mithilfe der JavaScript Object Notation, kurz JSON, lassen sich Daten menschen- und maschinenlesbar speichern und übertragen. Einen ähnlichen Ansatz verfolgen beispielsweise auch YAML und XML. (24).

Parser Parser analysieren Zeichenketten und bereiten den Quelltext auf. Auf diese Weise wandeln sie den Code in eine Repräsentation um, mit der andere Programme wie Compiler oder HTML-Renderer arbeiten können. (23).

REST REST steht für REpresentational State Transfer, API für Application Programming Interface. Gemeint ist damit ein Programmierschnittstelle, die sich an den Paradigmen und Verhalten des World Wide Web (WWW) orientiert und einen Ansatz für die Kommunikation zwischen Client und Server in Netzwerken beschreibt. (20).

UI Mithilfe des User Interface (abgekürzt UI) kann eine Person eine Soft- oder Hardware kontrollieren. Das UI stellt so eine Schnittstelle zwischen Mensch und Maschine dar, bei dem ein Set von Befehlen oder Menüs es dem Anwendungsnutzer ermöglicht, mit einem Programm zu kommunizieren. Das Ziel eines UI ist es, die Arbeit zu erleichtern. (33).

Unit Test Ein Unit Test ist ein automatisierter Modultest, welcher die Lauffähigkeit von Komponenten überprüft. Dabei wird der Test mehrfach mit unterschiedlichen Parametern durchgeführt.

URL Als URL bzw. Uniform Resource Locator wird eine definierte Adresse bezeichnet, welche auf die Position einer Datei auf einem Server angezeigt wird und von diesem abgerufen wird. (32).

Usability Der Begriff **Usability** bedeutet ins deutsch übersetzt Bedienbarkeit. Usability bezeichnet das Ausmaß, in dem ein Produkt, System oder Dienst durch bestimmte Benutzer in einem bestimmten Anwendungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen. (29).

UX **User Experience (UX)** erweitert den Begriff Usability um ästhetische und emotionale Faktoren wie eine ansprechende, begehrenswerte Gestaltung, Aspekte der Vertrauensbildung oder Spaß bei der Nutzung (Joy of use). Dieser ganzheitliche Ansatz umfasst das gesamte Nutzungserlebnis, welches man bei der Verwendung eines Produktes erfährt. Die Nutzer sollen nicht nur schnell und reibungslos zum Ziel kommen, sondern - abhängig vom Anwendungsbereich - auch positive Gefühle wie Spaß oder Freude bei der Benutzung erleben. (29).

Webapp Eine Webapp ist eine Anwendung, welche über einen Browser aufgerufen wird. Dabei handelt es sich um eine Webseite, welche die Funktionalität einer Anwendung beinhaltet.

XML XML (Extensible Markup Language) ist ein text-basiertes Format für den Austausch strukturierter Information. Dies können Dokumente, Konfigurationen, Bücher, Rechnungen und vieles mehr sein. Es ist abgeleitet vom älteren Standard SGML (ISO 8879) für die Anwendung im Web. (64).

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Die vorliegende Arbeit hat in dieser oder einer ähnlichen Form noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

Literaturverzeichnis

[1] *IEEE Standard Glossary of Software Engineering Terminology, 610.12-1990*

[2] Bundesamt für Sicherheit in der Informationstechnik (BSI): Kryptographische Verfahren: Empfehlungen und Schlussellangen, Stand: 2019-01. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.html. – Forschungsbericht. – Zugegriffen am 25.09.2019 16:58

[3] *Magpi*. https://home.magpi.com/. – Zugegriffen am 01.04.19 08:12

[4] *ohmage*. http://ohmage.org/. – Zugegriffen am 01.04.19 08:30

[5] *XLSForm*. W3C. http://xlsform.org/en/. – Zugegriffen am 01.04.19 13:13

[6] 2. Physikalischen Institut der RWTH Aachen University: *phyphox*. https://phyphox.org/de/home-de/. – Zugegriffen am: 18.03.19 12:30

[7] Agriya: *AngularJS is No More – The Future of TypeScript and Angular 2?* https://www.agriya.com/blog/angularjs-is-no-more-the-future-of-typescript-and-angular-2/. – Zugegriffen am 06.05.2019 14:52

[8] Akers, A. : *v0.1.0: Initial public release*. https://github.com/facebook/react-native/releases?after=v0.2.1. Version: 2015. – Zugegriffen am 30.04.2019 11:31

[9] Amadeo, R. : *Google's Dart language on Android aims for Java-free, 120 FPS apps*. ArsTechnica. https://arstechnica.com/gadgets/2015/05/googles-dart-language-on-android-aims-for-java-free-120-fps-apps/. – Zugegriffen am 26.03.19 12:50

[10] Angermeier, D. G.: *Agiles Projektmanagement*. https://www.projektmagazin.de/glossarterm/agiles-projektmanagement. – Zugegriffen am: 25.03.19 10:30

[11] App Entwickler Verzeichnis: *Web-App Entwicklung mit HTML5: Der Große Praxis-Guide*. https://app-entwickler-verzeichnis.de/app-news/13-programmierung-a-developer-tools/582-web-app-entwicklung-mit-html5-der-grosse-praxis-guide. – Zugegriffen am 09.04.19 09:51

[12] Apple: *iOS Design Themes*. https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/. – Zugegriffen am 24.04.2019 09:35

[13] Bar, A. : *What Web Can Do Today*. https://whatwebcando.today/. – Zugegriffen am 09.04.19 09:57

[14] betaros: *Deserializing Json to list of objects in C# with Newtonsoft*. StackOverflow. https://stackoverflow.com/questions/57508015/deserializing-json-to-list-of-objects-in-c-sharp-with-newtonsoft. – Zugegriffen am 15.08.2019 13:45

[15] BETAROS, Z. : *Xunit test fails with NU1701*. StackOverflow. https://stackoverflow.com/questions/56991293/xunit-test-fails-with-nu1701/57041080. – Zugegriffen am 16.07.2019 10:45

[16] BORG, E. : *DEMMIN*. https://www.dlr.de/eoc/de/desktopdefault.aspx/tabid-9539/16442_read-40235/. – Zugegriffen am: 25.03.19 10:12

[17] BORG, E. : *Kalibrations- und Validationseinrichtung DEMMIN*. https://www.dlr.de/eoc/de/desktopdefault.aspx/tabid-5395/10255_read-40097/. – Zugegriffen am 12.04.2019 12:40

[18] BUDIU, R. ; PERNICE, K. : *Mobile First Is NOT Mobile Only*. https://www.nngroup.com/articles/mobile-first-not-mobile-only/. Version: 2016. – Zugegriffen am 30.04.2019 11:07

[19] CLEVERISM: *Definition of Swift*. https://www.cleverism.com/skills-and-tools/swift/. – zuletzt besucht am 11.03.19 14:17

[20] CLOUD COMPUTING INSIDER - DIRK SROCKE: *Was ist eine REST API?* https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/. – Zugegriffen am 25.09.2019 18:53

[21] CODER WELTEN: *Ein kleines Glossar*. http://www.coder-welten.de/glossar/anwendungen.htm. – Zugegriffen am 30.08.2019 10:35

[22] CORPORATION, M. : *SQLite EF Core Database Provider Limitations*. https://docs.microsoft.com/en-us/ef/core/providers/sqlite/limitations. – zugegeriffen am 19.06.2019 13:31

[23] DEV-INSIDER: *Was ist ein Parser?* https://www.dev-insider.de/was-ist-ein-parser-a-756662/. – Zugegriffen am 30.08.2019 11:02

[24] DEV-INSIDER: *Was ist JSON*. https://www.dev-insider.de/was-ist-json-a-702243/. – Zugegriffen am 30.08.2019 11:05

[25] DLR E.V.: *Information about Data Protection*. https://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-11049/1890_read-27253/. – Zugegriffen am 04.04.19 13:13

[26] DLR E.V.: *Informationen zum Datenschutz*. https://www.dlr.de/dlr/desktopdefault.aspx/tabid-11049/1890_read-27253/. – Zugegriffen am 04.04.19 13:13

[27] DOBILITY INC.: *SurveyCTO*. https://www.surveycto.com/. – Zugegriffen am 01.04.19 08:21

[28] FEDERAL REGISTER: *Announcing Approval of Federal Information Processing Standard (FIPS) Publication 180-4, Secure Hash Standard (SHS) a Revision of FIPS 180-3*. https://www.federalregister.gov/documents/2012/03/06/2012-5400/announcing-approval-of-federal-information-processing-standard-fips-publication-180-4-secure- Version: 2012. – Zugegriffen am 17.07.2019

[29] GMBH & CO. KG usability.de: *Definition von Usability und UX. Usability vs. User Experience*. https://www.usability.de/usability-user-experience.html. – Zugegeriffen am 05.07.2019 11:21

[30] GOOGLE LLC: *Android Studio*. https://developer.android.com/studio. – Zugegriffen am: 11.03.19 10:41

[31] GOOGLE LLC: *Design for Android*. https://developer.android.com/design/index.html. – Zugegriffen am 24.04.2019 09:33

[32] Gründerszene: *Uniform Resource Locator (URL).* https://www.gruenderszene.de/lexikon/begriffe/uniform-resource-locator-url?interstitial_click. – Zugegriffen am 25.09.2019 19:02

[33] Gründerszene: *User Interface.* https://www.gruenderszene.de/lexikon/begriffe/user-interface?interstitial. – Zugegriffen am 12.08.2019 12:54

[34] Hansen, T. ; 3rd, D. E. E.: *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF).* RFC 6234. http://dx.doi.org/10.17487/RFC6234. Version: Mai 2011 (Request for Comments)

[35] Harvard Humanitarian Initiative: *KoBoToolbox.* https://www.kobotoolbox.org/. – Zugegriffen am 01.04.19 08:38

[36] Higgins, C. ; O'Hare, P. : *COBWEB - Citizen Observatory Web.* https://cobwebproject.eu. – Zugegriffen am: 01.04.19 08:03

[37] Icaza, M. de: *Announcing Xamarin.* https://tirania.org/blog/archive/2011/May-16.html. – Zugegriffen am 26.03.19 10:20

[38] ICSharpCode: *Github.* https://github.com/icsharpcode/SharpZipLib. – Zugegriffen am 17.06.2019 12:45

[39] Imperial College London: *EpiCollect.* http://www.epicollect.net/. – Zugegriffen am 01.04.19 14:02

[40] Imperial College London: *EpiCollect5.* https://five.epicollect.net/. – Zugegriffen am 01.04.19 08:42

[41] ISO 9241-11:2018: Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts / International Organization for Standardization. Version: März 2018. https://www.iso.org/standard/63500.html. Geneva, CH, März 2018. – Forschungsbericht. – Zugegriffen am 15.04.2019 14:26

[42] ISO 9241-210:2010: Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems / International Organization for Standardization. Version: März 2010. https://www.iso.org/standard/52075.html. Geneva, CH, März 2010. – Forschungsbericht. – Zugegriffen am 15.04.2019 14:28

[43] Jemerov, D. : *Hello World.* https://blog.jetbrains.com/kotlin/2011/07/hello-world-2/. – Zugegriffen am: 11.03.19 13:52

[44] Jens Jacobsen, L. M.: *Praxisbuch Usability und UX.* Rheinwerk Verlag GmbH https://www.ebook.de/de/product/28238120/jens_jacobsen_lorena_meyer_praxisbuch_usability_und_ux.html. – ISBN 3836244233

[45] Jetbrains s.r.o.: *IntelliJ IDEA.* https://www.jetbrains.com/idea/. – Zugegriffen am: 11.03.19 10:43

[46] John M. Boyer, IBM: *XForms 1.1.* W3C. https://www.w3.org/TR/xforms11/. Version: 2009. – Zugegriffen am 01.04.19 13:07

[47] Kalenda, F. : *Microsoft schluckt Cloud-Entwicklerplattform Xamarin.* ZDNet. https://www.zdnet.de/88261158/microsoft-schluckt-cloud-entwicklerplattform-xamarin/. – Zugegriffen am: 26.03.19 10:24

[48] Kriegisch, A. : *Scrum - auf einer Seite erklärt.* https://scrum-master.de/Was_ist_Scrum/Scrum_auf_einer_Seite_erklaert. – Zugegriffen am: 25.03.19 11:05

[49] Krueger, F. A.: *Github*. https://github.com/praeclarum/sqlite-net. – Zugegriffen am 13.06.2019 14:54

[50] Microsoft Corporation: *Das Model-View-ViewModel-Muster*. https://docs.microsoft.com/de-de/xamarin/xamarin-forms/enterprise-application-patterns/mvvm. – Zugegriffen am 17.07.2019 17:22

[51] Microsoft Corporation: *Eine Einführung in NuGet*. https://docs.microsoft.com/de-de/nuget/what-is-nuget. – Zugegriffen am 04.04.19 15:47

[52] Microsoft Corporation: *Lifecycle FAQ—Device operating systems*. https://support.microsoft.com/en-us/help/18403/lifecycle-faq-device-operating-systems. – Zugegriffen am: 11.03.19 09:46

[53] Microsoft Corporation: *Xamarin.Essentials*. https://docs.microsoft.com/de-de/xamarin/essentials/?context=xamarin/android. – Zugegriffen am 04.04.19 15:44

[54] Mono Project: *Mono Releases*. https://www.mono-project.com/docs/about-mono/releases/. – Zugegriffen am 26.03.19 10:39

[55] Newkirk, J. ; Wilson, B. : *Github*. https://github.com/xunit/xunit. – zugegeriffen am 20.06.2019 11:28

[56] Newton-King, J. : *Json.NET*. https://www.newtonsoft.com/json. – Zugegriffen am 13.06.2019 14:58

[57] Nielsen, J. : Enhancing the explanatory power of usability heuristics. In: *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence*, ACM Press, 1994

[58] ODK Community: *ODK Build*. https://build.opendatakit.org/. – Zugegriffen am 12.08.2019 14:20

[59] ODK Community: *Open Data Kit*. https://opendatakit.org/. – Zugegriffen am 18.03.19 12:34

[60] ODK Community: *Welcome to ODK's Docs!* https://docs.opendatakit.org/. – Zugegriffen am 10.04.19 10:36

[61] ODK Community: *Welcome to Open Data Kit 2's documentation!* https://docs.opendatakit.org/odk-x/. – Zugegriffen am 10.04.19 10:36

[62] Roberts, J. : *Creating Your Own Custom Dynamic C# Classes*. http://dontcodetired.com/blog/post/Creating-Your-Own-Custom-Dynamic-C-Classes. Version: 2016. – Zugegriffen am 23.08.2019

[63] Russell, A. : *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/. Version: 2015. – Zugegriffen am: 30.04.2019 11:02

[64] SelfHTML: *XML*. https://wiki.selfhtml.org/wiki/XML. – Zugegriffen am 25.09.2019 19:04

[65] Starke, G. : *Effektive Softwarearchitekturen*. Hanser Fachbuchverlag https://www.ebook.de/de/product/30408382/gernot_starke_effektive_softwarearchitekturen.html. – ISBN 3446452079

[66] StatCounter: *Marktanteile der führenden mobilen Betriebssysteme an der Internetnutzung mit Mobiltelefonen in Deutschland von Januar 2009 bis Januar 2019.* https://de.statista.com/statistik/daten/studie/184332/umfrage/marktanteil-der-mobilen-betriebssysteme-in-deutschland-seit-2009/. – Zugegriffen am: 11.03.19 08:06

[67] Steinberg, M. D.: Software solutions for form-based collection of data and the semantic enrichment of form data. . – arXiv:1901.11053

[68] Truckenbrodt, S. : Weiterentwicklung des Monitorings landwirtschaftlicher Nutzflächen durch die geländebasierte Erfassung und Modellierung von biophysikalischen Parametern unter Einbeziehung von 'Local Knowledge' zu Bewirtschaftungsmaßnahmen und externen Beobachtungen zur phänologischen Entwicklung von Feldfrüchten. https://www.geographie.uni-jena.de/Professuren/Fernerkundung/Team/Truckenbrodt_S/Projekte.html. – Zugegriffen am: 06.10.2019 13:39

[69] Truckenbrodt, S. ; Klan, F. ; Pathe, C. ; Borg, E. ; Missling, K.-D. : *Citizen Science App - Nutzeranforderungsdokument.* 2019. – CSApp_URD_v.03_2019-03-07.docx

[70] Versluis, G. ; rafaelrmou ; vividos: *Github.* https://github.com/jfversluis/FilePicker-Plugin-for-Xamarin-and-Windows. – Zugegriffen am 20.06.2019 11:46

[71] Wered Software: *Google Play Store - Sensors Multitool.* https://play.google.com/store/apps/details?id=com.wered.sensorsmultitool. – Zugegriffen am 24.05.2019 um 12:55

[72] Williams, R. : *Apple iOS: a brief history.* https://www.telegraph.co.uk/technology/apple/11068420/Apple-iOS-a-brief-history.html. – Zugegriffen am 11.03.19 13:58

[73] Zhang, S. ; Wu, Q. ; Velthoven, M. H. ; Chen, L. ; Car, J. ; Rudan, I. ; Zhang, Y. ; Li, Y. ; Scherpbier, R. W.: Smartphone Versus Pen-and-Paper Data Collection of Infant Feeding Practices in Rural China. In: *Journal of Medical Internet Research* 14 (2012), sep, Nr. 5, S. e119. http://dx.doi.org/10.2196/jmir.2183. – DOI 10.2196/jmir.2183