HOCHSCHULE MITTWEIDA
University of Applied Sciences

MASTERARBEIT

Frau
Lea Jennifer Bunzel

AMBER-Alert Deutschland: Konzeptioneller Entwurf und prototypische Implementierung einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern

2022

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERARBEIT

AMBER-Alert Deutschland: Konzeptioneller Entwurf und prototypische Implementierung einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern

Autorin:
Lea Jennifer Bunzel

Studiengang:
Cybercrime / Cybersecurity

Seminargruppe:
CY20wC-M

Erstprüfer:
Prof. Dr. rer. nat. Dirk Labudde

Zweitprüfer:
M. Sc. Dipl.-Inf. (FH) Knut Altroggen

Mittweida, Oktober 2022

Faculty of **Applied Coumputer Sciences and Biosciences**

MASTER THESIS

AMBER-Alert Germany: Conceptual design and prototypical implementation of an Android application for alerting the population of missing children

Author:
Lea Jennifer Bunzel

Study Programme:
Cybercrime / Cybersecurity

Seminar Group:
CY20wC-M

First Referee:
Prof. Dr. rer. nat. Dirk Labudde

Second Referee:
M. Sc. Dipl.-Inf. (FH) Knut Altroggen

Mittweida, October 2022

Bibliografische Angaben

Bunzel, Lea Jennifer: AMBER-Alert Deutschland: Konzeptioneller Entwurf und prototypische Implementierung einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern, 103 Seiten, 39 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Masterarbeit, 2022

Referat

Die vorliegende Masterarbeit befasst sich mit der Entwicklung einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland. Dabei richtet sich der Fokus zuerst auf das aus den USA stammende AMBER-Alert-System – ein System zur Suche vermisster Kinder – und im weiteren Verlauf auf den aktuellen Stand eines solchen Systems innerhalb Deutschlands. Bisher haben sich nur wenige Arbeiten mit der Umsetzung einer solchen App auseinandergesetzt. Aus diesem Grund liegt der Schwerpunkt dieser Arbeit auf der Implementierung einer prototypischen App zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland. Diesbezüglich werden bereits existierende Applikationen betrachtet und daraus ein konzeptioneller Entwurf entwickelt. Dieser Entwurf dient als Grundlage für die prototypische Implementierung der App.
Die vorliegende Arbeit stellt eine effektive Möglichkeit dar, um einen Großteil der Bevölkerung in Deutschland zur schnellen Suche und sicheren Bergung bei vermissten Kindern zu erreichen.

Bibliographic Information

Bunzel, Lea Jennifer: AMBER-Alert Germany: Conceptual design and prototypical implementation of an Android application for alerting the population of missing children, 103 pages, 39 figures, Hochschule Mittweida, University of Applied Sciences, Faculty of Applied Coumputer Sciences and Biosciences

Master Thesis, 2022

Abstract

This master thesis deals with the development of an Android application for alerting the population of missing children in Germany. First, the focus is on the AMBER Alert System – a system for searching missing children – which originated in the USA, and then on the current status of such a system in Germany. So far, only a few papers have dealt with the implementation of such an app. For this reason, the focus of this thesis is on the implementation of a prototypical app for alerting the population of missing children in Germany. In this regard, existing apps are considered and a conceptual design is developed based on them. This design is the basis for the prototypical implementation of the app.
This thesis represents an effective way to reach a large part of the population in Germany for fast search and safe recovery of missing children.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AMBER ..... America's Missing: Broadcast Emergency Response

API ..... Application Programming Interface

EAS ..... Emergency Alert System

IDE ..... Integrierte Entwicklungsumgebung

IPAWS ..... Integrated Public Alert and Warning System

JDK ..... Java Development Kit

JSON ..... JavaScript Object Notation

NGO ..... Nichtregierungsorganisation

SDK ..... Software Development Kit

WEA ..... Wireless Emergency Alert

XML ..... Extensible Markup Language

1 Einleitung

1.1 Motivation

„[I]nsgesamt rund 1.600 ungeklärte Fälle vermisster Kinder“ [1].
Dieser Satz stammt von der Internetseite des Bundeskriminalamtes und belegt, dass am 15. März 2022 noch immer tausende Kinder seit dem frühesten registrierten Vermisstendatum am 14. Februar 1957 vermisst wurden. „Mehr als die Hälfte dieser Kinder sind unbegleitete Flüchtlinge [...], gehören zu den sogenannten Dauerausreißern/Streunern oder wurden ihren Sorgeberechtigten entzogen. [...] Bei dem verbleibenden Teil der vermissten Kinder ist zu befürchten, dass diese Opfer einer Straftat oder [eines] Unglücksfalls wurden, sich in einer Situation der Hilflosigkeit befinden oder nicht mehr am Leben sind.“ [1]
In den Jahren 2018 bis 2021 wurden durchschnittlich 15.700 Kinder (0 bis 13 Jahre) in Deutschland als vermisst gemeldet. Obwohl die jährliche Aufklärungsquote bei ca. 97 % liegt, bleiben trotzdem jedes Jahr mehrere hundert Fälle von vermissten Kindern ungeklärt. Im gleichen Zeitraum wurden ca. 78.700 Jugendliche (14 bis 17 Jahre) als vermisst registriert. Auch hier werden jährlich ca. 3 % der Fälle nicht aufgeklärt. [1]
Um Fälle von vermissten Kindern und Jugendlichen aufklären zu können, ist eine umfangreiche Alarmierung der Bevölkerung nötig. Dadurch kann die Öffentlichkeit bei der Suche nach den vermissten Minderjährigen mithelfen.
Das Vorbild, wie eine solche umfangreiche Alarmierung aussehen kann, stammt aus den USA. Dabei handelt es sich um das sog. AMBER-Alert-System, über welches die Bevölkerung durch verschiedene Verbreitungswege bei vermissten Kindern informiert wird. So können beispielsweise Radio- und Fernsehprogramme unterbrochen sowie Informationen auf digitalen Autobahnschildern angezeigt werden. [2] [3]
Ebenso können Meldungen an Mobiltelefone gesendet werden [4], was heutzutage einen wichtigen Verbreitungsweg darstellt. Allein in Deutschland besitzen über 97 % der Haushalte ein Mobiltelefon (Stand: 01. Januar 2021) [5], weshalb durch diese eine Vielzahl an Personen über vermisste Kinder und Jugendliche informiert werden könnte.

1.2 Problemstellung

Während das AMBER-Alert-System aus den USA bereits seit 1996 vom Staat betrieben wird [6], ist das in Deutschland bestehende System Teil der Nichtregierungsorganisation (NGO) „Initiative Vermisste Kinder“ [7]. Aus diesem Grund existieren hierzulande nicht die gleichen Möglichkeiten zur Alarmierung der Bevölkerung, wie sie in den USA bestehen. Zwar hat die Initiative Kooperationspartner, welche ihr bei der Verbreitung von Vermisstenmeldungen helfen, jedoch fehlen ihr auch entscheidende Verbreitungsmöglichkeiten. So ist es beispielsweise zum aktuellen Zeitpunkt (Stand: September 2022) nicht möglich, die Bevölkerung mit einem einheitlichen System über ihre Mobiltelefone zu alarmieren.

1.3 Zielsetzung

Das Ziel der vorliegenden Masterarbeit besteht darin, einen konzeptionellen Entwurf für eine Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern und Jugendlichen auszuarbeiten und prototypisch zu implementieren. Zur Vereinfachung wird im Folgenden teilweise nur von einer Alarmierung der Bevölkerung bei vermissten Kindern geschrieben, jedoch dient die App ebenso zur Suche nach vermissten Jugendlichen.
Zuerst wird in Kapitel 2 erläutert, was der AMBER-Alert ist und wie dieser zur Suche vermisster Kinder beiträgt. Weiterhin werden die wichtigsten Grundlagen zum Entwickeln einer Android-Applikation aufgezeigt.
Im 3. Kapitel wird ein mögliches Konzept für eine Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland entworfen. Hierfür wird zuerst die Frage beantwortet, warum eine solche Applikation sinnvoll ist. Anschließend wird der IST-Zustand einer entsprechenden App in Deutschland analysiert sowie der gewünschte SOLL-Zustand definiert.
Kapitel 4 beschäftigt sich mit der Umsetzung des Entwurfs. Dabei wird auf das Setup von Android Studio und Firebase sowie auf die Implementierung der App-Ressourcen und Java-Klassen eingegangen. Außerdem folgt ein kurzer Einblick in die Testphase der App.
Weiterhin wird in Kapitel 5 die Anleitung zum Anlegen, Bearbeiten und Löschen eines Falles dargestellt.
Im abschließenden Kapitel 6 werden die vorherigen Kapitel noch einmal zusammengefasst und die im Rahmen der Arbeit entwickelte Applikation mit schon existierenden Apps verglichen. Ebenso wird auf die Einbindung von Vermisstenfällen in bereits bekannte Warn-Apps und das sog. Cell Broadcast eingegangen. Zuletzt wird ein Fazit gezogen sowie ein Ausblick auf die Weiterarbeit mit der entwickelten App gegeben.

2 Grundlagen

In diesem Kapitel folgt eine Übersicht über die wichtigsten Grundlagen zum Entwickeln einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland. Hierfür wird zu Beginn auf den AMBER-Alert eingegangen, ein aus den USA stammendes Informationssystem und wie dieses in europäischen Ländern verbreitet ist. Im Zuge dessen wird der aktuelle Stand eines solchen Systems in Deutschland analysiert.
Weiterhin werden die verschiedenen Arten von Apps und die Grundlagen zur allgemeinen Entwicklung einer Android-Applikation erklärt. Dabei wird beispielsweise auf die Plattform Android generell sowie benötigte Software, App-Komponenten und weitere Dateien und Plattformen eingegangen.
Abschließend folgt ein kurzer Exkurs zu den benötigten Programmier- und Auszeichnungssprachen Java und XML sowie dem Datenaustauschformat JSON.

2.1 AMBER-Alert

Das AMBER-Alert-System, auch nur AMBER-Alert genannt, ist ein aus den USA stammendes Informationssystem zur Alarmierung der Bevölkerung bei vermissten bzw. entführten Kindern. Das System soll dazu beitragen, vermisste Kinder schnell und sicher zu bergen, indem ein Großteil der Bevölkerung über das Verschwinden informiert wird. Eingeführt wurde der AMBER-Alert 1996 nach der Entführung der neunjährigen Namensgeberin Amber Hagerman und steht für „*America's Missing: Broadcast Emergency Response*“[1]. [6]
Neben den USA wird das System in mehr als 30 weiteren Ländern weltweit verwendet [2]. Aus diesem Grund wird zuerst auf das amerikanische System unter Einbezug der Aktivierungskriterien und Verbreitungswege eines AMBER-Alerts sowie Erfolgsstatistiken eingegangen. Anschließend wird die Verwendung von AMBER-Alert-Systemen in anderen Ländern einschließlich Deutschland analysiert.

2.1.1 America's Missing: Broadcast Emergency Response

In den USA ist AMBER eine freiwillige Kooperation zwischen Strafverfolgungsbehörden, Rundfunkeinrichtungen sowie Verkehrsbehörden und wurde zur schnellen Suche und sicheren Bergung eines vermissten bzw. entführten Kindes eingeführt [8].
Wie wichtig das schnelle Auffinden eines Kindes ist, zeigt eine Studie aus dem Jahr 2006, in welcher über 800 Kindesentführungsmorde aus den Jahren 1968 bis 2002 untersucht wurden. Die Studie umfasste ein Untersuchungsgebiet von 44 US-amerikani-

[1] dt. Amerikas Vermisste: Rundfunk-Notfallmaßnahmen

schen Staaten und in 735 Fällen waren die Opfer jünger als 18 Jahre. [9, S. 6 ff.]
Ein Ergebnis dieser Studie ist in Tabelle 2.1 dargestellt. Dabei wird deutlich, dass die Zeit bei einer Kindesentführung eine sehr große Rolle spielt. Wurde ein entführtes Kind getötet, geschah dies in 76,2 % der Fälle innerhalb der ersten drei Stunden nach der Entführung. Innerhalb der ersten 24 Stunden waren es bereits 88,5 %. Aus diesem Grund ist es besonders wichtig, dass eine Vielzahl an Menschen bei einer Kindesentführung erreicht werden und somit zur schnellen Suche des entführten Kindes beitragen können [10, S. 3]. [9, S. 13 f.]

Tabelle 2.1: Häufigkeitsangabe, innerhalb welcher Zeitspanne ein entführtes Kind getötet wurde, wenn das Kind im Rahmen der Entführung getötet wurde (adaptiert nach [9, S. 14])

Zeitspanne	**kumulierte Häufigkeit in %**
< 1 Stunde	46,8
innerhalb von 3 Stunden	76,2
innerhalb von 24 Stunden	88,5
innerhalb von 7 Tagen	97,9
innerhalb von 30 Tagen	100

Das AMBER-Alert-System wird zum einen in allen 50 Bundesstaaten der USA verwendet und zum anderen innerhalb der folgenden Gebiete:

- Washington, D.C.,
- Puerto Rico,
- den amerikanischen Jungferninseln,
- Indian Country und
- den nördlichen und südlichen Grenzgebieten. [2]

Hierdurch besitzt das amerikanische AMBER-Alert-System insgesamt 86 AMBER-Alert-Pläne. Diese Pläne umfassen die jeweiligen Aktivierungskriterien und Verbreitungswege eines AMBER-Alerts. [2]

Aktivierungskriterien eines AMBER-Alerts

Zur Vermeidung übermäßig vieler AMBER-Alerts und der daraus resultierenden Desensibilisierung der Bevölkerung hat das US-amerikanische Justizministerium Kriterien zur Aktivierung eines AMBER-Alerts empfohlen. Diese Kriterien sind für die einzelnen Staaten freiwillig, werden jedoch von den meisten in ähnlicher Weise umgesetzt. Die empfohlenen Aktivierungskriterien lauten wie folgt:

- Es besteht der begründete Verdacht, dass eine Entführung stattgefunden hat,
- es wird davon ausgegangen, dass dem Kind eine schwere Körperverletzung oder der Tod droht,
- es liegen genügend Informationen über das Kind vor, welche das Erscheinungsbild des Kindes beschreiben,
- das Kind ist noch keine 18 Jahre alt und
- der Name des Kindes und weitere kritische Daten, einschließlich der Kennzeichnung, dass es sich um eine Kindesentführung handelt, wurden in das *National Crime Information Center*[2] eingetragen. [2] [11]

Verbreitungswege eines AMBER-Alerts

Treffen die Aktivierungskriterien zu und von den Strafverfolgungsbehörden wird eine Meldung über ein vermisstes Kind abgesetzt, wird diese Meldung über das sog. *Integrated Public Alert and Warning System*[3] (IPAWS) verbreitet. Das IPAWS ist ein Zusammenschluss aus dem *Emergency Alert System*[4] (EAS), der *National Oceanic and Atmospheric Administration*[5], den *Wireless Emergency Alerts*[6] (WEA) sowie weiteren internetbasierten Diensten und individuellen Alarmsystemen. [8]
Zum EAS gehören Radio-, Fernseh-, Kabel- und Satellitenanbieter, wodurch eine Notfallmeldung aktuelle Radio- und Fernsehprogramme unterbrechen kann. Mithilfe von WEAs können Notfallmeldungen an WEA-fähige Mobilgeräte gesendet und ohne Installieren einer speziellen Applikation empfangen werden. Sowohl das EAS als auch WEAs sind geografisch abhängig, weshalb Notfallmeldungen nur in einem lokal definierten Bereich gesendet werden. [3] [4]
Weiterhin können die Meldungen über Verkehrsbehörden auf Autobahnschildern angezeigt sowie über zusätzliche Partner (z. B. Lotterien) verbreitet werden. [2]
Neben den Warnmeldungen über das IPAWS, die Verkehrsbehörden und Lotterien gibt es zusätzlich sekundäre Verbreitungswege. Wird ein AMBER-Alert ausgelöst, leitet das *National Center for Missing and Exploited Children*[7] die Meldungen an die sekundären Partner weiter. Zu diesen zählen verschiedene Bundesbehörden (z. B. Einwanderungs- und Zollbehörden, FBI), Internetdienstanbieter, Soziale Medien, ausgewählte Hotels, digitale Anzeigetafelbetreiber und weitere individuelle Alarmsysteme. [12] [13] [14]
In Abbildung 2.1 sind die primären (mittig) und sekundären (rechts) Verbreitungswege nach Ausgabe eines AMBER-Alerts durch US-amerikanische Strafverfolgungsbehörden dargestellt.

[2] US-Datenbank zur Sammlung von Informationen im Zusammenhang mit der Kriminalitätsbekämpfung
[3] dt. integriertes öffentliches Alarm- und Warnsystem
[4] dt. Notfall-Alarmsystem
[5] dt. Nationale Ozean- und Atmosphärenbehörde
[6] dt. Drahtlose Notfallwarnung
[7] dt. Nationales Zentrum für vermisste und ausgebeutete Kinder

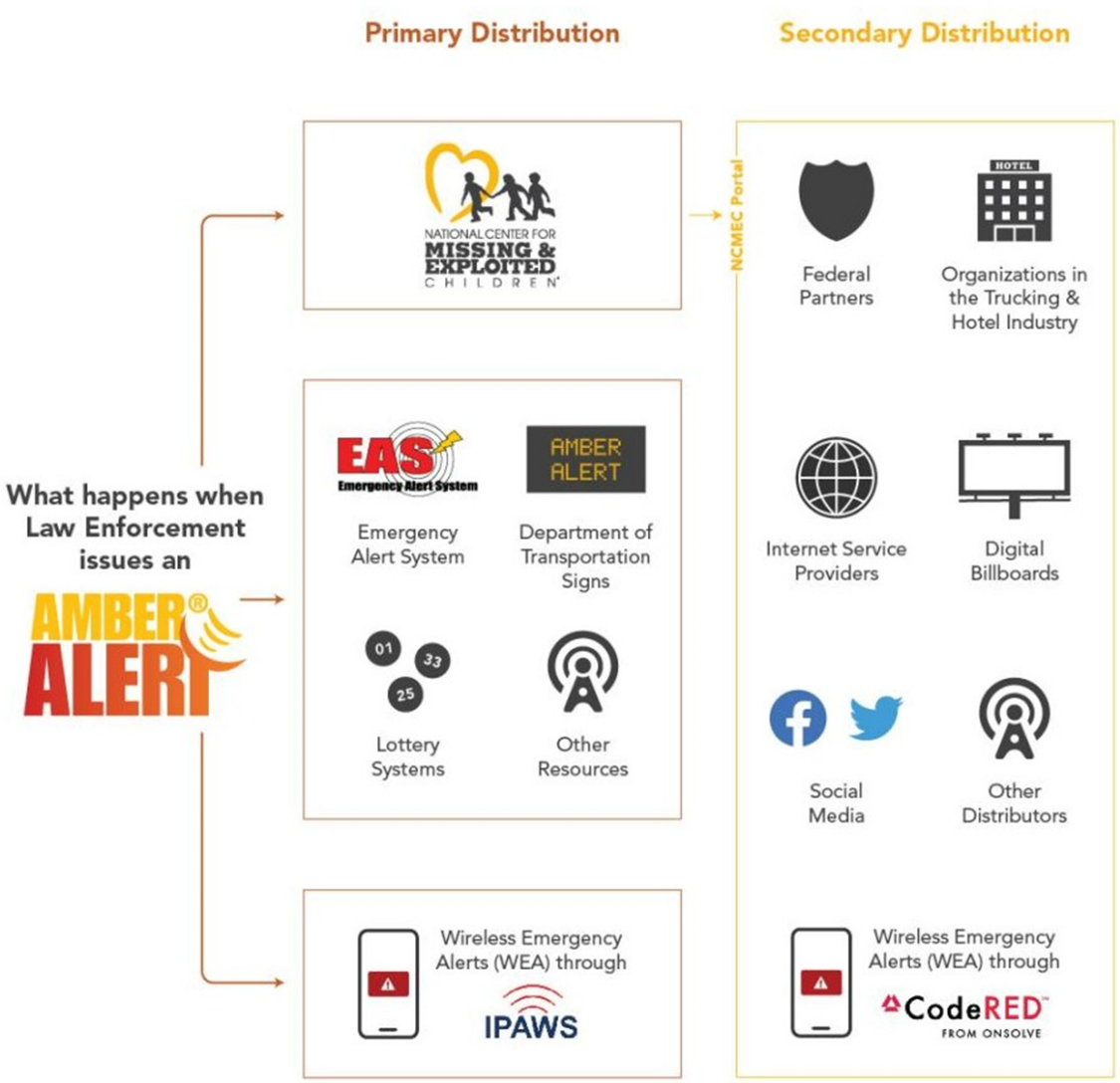


Abbildung 2.1: Darstellung der primären und sekundären Verbreitungswege bei Ausgabe eines AMBER-Alerts durch eine US-amerikanische Strafverfolgungsbehörde [14]

AMBER-Alert Statistik

Im Jahr 2020 wurden in den USA 200 AMBER-Alerts mit 260 vermissten Kindern ausgegeben. Bis zum 23. Februar 2021 wurden die Kinder aus 196 Fällen aufgefunden. Während es sich bei zehn Fällen um Falschmeldungen handelte, zwölf Fälle unbegründet waren und zehn Kinder tot aufgefunden wurden, sind 46 erfolgreiche Rettungen als direkte Folge von AMBER-Alerts zu verzeichnen. [15, S. 8]
Insgesamt wurden bis zum 31. Dezember 2021 in den USA 1.111 Kinder durch das AMBER-Alert-System erfolgreich aufgefunden. 120 dieser Kinder wurden mithilfe von WEAs gerettet. [16]

2.1.2 AMBER-Alert in Europa

Im Jahr 2008 haben die Niederlande ihr eigenes AMBER-Alert-System gestartet, mit welchem in 94 % der Fälle ein vermisstes Kind erfolgreich gefunden wurde. Dieser Erfolg führte dazu, dass 2013 die Stiftung *AMBER Alert Europe* gegründet wurde. Die Stiftung soll dazu beitragen, vermisste bzw. entführte Kinder in ganz Europa lebend wiederzufinden. Dabei werden Strafverfolgungsbehörden, die Öffentlichkeit und NGOs aus mehreren europäischen Ländern miteinander vernetzt, um durch Schulungen, Präventions- und Sensibilisierungsmaßnahmen sowie die Verbesserung von bereits bestehenden AMBER-Alert-Systemen „*Zero Missing Kids*“[8] in Europa zu erreichen. [17] [18] [19]
Die Aktivierungskriterien eines AMBER-Alerts sind dabei in allen Ländern unterschiedlich. So wird beispielsweise in Frankreich nur dann ein AMBER-Alert ausgelöst, wenn es sich nachweislich um eine Entführung handelt. Ist das Kind verschwunden, aber nicht entführt worden (z. B. sog. Ausreißer oder das Kind hat sich verlaufen) und es befindet sich möglicherweise in (Lebens-)Gefahr, ist das trotzdem kein Kriterium, um einen AMBER-Alert auszulösen. [20] In Luxemburg hingegen wird bei jedem vermissten Kind ein AMBER-Alert verbreitet, wenn es sich in Lebensgefahr befindet bzw. befinden könnte [21].
Ebenso unterscheiden sich die Verbreitungswege eines AMBER-Alerts in den verschiedenen Ländern. Während in den meisten Ländern Radio- und Fernsehsender, E-Mails und SMS-Nachrichten über vermisste Kinder informieren, gibt es ebenso Länder, welche zusätzlich beispielsweise Callcenter, Banken oder eigene Apps einsetzen. [22]
Europäische Länder mit einem eigenen AMBER-Alert-System sind (Stand: 11. April 2022): Belgien, Bulgarien, Deutschland, Frankreich, Griechenland, Großbritannien, Irland, Italien, Litauen, Luxemburg, Malta, die Niederlande, Polen, Portugal, Rumänien, die Schweiz, die Slowakei, Spanien, die Tschechische Republik und Zypern. [22]

2.1.3 AMBER-Alert in Deutschland

Im Gegenzug zu den USA wird das deutsche AMBER-Alert-System, das sog. AKUT-Alarmierungssystem, nicht von staatlichen Organisationen, sondern einer NGO betrieben, der „Initiative Vermisste Kinder“ [7]. Die Initiative arbeitet für Angehörige von vermissten Kindern und unterstützt diese bei der Suche nach den Vermissten [23]. Dies erfolgt mithilfe verschiedener Kooperationspartner, beispielsweise Ströer[9], welcher digitale Werbeflächen zur Verbreitung von Vermisstenmeldungen zur Verfügung stellt. Weiterhin ist die NGO Teil von *AMBER Alert Europe* (siehe Abschnitt 2.1.2), wodurch eine grenzüberschreitende Suche unterstützt wird. [24]

[8] dt. null vermisste Kinder
[9] `https://www.stroeer.de/`

Zur Aktivierung eines deutschen AMBER-Alerts (sog. AKUT-Alarmierung) wird zu Beginn das Risiko des vermissten Kindes bestimmt. Sind folgende Kriterien und somit die höchste Risikostufe erfüllt, wird der Fall über das AKUT-Alarmierungssystem verbreitet:

- Das Kind ist jünger als 18 Jahre,
- wurde (vermutlich) entführt,
- befindet sich in einer lebensbedrohlichen Lage und
- wird durch die Veröffentlichung von Informationen keinem (zusätzlichen) Risiko ausgesetzt. [7]

Neben diesen sog. AKUT-Warnmeldungen gibt es noch zwei weitere Risiko-Kategorien: *vermisste Kinder* und *gefährdete vermisste Kinder*. Das Kriterium für ein gefährdetes vermisstes Kind ist, dass ein „unmittelbares signifikantes Gefährdungsrisiko für das vermisste Kind“ besteht [7]. Jedoch dürfen nicht die oben genannten Kriterien einer AKUT-Warnmeldung erfüllt sein. Die Fälle von gefährdeten vermissten Kindern werden ebenso über das AKUT-Alarmierungssystem verbreitet. [7]
In Abbildung 2.2 werden die Risikogruppen zur Bestimmung der Verbreitung von Kinder-Vermisstenfällen über das AKUT-Alarmierungssystem bestimmt.

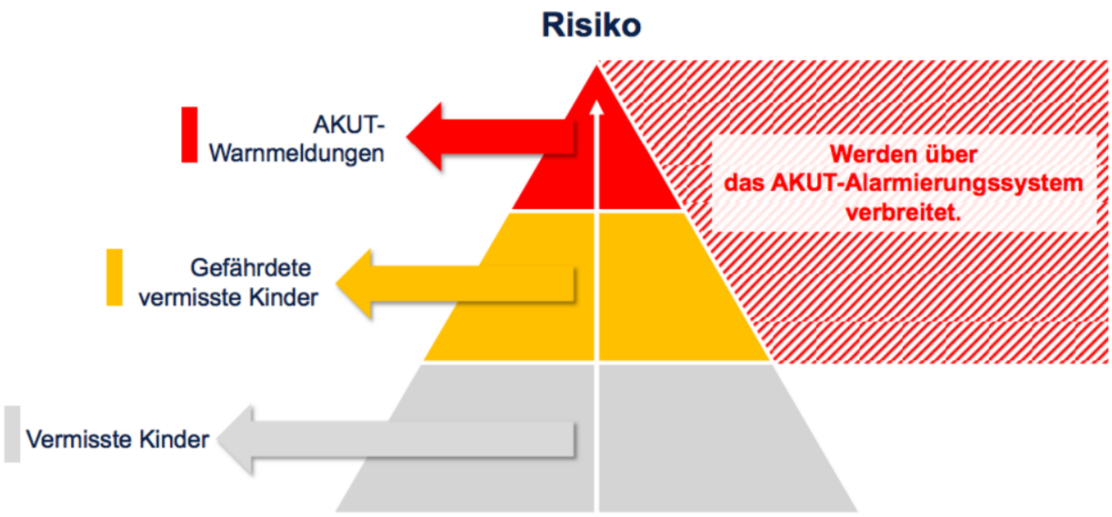


Abbildung 2.2: Darstellung der Risikogruppen zur Bestimmung der Verbreitung von Kinder-Vermisstenfällen über das AKUT-Alarmierungssystem [7]

Auf den aktuellen Zustand (IST-Zustand) einer Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland wird im nächsten Kapitel unter Abschnitt 3.2 eingegangen.

2.2 Arten von Apps

Das Ziel der vorliegenden Arbeit besteht darin, eine Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland zu implementieren. Deshalb stellt sich zu Beginn der Entwicklung einer solchen App die Frage, für welche Plattform(en) die Applikation entwickelt werden soll. Dabei lässt sich zwischen plattformspezifischen und -unabhängigen Apps unterscheiden [25, S. 59], welche im Folgenden näher betrachtet werden.

2.2.1 Native Apps

Die sog. nativen Applikationen sind plattformabhängig. Diese Apps laufen nur auf einem bestimmten Betriebssystem, wobei eine Installation auf dem Endgerät vorausgesetzt wird. Die Entwicklung solcher Apps ist dann sinnvoll, wenn diese auf die Hardware- und/oder Software-Ressourcen des Endgerätes zugreifen sollen. [25, S. 60]

2.2.2 Webbasierte Apps

Webbasierte Applikationen werden über das Internet angeboten und sind so unabhängig von plattformspezifischen Marktplätzen. Sie können von jedem mobilen Endgerät (unabhängig vom Betriebssystem) über einen Internetbrowser aufgerufen und ohne Installation verwendet werden. Für die Verwendung von webbasierten Apps wird somit nur eine aktive Internetverbindung vorausgesetzt. Jedoch sind Zugriffe auf die Hardware-Ressourcen der Endgeräte eingeschränkt. [25, S. 60]

2.2.3 Hybride Apps

Hybride Applikationen versuchen die Entwicklungsstrategien der nativen und webbasierten Apps zu verbinden. Dabei soll es möglich sein, auf Hardware-Ressourcen zuzugreifen und plattformunabhängig zu funktionieren. Diese Möglichkeit der App-Entwicklung steht jedoch aufgrund mangelnder Sicherheit in Kritik. Außerdem versuchen Betriebssystem-Hersteller hybride Technologien zu verhindern. [25, S. 60]

2.3 Android-App-Entwicklung

Im Rahmen der vorliegenden Arbeit soll eine Applikation für das Betriebssystem Android entworfen und implementiert werden. Hierbei handelt es sich um eine native App (siehe Abschnitt 2.2). In diesem Unterkapitel folgen die wichtigsten Grundlagen zum Verständnis der Entwicklung einer Android-Applikation. Im Zuge dessen wird zuerst auf die Plattform Android allgemein, die Software Android Studio sowie Android-App-Komponenten und weitere benötigte Dateien eingegangen. Zum Schluss folgt eine kurze Übersicht über die Plattform Firebase, welche verschiedene Funktionen für App-Entwickler*innen bietet.

2.3.1 Plattform Android

Das Unternehmen Android wurde 2003 von Andy Rubin gegründet und ist seit 2005 in Besitz von Google. Das Ziel von Android war es, ein offenes Betriebssystem zu schaffen, welches von jeder Person und jedem Unternehmen kostenlos verwendet und verändert werden kann. Somit hatte Android im März 2022 einen Marktanteil von 71,70 % der mobilen Betriebssysteme weltweit und kann sich aufgrund dessen seit Jahren gegenüber seinem direkten Konkurrenten iOS (mobiles Betriebssystem von Apple) mit 27,57 % durchsetzen [26]. [27]
Neben dem Betriebssystem liefert die quelloffene Plattform Android ebenso die benötigten Treiber, Bibliotheken, Frameworks und System-Applikationen [28]. Außerdem können zusätzliche Apps für das Betriebssystem entwickelt werden, welche über den zentralen Android Store *Google Play* oder andere Android-App-Stores (z. B. Samsung Galaxy Store) vertrieben, heruntergeladen und vom eigenen Virenscanner geprüft werden können [27].
Die aktuellste verfügbare Betriebssystem-Version (Stand: 13. April 2022) ist Android 12 [29]. Da mit jeder neuen Version neue Programmierschnittstellen (engl. *Application Programming Interface*, kurz API) hinzugefügt werden, gibt jede Version ein API-Level an. Beispielsweise besitzt Android 12 das API-Level 31 [30]. Damit Applikationen auf verschiedenen Betriebssystemversionen funktionieren können, kann die Mindestversion des API-Levels beim Programmieren einer Android-App angegeben werden. Neuere APIs sind dabei immer abwärtskompatibel, d. h. jede zukünftige Android-Version ist mit Apps kompatibel, die mit den APIs einer älteren Android-Version erstellt wurden. [31]

Android-Architektur

Android basiert auf Linux und ist ein quelloffener Software-Stack [28]. Der Aufbau und die Hauptkomponenten des Android-Software-Stacks werden in der nachfolgenden Abbildung 2.3 dargestellt.

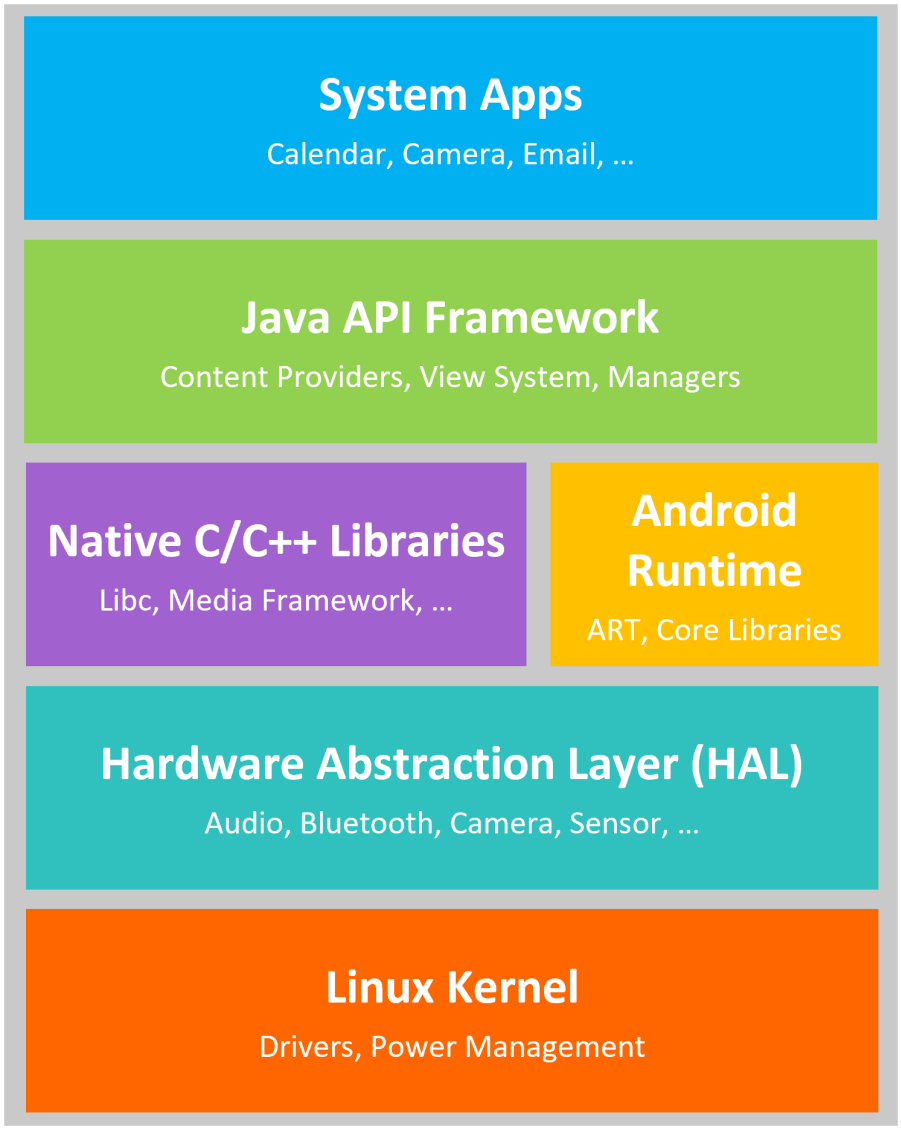


Abbildung 2.3: Aufbau des Android-Software-Stacks (adaptiert nach [28])

Der **Linux Kernel** ist das Fundament der Android-Architektur. Dadurch kann das System zum einen wichtige Sicherheitsfunktionen nutzen, um Applikationen voneinander zu isolieren. Somit werden die Apps und das System vor schädlichen Applikationen geschützt. Zum anderen können Gerätehersteller Hardwaretreiber für den bereits bekannten Kernel herstellen. [32] [28]

Der **Hardware Abstraction Layer**[10] isoliert den Kernel und die Software von der Hardware. Außerdem dient die Schicht zum Ansprechen der Hardware und besteht aus mehreren Bibliotheksmodulen. Jedes Bibliotheksmodul implementiert eine Schnittstelle für eine bestimmte Hardwarekomponente, um Hardware-Funktionen für das übergeordnete Java API Framework zugänglich zu machen. Wenn nun eine Framework-API auf die Gerätehardware zugreift, lädt Android das entsprechende Bibliotheksmodul. [28]

Die **Android Runtime**[11] dient ab Android Version 5.0 als eine eigene kleine Plattform, um entwickelte Applikationen auszuführen. Dafür kompiliert und generiert die Android Runtime zur Installationszeit die sog. DEX-Dateien[12] zu einer auf dem Zielgerät ausführbaren App. Außerdem werden eine Reihe von zentralen Laufzeitbibliotheken bereitgestellt, welche Funktionen der Programmiersprache Java enthalten und vom Java API Framework verwendet werden. [28]

Die **nativen C/C++ Libraries**[13] sind für viele Kernkomponenten und -dienste des Android-Systems (z. B. Hardware Abstraction Layer und Android Runtime) erforderlich. Über Java-Framework-APIs können einige dieser nativen Bibliotheken ebenso für selbst entwickelte Apps zugänglich gemacht werden. [28]

Das **Java-API-Framework** besteht aus in Java geschrieben APIs, welche als Bausteine für die Erstellung von Apps benötigt werden. Mithilfe dieser APIs ist der gesamte Funktionsumfang des Android-Betriebssystems verfügbar und Entwickler*innen können auf die gleichen APIs zugreifen, welche ebenso von Android-System-Apps verwendet werden. Das API-Framework umfasst beispielsweise ein *View System*, mit welchem die Benutzeroberfläche der App aufgebaut werden kann und *Content Providers*, mit denen Apps auf Daten anderer Apps zugreifen können. [28]

Die **System-Apps** sind eine Reihe von Kernanwendungen, welche von Android mitgeliefert werden, z. B. für SMS, Kalender und Internet-Browsing. Die entsprechenden Apps können, müssen aber nicht verwendet werden. Jedoch gibt es auch Ausnahmen, beispielsweise die „Einstellungen“-App des Systems. Diese System-Apps dienen zum einen als Standardapps für die Benutzer*innen und zum anderen stellen sie Kernfunktionen bereit, auf welche Entwickler*innen über selbst entwickelte Apps zugreifen können. [28]

[10] dt. Hardware-Abstraktionsschicht
[11] dt. Laufzeitumgebung
[12] Dalvik Executable Format: ein speziell für Android entwickeltes Bytecode-Format
[13] dt. Bibliotheken

2.3.2 Android Studio

Android Studio[14] ist die offizielle integrierte Entwicklungsumgebung (IDE) zum Entwickeln von Android-Apps. Da Android Studio auf der IDE IntelliJ IDEA[15] basiert, enthält es dessen Code-Editor und Entwicklertools. Dabei werden die Programmiersprachen Java, Kotlin und C++ unterstützt, welche zum Entwickeln von Android-Apps verwendet werden können [33]. Zusätzlich bietet Android Studio z. B. eine einheitliche Umgebung, in der für alle Android-Versionen und -Geräte entwickelt werden kann sowie Codevorlagen für allgemeine App-Funktionen, einen Emulator zum Testen der App und weitere Funktionen. Die IDE organisiert den Code einer App in einem Projekt, welches alle Dateien zum Erstellen einer App enthält, z. B. Quellcode und Build-Konfigurationen [34]. Das in Android Studio integrierte Gradle-basierte Build-System kann daraus automatisch ein Anwendungsprogramm erzeugen. [35]

2.3.3 Android-App-Komponenten

App-Komponenten sind Einstiegspunkte, über welche das System oder ein*e Benutzer*in in die App gelangen kann. Insgesamt gibt es vier Arten von Android-App-Komponenten, wovon jede einen bestimmten Zweck erfüllt: *Activities*, *Services*, *Broadcast Receivers* und *Content Providers*. [33]

Activities

Eine *Activity*[16] stellt einen einzelnen Bildschirm mit einer Benutzeroberfläche dar und dient somit als Einstiegspunkt, über welchen die Benutzer*innen mit der App interagieren können. Normalerweise besteht eine App aus mehreren *Activities*, wovon eine als *Main Activity*[17] angegeben wird. Die *Main Activity* ist der erste Bildschirm, der angezeigt wird, wenn die Nutzer*innen die App öffnen. Obwohl die verschiedenen *Activities* einer App zusammenarbeiten, sind sie trotzdem unabhängig voneinander. So kann jede *Activity* einzeln angesteuert (z. B. von anderen Apps) oder pausiert werden, sobald eine neue *Activity* aufgerufen wird. Das Aussehen der Benutzeroberfläche einer *Activity* wird in der sog. *layout-Ressource* (siehe Abschnitt 2.3.5) festgelegt. [33] [36]
Activities können zudem sog. Fragmente hosten. Das bedeutet, eine *Activity* kann ein oder mehrere Fragment(e) beinhalten und an entsprechender Stelle anbieten bzw. anzeigen. Fragmente besitzen ihr eigenes Layout und sind Teil einer *Activity*. So können *Activities* als globales Element der Benutzeroberfläche dienen und Fragmente als einzelne Bildschirme innerhalb dieser *Activity*. [37]

[14] `https://developer.android.com/studio`
[15] `https://www.jetbrains.com/idea/`
[16] dt. Aktivität
[17] dt. Haupt-Aktivität

In Abbildung 2.4 ist ein Beispiel einer *Activity*-Fragmente-Beziehung dargestellt. Die *Activity* (blau) besteht dabei aus einer Navigationsleiste und einem Navigationsmenü, welches über Berührung der drei Striche in der Navigationsleiste aufgerufen werden kann. Über das Menü kann auf die Fragmentbildschirme (grün) zugegriffen werden, wobei beide Fragmente Teil derselben *Activity* sind.

Abbildung 2.4: Darstellung einer *Activity*-Fragmente-Beziehung. Die *Activity* (blau) besteht aus einer Navigationsleiste und einem Navigationsmenü, über welches auf beide Fragmente (grün) zugegriffen werden kann. (eigene Darstellung)

Wenn Nutzer*innen durch die App navigieren, werden verschiedene *Activity*-Zustände durchlaufen, z. B. dass eine *Activity* erstellt oder beendet wurde. Insgesamt gibt es sechs sog. *Callback*[18]-Methoden, um diese Aktivitäts-Zustände zu steuern. Mithilfe dieser Methoden können App-Funktionen implementiert werden, welche vom Zustand der *Activity* abhängig sind. [38]
In der nachfolgenden Tabelle 2.2 werden diese *Callback*-Methoden genannt und es wird beschrieben, wann sie aufgerufen werden.

Tabelle 2.2: Übersicht über die *Callback*-Methoden, welche die *Activity*-Zustände steuern (angelehnt an [38])

***Callback*-Methode**	**Wird aufgerufen, wenn...**
`onCreate()`	... die *Activity* erstellt wird.
`onStart()`	... die *Activity* sichtbar wird.
`onResume()`	... die Nutzer*innen mit der *Activity* interagieren.
`onPause()`	... die *Activity* nicht mehr im Vordergrund ist.
`onStop()`	... die *Activity* nicht mehr sichtbar ist.
`onDestroy()`	... die *Activity* beendet wird.

[18] dt. Rückruf

Obwohl ein Fragment an eine *Activity* gebunden ist, durchläuft dieses ebenso verschiedene Zustände. Deshalb gibt es neben den *Callback*-Methoden der *Activities* ebenso welche für Fragmente. So wird beispielsweise die Methode `onCreateView()` aufgerufen, wenn das Fragment seine Benutzeroberfläche generiert. [39]

Services

Ein *Service*[19] dient dazu, eine App im Hintergrund ausführen zu können. Dementsprechend besitzt ein Service keine Benutzeroberfläche, mit welcher interagiert werden kann. Es wird zwischen zwei Arten von Services unterschieden: *Started Services* und *Bound Services*. [33]
Started Services laufen so lange, bis ihre Arbeit abgeschlossen ist und werden in zwei verschiedene Arten unterteilt, je nachdem, wie das System mit ihnen umgeht. Bei der ersten Art teilt der*die Benutzer*in der App bzw. dem System direkt etwas mit, beispielsweise bei der Wiedergabe von Musik. Auch beim Verlassen der App erwartet der*die Benutzer*in vom System, dass dieses die Musik weiterspielt. Die zweite Art der *Started Services* sind die Hintergrunddienste, von dem die Benutzer*innen nichts direkt mitbekommen, z. B. die Synchronisierung von Daten. Dadurch, dass der*die Benutzer*in nicht direkt weiß, dass der Service im Hintergrund läuft, hat das System mehr Freiheit bei der Verwaltung des Prozesses, um diesen zu starten und zu beenden, wenn es nötig ist. [33]
Bound Services sind gebundene Dienste und laufen, weil andere Anwendungen oder das System den Service nutzen möchten. So stellt der jeweilige Service dem anderen Prozess eine API zur Verfügung und das System weiß automatisch, dass zwischen diesen Prozessen eine Verbindung besteht. [33]
Des Weiteren werden Services für z. B. Live-Hintergrundbilder, Bildschirmschoner und andere zentrale Systemfunktionen verwendet, welche als Applikationen implementiert sind und sich an das System binden, wenn sie ausgeführt werden. [33]

Broadcast Receivers

Broadcast Receivers[20] dienen innerhalb einer Art Benachrichtigungssystem dazu, dass Apps außerhalb der normalen Interaktionen von Benutzer*innen auf systemweite Ankündigungen (sog. Broadcasts) reagieren können. Diese Ankündigungen werden ausgelöst, wenn vorher festgelegte Ereignisse eintreten. Wird beispielsweise der Flugmodus ein- oder ausgeschaltet, wird diese Information an alle Apps gesendet, die den Empfang dieses Ereignisses abonniert haben und diese Apps können wiederum mit entsprechenden Ereignissen reagieren. [33] [40]

[19] dt. Dienst
[20] dt. Übertragungsempfänger

Die meisten Broadcasts stammen vom System (z. B. Flugmodus aktiviert, Akku leer, Bildschirm ausgeschaltet), jedoch können auch Apps solche Ankündigungen initiieren, z. B. dass heruntergeladene Daten auf dem Gerät zur Verfügung stehen. Außerdem besitzen *Broadcast Receivers* keine Benutzeroberfläche, können jedoch eine Benachrichtigung in der Statusleiste erstellen, um den*die Benutzer*in über das Ereignis zu informieren. [33]

Content Providers

Content Providers[21] verwalten einen Satz von App-Daten, welche an den folgenden Orten gespeichert sein können: im Dateisystem, einer SQLite-Datenbank, im Internet oder einem anderen permanenten Speicherort, auf den eine App zugreifen kann. Über *Content Provider* können Apps diese Daten abfragen oder auch ändern, wenn die jeweiligen Berechtigungen vorliegen. [33]

2.3.4 Manifestdatei

Ein weiterer wichtiger Bestandteil einer Android-Applikation ist die Manifestdatei. Diese Datei befindet sich im Root-Verzeichnis des App-Projektes und heißt *AndroidManifest.xml*. Bei Erstellung eines Projekts in Android Studio wird die Datei automatisch hinzugefügt. [33]
In dieser Datei müssen zum einen alle implementierten Komponenten der App (siehe Abschnitt 2.3.3) deklariert werden. Zum anderen werden dort alle Berechtigungen der Benutzer*innen, welche die App benötigt (z. B. Internetzugang) sowie das minimal benötigte API-Level definiert. Ebenso werden die von der App benötigten Hardware- und Softwarefunktionen (z. B. Bluetooth) und API-Bibliotheken deklariert. [33]

2.3.5 App-Ressourcen

Die sog. App-Ressourcen sind zusätzliche Dateien neben dem eigentlichen Quellcode. Sie sind in verschiedene Typen unterteilt und enthalten z. B. Bilder und Audiodateien sowie alles weitere, was mit der visuellen Präsentation einer App in Verbindung steht. Bei den meisten dieser Dateien handelt es sich um sog. XML-Dateien (siehe Abschnitt 2.4.2). Der Quellcode kann dabei auf die App-Ressourcen zugreifen, ohne selbst verändert zu werden. [33] [41]
In der folgenden Tabelle 2.3 werden die im Rahmen der vorliegenden Arbeit verwendeten App-Ressourcen kurz erläutert.

[21] dt. Inhaltsanbieter

Tabelle 2.3: Beschreibung der verwendeten App-Ressourcen

App-Ressource	Beschreibung
drawable	Enthält Bitmap-Dateien (z. B. .png und .jpg) oder XML-Dateien, welche zu Grafik-Dateien (z. B. Bitmap-Dateien) kompiliert werden können. [41]
layout	Enthält XML-Dateien, welche das Aussehen der Benutzeroberfläche (z. B. *Activities* und Fragmente) darstellen. [41]
menu	Enthält XML-Dateien, welche App-Menüs definieren. [41]
mipmap	Enthält die gleichen Grafik-Dateitypen wie die *drawable-Ressource* für verschiedene Versionen des App-Icons. Die Versionen unterscheiden sich in der Pixeldichte sowie der Abrundung der Ecken. [41]
navigation	Enthält einen Navigationsgraphen in Form einer XML-Datei. Teil des Navigationsgraphen sind alle Bereiche, zu denen die Nutzer*innen innerhalb der App navigieren können (sog. *Destinations*[22]) sowie die möglichen Navigationspfade (sog. *Actions*[23]) von einer *Destination* zu einer anderen. [42]
values	Enthält XML-Dateien, welche Daten mit unterschiedlichen Eigenschaften definieren. Dabei können in einer Datei Daten mit verschiedenen Eigenschaften definiert werden. Aufgrund einer besseren Übersicht sollten Daten mit verschiedenen Eigenschaften jedoch in unterschiedliche Dateien aufgeteilt werden. So können in der Datei `strings.xml` alle im Rahmen der App verwendeten Texte definiert werden, in der Datei `colors.xml` alle benötigten Farbwerte und in `themes.xml` die zum Beschreiben des allgemeinen Designs verwendeten Werte. [41]

2.3.6 Layoutgestaltung

Wie bereits in Abschnitt 2.3.5 erwähnt, befinden sich in der *layout-Ressource* Dateien zum Definieren der Benutzeroberfläche. Innerhalb dieser Dateien werden die Elemente des Layouts mithilfe von sog. *View-* und *ViewGroup*-Objekten dargestellt. [43]
Ein ***View***-Objekt ist ein Element, welches Benutzer*innen in einer App sehen und mit welchem sie interagieren können (z. B. ein Button). [43]
In der folgenden Tabelle 2.4 werden alle im Rahmen der vorliegenden Arbeit verwendeten *View*-Objekte genannt und beschrieben.

[22] dt. Ziele
[23] dt. Aktionen

Tabelle 2.4: Nennung und Beschreibung der verwendeten *View*-Objekte

***View*-Objekte**	**Beschreibung**
ImageView	Zeigt Bildressourcen an. [44]
ListView	Eine vertikal scollbare Liste von Einträgen (sog. *Views*), wobei jede *View* gleich aufgebaut ist und direkt unter der vorherigen Ansicht in der Liste positioniert wird. Mithilfe einer eigenen *layout*-Ressource kann das Aussehen der *Views* festgelegt und mithilfe eines Adapters gefüllt werden. [45]
NavigationView	Stellt ein Navigationsmenü dar, welches in einer *menu*-Ressource definiert ist. [46]
TextView	Zeigt Texte an. [47]

Die sog. ***ViewGroup***-Objekte definieren die Struktur, wie *View*- und andere *ViewGroup*-Objekte angeordnet sind. [43]
In Tabelle 2.5 werden die im Rahmen der vorliegenden Arbeit verwendeten *ViewGroup*-Objekte genannt und beschrieben.

Tabelle 2.5: Nennung und Beschreibung der verwendeten *ViewGroup*-Objekte

***ViewGroup*-Objekte**	**Beschreibung**
AppBarLayout	Ein Symbolleisten-Layout, welches *Views* vertikal anordnet und Funktionen für die Symbolleiste bereitstellt. [48]
ConstraintLayout	In diesem Layout können Views flexibel angeordnet und in ihrer Größe angepasst werden. [49]
CoordinatorLayout	Ein leistungsstarkes FrameLayout, welches als Container für eine bestimmte Interaktion mit untergeordneten *Views* dienen kann. [50]
DrawerLayout	Ermöglicht sog. „*Drawer-Views*“ wie Schubladen aus dem linken oder rechten Rand des Bildschirms herauszuziehen. [51]
FrameLayout	Ein Layout, um ein einzelnes Element (*View*) auf einem Bildschirm anzuzeigen. [52]
LinearLayout	Ordnet alle verwendeten *Views* entweder horizontal in einer Zeile oder vertikal in einer Spalte an. [53]
RelativeLayout	In diesem Layout können die Views im Verhältnis zu anderen Views angeordnet werden. [54]
ScrollView	Eine *ViewGroup*, mit welcher durch die darin platzierten *Views* gescrollt werden kann. [55]
Toolbar	Eine Symbolleiste, welche z. B. einen Navigationsbutton zum Wechseln durch die *Destinations* enthalten kann. [56]

Auf die *View*-Objekte können sog. *Event-Listener* angewendet werden. Mithilfe dieser ist es möglich, die Interaktionen der Nutzer*innen mit der Benutzeroberfläche der App abzufangen. Dies geschieht mit *Callback*-Methoden, welche aufgerufen werden, wenn die entsprechende Interaktion mit einem *View*-Objekt stattfindet. So können App-Funktionen implementiert werden, welche von den Interaktionen der Benutzer*innen innerhalb der App abhängen. Beispielsweise fängt der `OnClickListener()` das Event ab, wenn Nutzer*innen ein *View*-Objekt berühren und der `OnItemClickListener()`, wenn ein Listeneintrag innerhalb eines *ListViews* berührt wird [57]. [58]

2.3.7 Firebase

Firebase ist eine App-Entwicklungsplattform von Google, welche verschiedene Funktionen für App-Entwickler*innen bietet [59].
Zum einen kann Firebase als Backend-Infrastruktur genutzt werden. Es besteht z. B. die Möglichkeit, App-Daten in der Cloud zu speichern, sie von dort abzurufen und sie zu synchronisieren (*Cloud Firestore*). Mithilfe der *Realtime Database* können JSON-Daten (siehe Abschnitt 2.4.3) in Echtzeit gespeichert und synchronisiert werden. Durch die *Authentication*-Option kann der App ein Ende-zu-Ende-Identitätsnachweis zur Authentifizierung von Benutzer*innen geboten werden und der *Cloud Storage* speichert und stellt von Benutzer*innen generierte Daten bereit. Neben den vier genannten Optionen bietet Firebase auch noch weitere sog. *Build*-Produkte an. [60]
Zum anderen kann Firebase mittels der sog. *Release & Monitor* Produkte zur Veröffentlichung und Überwachung der App verwendet werden. Beispielsweise kann mit *Google Analytics* die Gewinnung von Nutzer*innen und mithilfe von *Performance Monitoring* die Leistung der App überwacht werden. [61]
Zuletzt gibt es noch die sog. *Engage*-Produkte, welche zur Steigerung der Usability dienen können. Auch hier überwacht *Google Analytics* die Aktivitäten der Benutzer*innen und es können z. B. Versuche zum Testen verschiedener App-Ideen unter Einbezug der Nutzer*innen durchgeführt werden (*A/B Testing*). Außerdem ist es möglich, mithilfe von *Cloud Messaging* Push-Benachrichtigungen (sog. *Notifications*) an die Benutzer*innen der App zu senden. [62]

2.4 Sprachen und Austauschformate

In diesem Abschnitt folgt ein kurzer Exkurs zu der Programmiersprache Java, der Auszeichnungssprache XML und dem Datenaustauschformat JSON. Sowohl die zwei Sprachen als auch das Datenaustauschformat können zum Entwickeln einer Android-App verwendet werden und wurden im Rahmen der vorliegenden Arbeit eingesetzt.

2.4.1 Java

Java ist sowohl eine Programmiersprache als auch eine Entwicklungsplattform und gehört zu Oracle[24] [63, S. 2]. Zum Programmieren mit Java wird das *Software Development Kit*[25] (SDK) von Java benötigt, das sog. *Java Development Kit* (JDK). Dieses stellt z. B. eine Laufzeitumgebung und Tools für die Entwicklung bereit [64].
Bis 2019 war Java die von Google bevorzugte Programmiersprache zum Entwickeln einer Android-Applikation, welche jedoch von Kotlin abgelöst wurde. Das bedeutet aber nicht, dass Java nicht mehr verwendet wird, sondern nur, dass Google selbst seine eigenen Apps mittlerweile in Kotlin programmiert. Java kann weiterhin zum Entwickeln von Apps und auch zusammen mit Kotlin verwendet werden. [65] [66]
Außerdem ist Java noch immer auf Platz zwei der beliebtesten Programmiersprachen weltweit (Stand: 4. Mai 2022) [67], weshalb im Rahmen dieser Arbeit mit der Sprache Java programmiert wurde.

2.4.2 XML

Die *Extensible Markup Language*[26] (XML) ist eine sog. Metasprache, eine Sprache zum Beschreiben von anderen Sprachen. Das bedeutet, XML selbst stellt Daten nicht dar, sondern beschreibt nur die Struktur, wie die Daten aussehen sollen. [63, S. 641]
Ein XML-Dokument besteht aus sog. *Markups* und Inhalt. Markups sind die codierten Bezeichnungen der logischen Struktur und der Inhalt ist der Dokumententext, welcher nicht als Markup interpretiert wird. Die Markups werden durch sog. *Tags* („<>“) dargestellt, welche jeweils einen Namen besitzen. Der Unterschied zu anderen Auszeichnungssprachen wie HTML ist, dass ein eigenes Vokabular mit benutzerdefinierten Tags deklariert werden kann. [63, S. 641 f.]
Android Studio bietet ein eigenes XML-Vokabular zum Erstellen der Benutzeroberfläche einer App. Somit kann das in XML definierte Layout vom eigentlichen Code getrennt werden. Jedoch werden innerhalb des Programmcodes die jeweiligen Funktionen der Benutzeroberfläche programmiert. [43]

[24] `https://www.oracle.com/index.html`
[25] dt. Software-Entwicklungspaket
[26] dt. erweiterbare Auszeichnungssprache

2.4.3 JSON

Die *JavaScript Object Notation* (JSON) ist ein Datenaustauschformat in Textform, welches für den Menschen einfach zu lesen und schreiben ist. Obwohl es auf der Programmiersprache JavaScript basiert, ist JSON programmiersprachenunabhängig und wird durch eine Liste von Name/Wert-Paaren dargestellt. [68, S. 187]
JSON wird von verschiedenen Anwendungen genutzt, z. B. Twitter und Facebook, der Google Maps API oder auch NoSQL-Datenbankmanagementsystemen [68, S. 187]. So verwendet auch die *Realtime Database* von Firebase JSON, um Daten zu speichern und zu synchronisieren. [60]

3 Konzeptioneller Entwurf

In diesem Kapitel wird ein mögliches Konzept einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland entworfen. Dabei folgt zu Beginn eine Übersicht darüber, warum hierfür eine Android-Applikation sinnvoll wäre. Anschließend wird der IST-Zustand einer solchen App in Deutschland analysiert und abschließend der gewünschte SOLL-Zustand definiert. Bei Letzterem wird sowohl auf die funktionalen sowie nicht-funktionalen Anforderungen eingegangen und anschließend eine daraus folgende erste App-Struktur dargestellt.

3.1 Warum eine Android-Applikation?

Im Rahmen der vorliegenden Arbeit stehen zu Beginn die folgenden zwei Fragen besonders im Fokus:

1. Warum wird hierfür eine App benötigt bzw. warum ist eine App hierfür sinnvoll?
2. Warum eine Applikation für Android-Geräte?

Wie bereits in Kapitel 2.1 beschrieben, dient ein AMBER-Alert zur schnellen Suche vermisster Kinder, da die Zeit zum Auffinden eines Kindes eine wesentliche Rolle spielt. Deshalb ist eine umfassende Alarmierung der Bevölkerung zum Ausschauhalten nach diesen Kindern besonders wichtig.
In einer Studie vom Statistischen Bundesamt wird deutlich, dass 97,6 % der Haushalte in Deutschland (Stand: 1. Januar 2021) ein Mobiltelefon besitzen, wovon 85,6 % einem Smartphone entsprechen [5].
Eine Studie der Murmuras GmbH zeigt zusätzlich, dass die durchschnittliche tägliche Smartphone-Nutzung in Deutschland (Stand: 2020 vor und nach dem 1. coronabedingten Lockdown) zwischen 03:30 und 04:00 Stunden lag. Während des 1. Corona-Lockdowns betrug sie sogar über 04:00 Stunden täglich. [69]
Diese Zahlen zeigen deutlich, dass mithilfe einer Smartphone-App der Großteil der Bevölkerung in Deutschland schnellstmöglich erreicht werden könnte, um auf vermisste Kinder und Jugendliche aufmerksam zu machen.

Zur Beantwortung der zweiten Frage wird der Marktanteil der mobilen Betriebssysteme in Deutschland betrachtet. Dieser liegt für Android bei 70,03 % (Stand: April 2022) und befindet sich somit mit über 40 % vor seinem direkten Konkurrenten iOS von Apple [70], was deutlich zeigt, dass mithilfe einer Android-Applikation die meisten Menschen in Deutschland erreicht werden können.

3.2 IST-Zustand

Vor der Analyse der benötigten Anforderungen an eine Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland wird der aktuelle Stand einer solchen App genauer betrachtet. Hierfür wurden die vier Apps „Deutschland Findet Euch“, „Vermisst in Thüringen“, „hessenWARN“ und „ChildRescue“ analysiert sowie im Folgenden beschrieben. Bei diesen Apps handelt es sich um bereits existierende Apps zur Suche nach vermissten Personen in Deutschland.

Deutschland Findet Euch

Bereits 2010 wurde eine App zur Suche vermisster Kinder in Deutschland entwickelt. Die Initiative Vermisste Kinder hat mit der Aktion „Deutschland Findet Euch“ eine Möglichkeit geschaffen, um Personen aktiv an der Suche nach vermissten Kindern beteiligen zu können. Die Initiative nutzte hierfür verschiedene Plattformen und Tools, um auf vermissten Fälle aufmerksam zu machen, z. B. Facebook. Ebenso wurde eine iOS-App entwickelt, um die Nutzer*innen über vermisste Kinder zu informieren. [71]
Während die Facebook-Seite „Deutschland findet euch“[27] noch immer aktiv betrieben wird (Stand: August 2022), konnte sich die App jedoch nicht durchsetzen. Ein möglicher Grund hierfür könnte sein, dass im Jahr 2011 weniger als ein Viertel der deutschen Haushalte ein Smartphone besaß [72]. Außerdem wurde iOS von Android als das Betriebssystem mit den meisten Marktanteilen in Deutschland abgelöst (siehe Abschnitt 3.1).

Vermisst in Thüringen

Eine noch aktuell betriebene App zur Suche von vermissten Personen in Deutschland ist die App „Vermisst in Thüringen“[28]. Jedoch, wie aus dem Namen schon zu schließen ist, handelt es sich dabei nur um Vermisstenfälle aus Thüringen. Über die App sowie die dazugehörige gleichnamige Facebook-Seite[29] werden Öffentlichkeitsfahndungen zu vermissten Personen der Thüringer Polizei bereitgestellt, um die Personen somit (schnell) zu finden. Allerdings handelt es sich dabei nicht nur um Fälle von vermissten Minderjährigen, sondern um alle im Freistaat vermissten Personen. [73]

[27] https://www.facebook.com/deutschlandfindeteuch/
[28] https://play.google.com/store/apps/details?id=com.Tobit.android.Slitte7180706553&hl=gsw&gl=US
[29] https://de-de.facebook.com/VITHUERINGEN/

hessenWARN

Eine weitere noch aktuell betriebene App zur Suche nach vermissten Personen innerhalb Deutschlands ist die App „hessenWARN“. Diese Applikation wurde aus der Warn-App „KATWARN“ weiterentwickelt und ist für Android[30]- und iOS[31]-Geräte verfügbar. Neben den für „KATWARN“ typischen Gefahrenwarnungen enthält die App „hessenWARN“ zusätzlich Vermisstenmeldungen. Jedoch ist diese App ebenso nur auf ein Bundesland beschränkt und enthält Vermisstenfälle von Personen aller Altersgruppen aus Hessen. [74]

ChildRescue

Neben den speziell auf Deutschland zugeschnittenen Apps gibt es auch eine EU-weite App zur Suche vermisster Kinder. Die App wurde im Rahmen des Projektes „ChildRescue“ entwickelt. Das Projekt wurde vom Forschungs- und Innovationsprogramm *Horizon 2020* der Europäischen Union verwaltet und lief vom 1. Januar 2018 bis zum 31. Dezember 2020. [75]
Die dazugehörige App ist unter gleichem Namen im Google Play Store[32] und App Store[33] erhältlich. Dabei fällt jedoch auf, dass die Android-App zuletzt im Oktober 2020 (Stand: August 2022) aktualisiert wurde. Dies lässt darauf schließen, dass seit Projektende im Jahr 2020 nicht mehr an der App gearbeitet wird.

Zusammenfassung

Aus dem Vorangegangenen wird deutlich, dass es in Deutschland nur zwei tatsächlich aktiv betriebene Apps zur Suche von vermissten Personen gibt. Jedoch sind diese Apps auf jeweils ein Bundesland beschränkt und enthalten Vermisstenfälle zu Personen jeden Alters.
Diese IST-Analyse zeigt, dass es zum aktuellen Zeitpunkt (Stand: August 2022) keine deutschlandweite App zur Alarmierung der Bevölkerung bei vermissten Kindern gibt. Deshalb werden im folgenden Abschnitt die hierfür benötigten Anforderungen analysiert und im anschließenden Kapitel 4 die Umsetzung einer solchen App genauer betrachtet.

[30] `https://play.google.com/store/apps/details?id=de.hessen.hmdis.hessenwarn&hl=de&gl=US`
[31] `https://apps.apple.com/de/app/hessenwarn/id1482894790`
[32] `https://play.google.com/store/apps/details?id=eu.ubitech.childrescue`
[33] `https://apps.apple.com/us/app/id1524504801`

3.3 SOLL-Zustand

Da es keine aktiv betriebene deutschlandweite App zur Alarmierung der Bevölkerung bei vermissten Kindern gibt, soll in diesem Abschnitt der gewünschte SOLL-Zustand einer solchen App definiert werden. Im Folgenden wird dabei zwischen den funktionalen und nicht-funktionalen Anforderungen an die App unterschieden. Außerdem wird ein erstes mögliches Modell der App-Struktur mit den benötigten *Activities* und Fragmenten aufgezeigt.

Funktionale Anforderungen

Es ist vorgesehen, dass die App aus einem Hauptmenü bestehen soll, über welches die Nutzer*innen zwischen verschiedenen Bereichen der App navigieren können.
Einer dieser Bereiche soll eine Übersicht über die aktuellen Vermisstenfälle bieten, was zugleich auch der erste Bildschirm sein soll, den die Nutzer*innen beim Öffnen der App sehen. Des Weiteren sollen die Nutzer*innen die Möglichkeit haben, durch Auswahl einer der angezeigten Fälle detaillierte Informationen sowie ein Foto zum jeweiligen vermissten Kind zu erhalten.
Neben der Fallübersicht sollte die App zudem Informationen zu Handlungsmaßnahmen, wenn das eigene Kind vermisst wird, allgemeine Informationen über AMBER sowie Informationen über die App bereithalten. Ebenso soll es eine Möglichkeit geben, auf eventuell vorhandene Social-Media-Kanäle zugreifen zu können.
Weiterhin muss sichergestellt werden, dass die App ohne eine nötige Registrierung verwendet werden kann, um zu verhindern, dass Personen die App aufgrund dessen nicht nutzen. Außerdem sollten die angezeigten Fälle nur von bestimmten Personen oder einer zentralen Stelle über ein entsprechendes Backend angelegt und verwaltet werden können. Wurde ein neuer Fall angelegt, soll zusätzlich eine Benachrichtigung an die Nutzer*innen der App gesendet werden.

Nicht-funktionale Anforderungen

Neben den funktionalen Anforderungen an das System werden ebenso nicht-funktionale Anforderungen gestellt. Hierzu zählt zum einen, dass die App auf möglichst vielen Geräten sowie im *Dark Mode* unterstützt werden sollte. Weiterhin darf die Usability nicht außer Acht gelassen werden. Die App sollte einfach und verständlich aufgebaut sowie selbsterklärend sein, sodass den User*innen die Nutzung nicht unnötig erschwert wird. Die Nutzer*innen sollten innerhalb der App mit Höflichkeitsformen angesprochen werden sowie Feedback unter bestimmten Bedingungen erhalten. Außerdem sollte beim Entwickeln der App ressourcensparend gearbeitet werden, z. B. durch eine speicherplatzoptimierte Entwicklung.

Erste App-Struktur

In Abbildung 3.1 ist eine erste Struktur der App bzw. der benötigten *Activities* und Fragmente sowie deren Zusammenhänge dargestellt. Dabei wird deutlich, dass über eine *Main Activity* auf die in den funktionalen Anforderungen definierten Bereiche (Fragmente) zugegriffen werden soll. Zusätzlich zu diesen muss nach Art. 13 DSGVO zu einer Datenschutzerklärung sowie nach § 5 Abs. 1 TMG bzw. § 18 Abs. 1 MStV zu einem Impressum navigiert werden können. Durch Auswahl einer der angezeigten Fälle im Fragment, welches die aktuellen Fälle enthält, soll eine *Activity* mit detaillierten Fallinformationen zum ausgewählten Fall angezeigt werden. Diese *Activity* soll ebenfalls ein Bild des vermissten Kindes enthalten, welches durch Berührung im Vollbild angezeigt wird. Dieses Vollbild soll ebenso als selbstständige *Activity* implementiert werden.

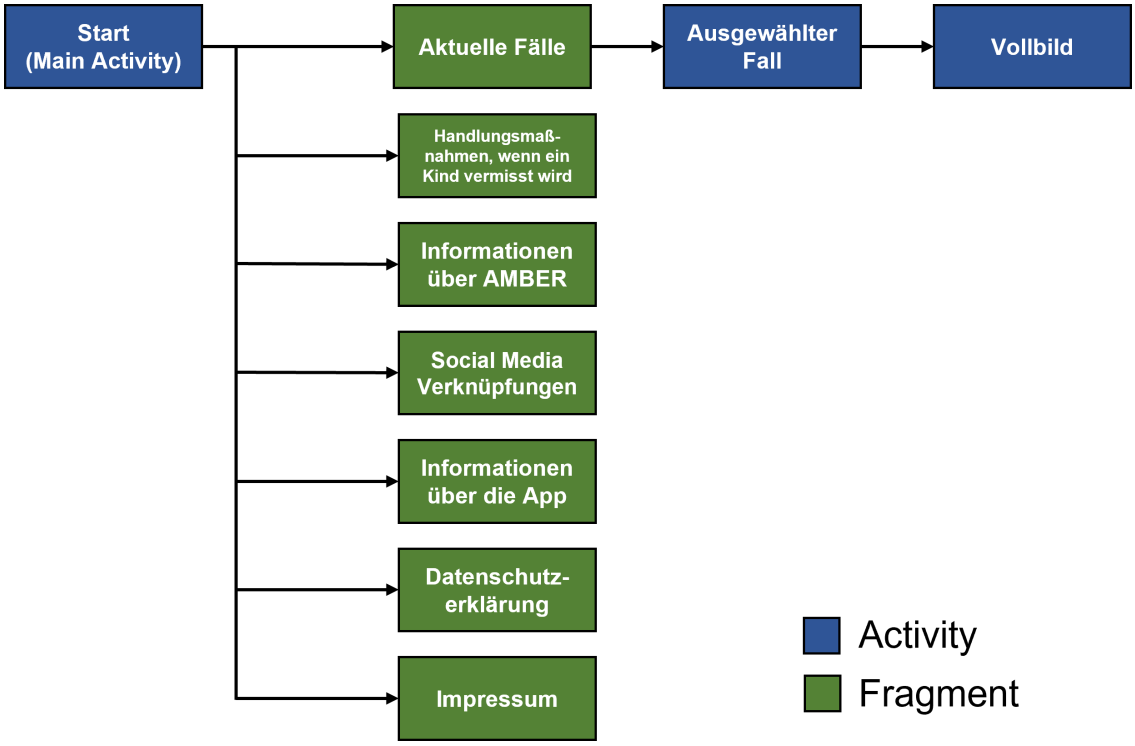


Abbildung 3.1: Struktur der benötigten *Activities* (blau) und Fragmente (grün) sowie ihre Zusammenhänge (eigene Abbildung)

4 Implementierung

In diesem Kapitel wird auf die Implementierung der Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern eingegangen. Dabei wird sich an dem in Abschnitt 3.3 definierten SOLL-Zustand orientiert. Zur Umsetzung dessen wurde zwischen zwei wesentlichen Bereichen unterschieden: den App-Ressourcen, in welchem das für die Nutzer*innen sichtbare Layout definiert ist, und den Funktionalitäten des Layouts, welche die Interaktivitäten der Benutzer*innen verarbeiten (Java-Klassen). Für die Umsetzung des Layouts und der Funktionalitäten wurde mit Android Studio gearbeitet. Zusätzlich wurde die Entwicklungsplattform Firebase als Backend-Infrastruktur zum Verwalten der Fälle herangezogen. Deshalb wird zu Beginn auf das Setup von Android Studio und Firebase eingegangen und anschließend auf die App-Ressourcen sowie die Java-Klassen. Abschließend folgt ein Einblick in die Testphase, da ein fehlerfreier Betrieb der App nur mit Tests gewährleistet werden kann.

4.1 Setup Android Studio

Im Rahmen der vorliegenden Arbeit wurde mit Android Studio und der Programmiersprache Java gearbeitet, weshalb zu Beginn das JDK[34] und anschließend Android Studio[35] installiert werden musste. Das JDK wurde in der Version 17.0.2 und Android Studio in der Version „Android Studio Chipmunk | 2021.2.1 Patch 2“ verwendet. Bei der Installation von Android Studio wurde zusätzlich *Android Virtual Device* ausgewählt und mit installiert. *Android Virtual Device* wird zum Testen der App auf einem virtuellen Gerät (in einem Emulator) genutzt. Zu einer schnelleren Nutzung des Emulators auf Computern mit Intel-Prozessoren kann zusätzlich der *Intel Hardware Accelerated Execution Manager*[36] installiert werden.
Nach der Installation von Android Studio musste dieses noch eingerichtet werden. Hierfür war es nötig, einige Android-SDK-Komponenten nachzuinstallieren. Anschließend wurde beim *Install Type* die Standard-Variante ausgewählt.
Nach Abschluss der Einrichtung konnte ein neues Projekt erstellt werden. Hierfür wurde eine *Empty Activity* ausgewählt. Eine *Empty Activity* ist, wie es der Name schon sagt, leer und besitzt dementsprechend keine von Android vorgenommenen Voreinstellungen. Anschließend wurden die Projekt-Einstellungen wie in Abbildung 4.1 festgelegt. Als minimale Android-Version, auf welcher die App ausführbar sein soll, wurde die Version 5.0 gewählt, da diese laut Android selbst auf 98,6 % aller Android-Geräte läuft (siehe Abbildung 4.1, Stand: 4. August 2022).

[34] `https://www.oracle.com/java/technologies/downloads/`
[35] `https://developer.android.com/studio#downloads`
[36] `https://github.com/intel/haxm`

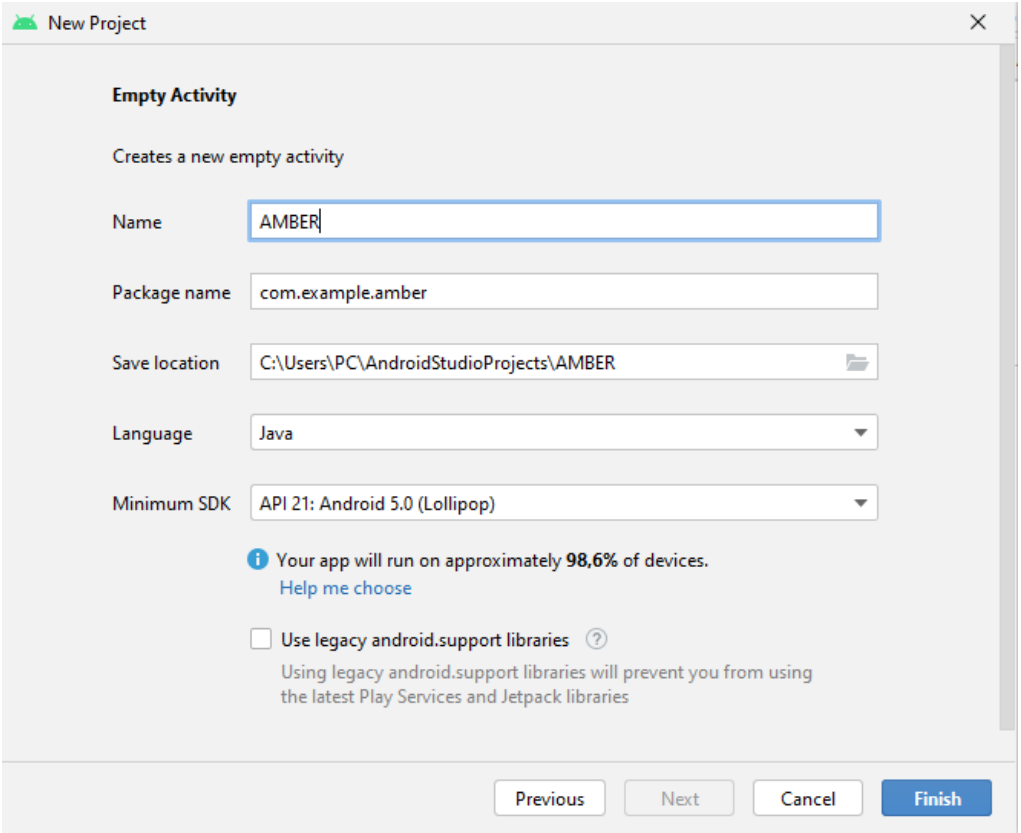


Abbildung 4.1: Screenshot der festgelegten Projekt-Einstellungen (eigener Screenshot)

Anschließend wurde durch Betätigung des *Finish*-Buttons das Projekt und somit alle benötigten Projektdateien erstellt. In Abbildung 4.2 ist die Android-Ansicht der erstellten Projektverzeichnisse dargestellt. Dabei handelt es sich jedoch nicht um die tatsächliche Dateihierarchie auf der Festplatte, sondern um eine nach Dateitypen sortierte Ansicht zur vereinfachten Navigation zwischen den Dateien [76].

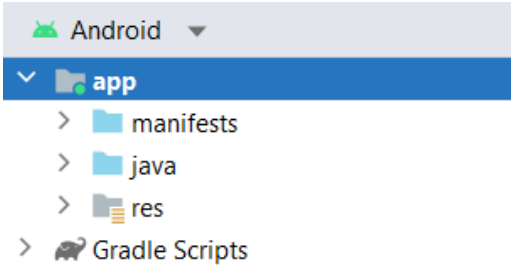


Abbildung 4.2: Screenshot der Projektverzeichnisse in der Android-Ansicht (eigener Screenshot)

Das „manifests“-Verzeichnis besitzt nur die Manifestdatei „AndroidManifest.xml“ (siehe Abschnitt 2.3.4). In dieser sind unter anderem die implementierten *Activities* der App sowie die Internetberechtigung definiert, um die Falldaten herunterladen zu können. Ebenso werden hier das App-Icon und der App-Name über den Zugriff auf App-Ressourcen festgelegt.

Im „java“-Verzeichnis befinden sich alle Java-Klassen. Die Klassen verarbeiten beispielsweise die Funktionalitäten des Layouts und werden in Abschnitt 4.4 detailliert beschrieben.
Das Verzeichnis „res“ beinhaltet die App-Ressourcen, welche in Abschnitt 4.3 erläutert werden.
Im Verzeichnis „Gradle Scripts“ sind alle Gradle-Build-Dateien enthalten, aus welchen das Anwendungsprogramm erstellt wird.

4.2 Setup Firebase

Neben Android Studio musste ebenso Firebase konfiguriert werden. Firebase wurde im Rahmen der vorliegenden Arbeit als Backend-Infrastruktur der Applikation eingerichtet. Zu Beginn wurde die offizielle Firebase-Konsole[37] aufgerufen und ein neues Projekt mit dem Namen „AMBER Germany“ erstellt. Im zweiten Schritt wurde *Google Analytics* dem Projekt hinzugefügt, da dieses verschiedene benötigte Funktionen bereitstellt. Hierbei wurde Deutschland als *Analytics*-Speicherort festgelegt und neben den Nutzungsbedingungen wurden keine Datenfreigabeeinstellungen aktiviert. Als Firebase-Abrechnungstarif wurde der kostenlose „Spark“-Tarif beibehalten.
Als Nächstes musste Firebase der App hinzugefügt werden. Hierfür wurde in der Firebase-Projektübersicht das Android-Symbol angeklickt, der gleiche Paketname wie in Abbildung 4.1 eingegeben und die App registriert. Dabei konnte der App optional ein Nickname (AMBER Germany Android) zugewiesen werden. Anschließend konnte die Konfigurationsdatei „google-services.json“ heruntergeladen und in der Projekt-Ansicht in das Verzeichnis `AMBER > app` eingefügt werden (siehe Abbildung 4.3).

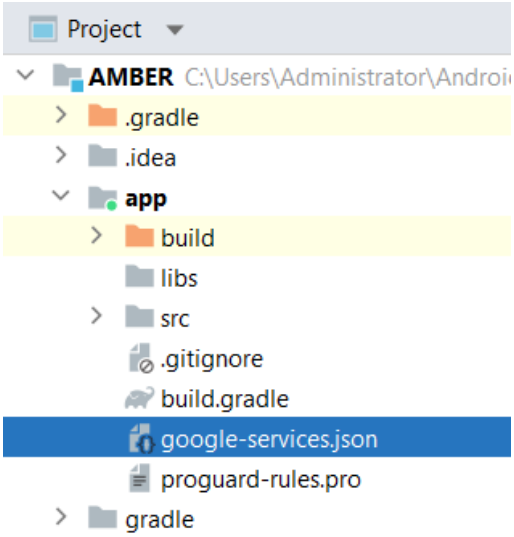


Abbildung 4.3: Screenshot des Pfades der Konfigurationsdatei „google-services.json“ in der Projekt-Ansicht (eigener Screenshot)

[37] `https://console.firebase.google.com/`

Um die Firebase SDKs der App hinzuzufügen, mussten zusätzlich die Gradle-Dateien angepasst werden. So wurden in die **build.gradle-Datei auf Projektebene** die folgenden Code-Zeilen direkt an den Anfang (vor „plugins“) hinzugefügt:

```
buildscript {
  dependencies {
    classpath 'com.google.gms:google-services:4.3.13'
  }
}
```

Die **build.gradle-Datei auf App-Ebene** wurde innerhalb von „plugins“ um folgende Code-Zeile ergänzt:

```
id 'com.google.gms.google-services'
```

Zusätzlich wurden innerhalb der „dependencies“ die zwei folgenden Code-Zeilen hinzugefügt:

```
implementation platform('com.google.firebase:firebase-bom:30.0.1')
implementation 'com.google.firebase:firebase-analytics'
```

Abschließend musste das Projekt über den Button „*Sync now*“ synchronisiert werden.

Nachdem Firebase allgemein der App hinzugefügt wurde, konnte diese im Anschluss um die benötigten Produkte ergänzt werden. Diese Produkte werden im Folgenden erläutert.

4.2.1 Realtime Database

Das erste *Firebase-Build*-Produkt, mit welchem die App verknüpft wurde, ist die *Realtime Database*[38]. Für diese wurde Belgien als Standort und der gesperrte Modus als Startmodus für die Sicherheitsregeln festgelegt, wobei die Regeln in einem der folgenden Schritte noch abgeändert wurden. Um das *Realtime Database* SDK der App hinzuzufügen, musste in der **build.gradle-Datei auf App-Ebene** innerhalb der „dependencies“ folgende Code-Zeile eingefügt und anschließend das Projekt synchronisiert werden:

```
implementation 'com.google.firebase:firebase-database'
```

Nachdem die Datenbank zur App hinzugefügt wurde, konnten die Regeln der *Realtime Database* konfiguriert werden. Hierbei wurden nur die Leserechte erlaubt, um so die in der Datenbank gespeicherten Daten in der App anzeigen lassen zu können. Um zu verhindern, dass die Daten von außerhalb der Firebase-Konsole verändert werden können, wurde keine Schreibberechtigung erteilt.
Die *Realtime Database* wird als Speicherort für die Falldaten verwendet, welche später in der App angezeigt werden. Dabei ist es möglich, die Daten in der Firebase-Konsole

[38] dt. Echtzeit-Datenbank

anzupassen und in Echtzeit in der App anzeigen zulassen. Innerhalb der Datenbank werden die Daten in einer Art JSON-Baumdiagramm gespeichert, wobei die Knoten der ersten Ebene die einzelnen Fälle mittels eines eindeutigen Fallnamens definieren. Die Unterknoten dieser Fallnamen stellen die jeweiligen Fallinformationen dar, welche in der App zum Fall angezeigt werden. Jeder Knoten wird als Schlüssel-Werte-Paar festgelegt. Die Daten können über die Firebase-Konsole direkt in die Datenbank eingespeist oder als JSON-Datei importiert werden. Die Anwendung der *Realtime Database* wird in Kapitel 5 dargestellt.

4.2.2 Cloud Storage

Die zweite zusätzliche Verknüpfung ist der *Cloud Storage*[39]. Für diesen wurde der sog. Produktionsmodus als Startmodus für die Sicherheitsregeln und „eur3 (europe-west)“ als *Cloud-Storage*-Speicherort festgelegt. Weiterhin musste das *Cloud-Storage*-SDK der App hinzugefügt werden. Dafür wurde in der **build.gradle-Datei auf App-Ebene** innerhalb der „dependencies“ folgende Code-Zeile eingefügt:

```
implementation 'com.google.firebase:firebase-storage'
```

Anschließend wurde das Projekt synchronisiert und die Regeln des *Cloud Storages* konfiguriert. Hierbei wurden ebenso nur die Leserechte erlaubt, um die gespeicherten Daten in der App abrufen zu können und die Schreibrechte verweigert.
Der *Cloud Storage* wird zur Speicherung der Bilder der vermissten Kinder genutzt, welche zum jeweiligen Fall in der App angezeigt werden. Hierfür muss das Bild den gleichen eindeutigen Namen wie der zugehörige Knoten in der *Realtime Database* besitzen und eine PNG- oder JPG-Datei sein. Die Bilder können über die Firebase-Konsole in den *Cloud Storage* geladen werden. In Kapitel 5 wird die Anwendung des *Cloud Storages* dargestellt.

4.2.3 Cloud Messaging

Als drittes Firebase-Produkt wurde *Cloud Messaging*[40] hinzugefügt. Hierfür wurde die folgende Code-Zeile in die **build.gradle-Datei auf App-Ebene** innerhalb der „dependencies“ eingefügt:

```
implementation 'com.google.firebase:firebase-messaging'
```

Mithilfe des *Cloud Messaging* können Push-Benachrichtigungen an die Benutzer*innen der App gesendet werden, wenn ein neuer Fall eines vermissten Kindes der App hinzugefügt wurde. Ein Beispiel, wie hiermit eine Benachrichtigung erstellt wird und aussehen könnte, wird in Kapitel 5 erläutert.

[39] dt. Cloud-Speicher
[40] dt. Cloud-Benachrichtigungen

4.3 App-Ressourcen

Das Aussehen einer Android-Applikation wird in den App-Ressourcen definiert (siehe Abschnitt 2.3.5). Dabei gehört jede Datei je nach implementierten Dateieigenschaften einem bestimmten Ressourcentyp an. Die vorliegende App wird aus den folgenden Ressourcentypen aufgebaut: *drawable*, *layout*, *menu*, *mipmap*, *navigation* und *values* (siehe Abbildung 4.4)

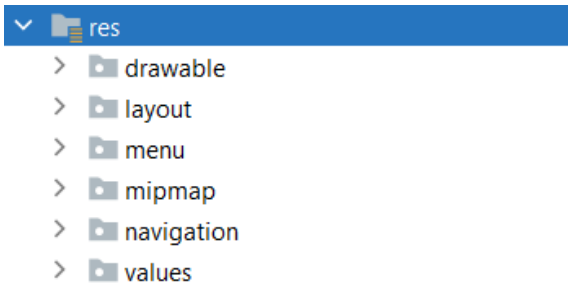


Abbildung 4.4: Screenshot der App-Ressourcen, aus denen die App aufgebaut ist (eigener Screenshot)

4.3.1 drawable

In der *drawable-Ressource* sind die in der App verwendeten Bilder gespeichert. Eine Übersicht über diese Bilder ist in Abbildung 4.5 dargestellt.

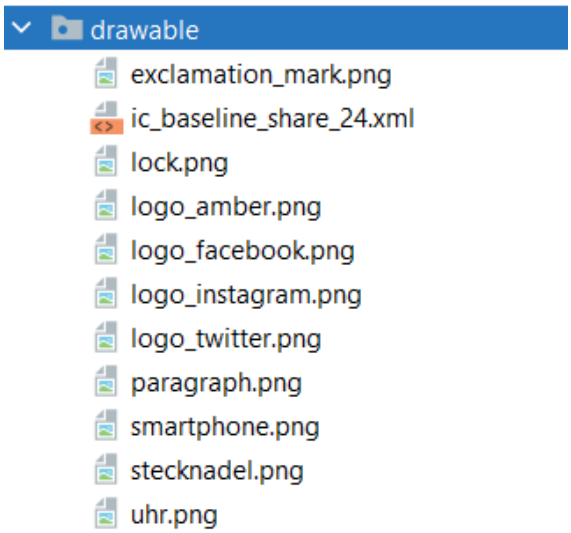


Abbildung 4.5: Screenshot der Bilder (*drawable-Ressource*), welche in der App verwendet werden (eigener Screenshot)

Das Bild **logo_amber.png** wird als App-Icon verwendet und ist in Abbildung 4.6 dargestellt. Dieses Logo wurde im Rahmen der vorliegenden Arbeit erstellt. Der Orangeton wurde gewählt, da er als Signalfarbe dient und somit den Benutzer*innen direkt ins Auge fällt. Aus diesem Grund wird diese Farbe an mehreren Stellen der App verwendet.

Abbildung 4.6: Darstellung des App-Icons (eigene Darstellung)

Auf die Funktion und Verwendung der anderen in der *drawable-Ressource* gespeicherten Bilder wird im nächsten Abschnitt bei den jeweils entsprechenden Dateien der *layout-Ressource* eingegangen.

4.3.2 layout

Die *layout-Ressource* umfasst 14 XML-Dateien, welche in Abbildung 4.7 aufgelistet sind.

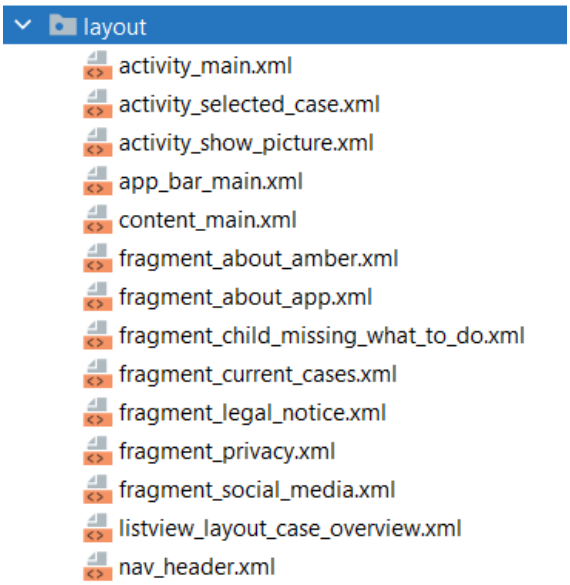
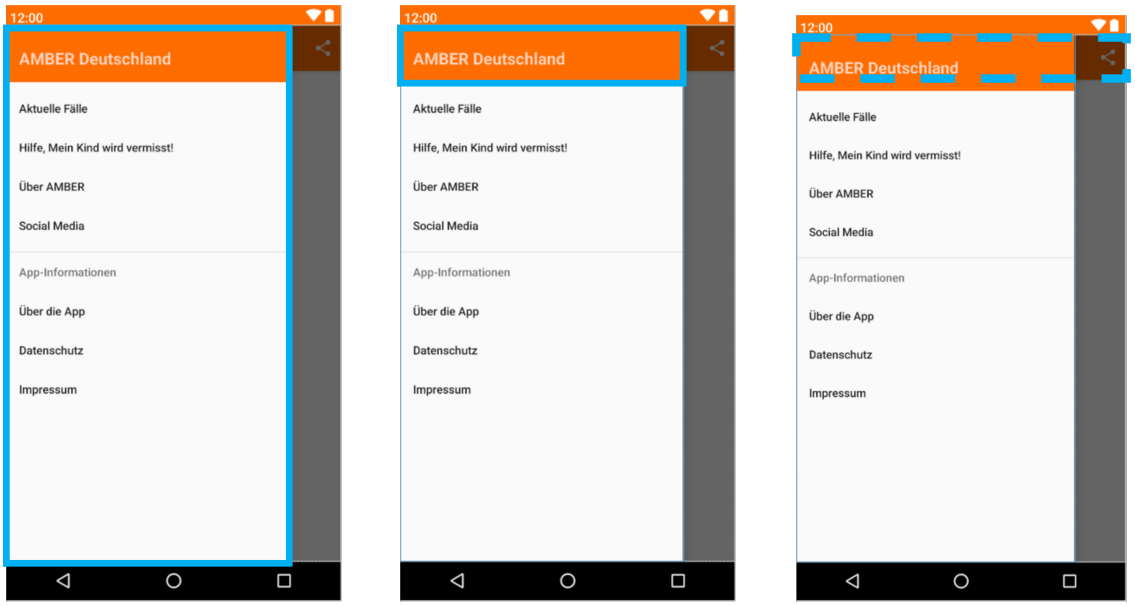


Abbildung 4.7: Screenshot der in der *layout-Ressource* enthaltenen XML-Dateien (eigener Screenshot)

Der Aufbau der Haupt-*Activity* ist in der Datei **activity_main.xml** festgelegt und wird in Abbildung 4.8 dargestellt. Für diese *Activity* wird ein *DrawerLayout* verwendet, welches ein Navigationsmenü (Abbildung 4.8a) mit *Header* (Abbildung 4.8b) und eine obere Navigationsleiste (Abbildung 4.8c) einbindet.
In der Regel ist zu Beginn das Navigationsmenü nicht zu sehen. Dieses kann jedoch vom linken Rand aus wie eine Schublade herausgezogen und somit sichtbar gemacht

werden, wobei anschließend die in Abbildung 4.8c gekennzeichnete Navigationsleiste vom Menü verdeckt wird.
Außerdem bindet die *Activity* über die Berührung der einzelnen Menüpunkte entsprechende Fragmente ein, welche als *Destinations* des Navigationsgraphen festgelegt wurden.



(a) Navigationsmenü (b) Navigationsmenü-Header (c) Navigationsleiste

Abbildung 4.8: Drei Screenshots vom Aufbau der Datei **activity_main.xml**, mit jeweils blau gekennzeichnetem Navigationsmenü, -header und -leiste (eigene Abbildung)

Das Aussehen des Navigationsmenüs (ohne Header) wird in der *menu-Ressource* definiert und ist in Abschnitt 4.3.3 beschrieben.

In der Datei **nav_header.xml** wird das Aussehen des Headers des Navigationsmenüs festgelegt. Dabei wurde für den Hintergrund ebenfalls der Orangeton des Logos aus Abbildung 4.6 gewählt und es wird ein einzelnes TextView mit der Beschriftung „AMBER Deutschland“ verwendet (siehe Abbildung 4.8b).

Das Aussehen der Navigationsleiste ist in der Datei **app_bar_main.xml** festgelegt und wird in Abbildung 4.9 dargestellt. Die XML-Datei besteht aus einem *CoordinatorLayout*, welches wiederum ein *AppBarLayout* verwendet und so die Funktionen für die Navigationsleiste bereitstellt. Hierdurch wird zum einen die *Toolbar* mit einem Navigationsbutton zum Wechseln durch die *Destinations* (grüne Kennzeichnung) und zum anderen das Aussehen des *Share-Buttons*[41] (blaue Kennzeichnung) implementiert. Der *Share-Button* wurde am rechten Rand der Navigationsleiste als *ImageView* eingefügt und zeigt die in der *drawable-Ressource* hinterlegte Datei **ic_baseline_share_24.xml**. Diese Datei ist ein von Android Studio zur Verfügung gestelltes Icon.

[41] dt. Teilen-Button

Abbildung 4.9: Aussehen der Navigationsleiste mit gekennzeichnetem Navigations- (grün) und Share-Button (blau) (eigene Abbildung)

Das *CoordinatorLayout* bindet neben dem *AppBarLayout* zusätzlich die Datei **content_main.xml** ein, welche als Navigationshost dient. Ein Navigationshost ist ein leerer Container, in welchem die *Destinations*, die im Navigationsgraph (siehe Abschnitt 4.3.5) definiert sind, ein- bzw. ausgewechselt werden können, während ein*e Benutzer*in durch die App navigiert [42]. Einfacher ausgedrückt wird so sichergestellt, dass über die Berührung eines Menüpunktes im Navigationsmenü der zum Menüpunkt gehörende Bildschirm angezeigt wird.

Die Datei **fragment_current_cases.xml** definiert das Aussehen der ersten *Destination* des Navigationsgraphen, welche über Berührung des Menüpunktes „Aktuelle Fälle“ im Navigationsmenü (siehe Abbildung 4.8a) aufgerufen wird. Außerdem ist dieses Fragment gleichzeitig die *Start-Destination*, also der erste Bildschirm, den die Nutzer*innen sehen, wenn sie die App öffnen. Dabei handelt es sich um eine Übersicht aller aktuellen Fälle von vermissten Kindern, welche in die App aufgenommen wurden. Diese Übersicht ließ sich mittels eines *FrameLayouts* und *ListViews* implementieren.
In Abbildung 4.10 ist der Startbildschirm der App mit vier fiktiven Beispielfällen zu sehen.

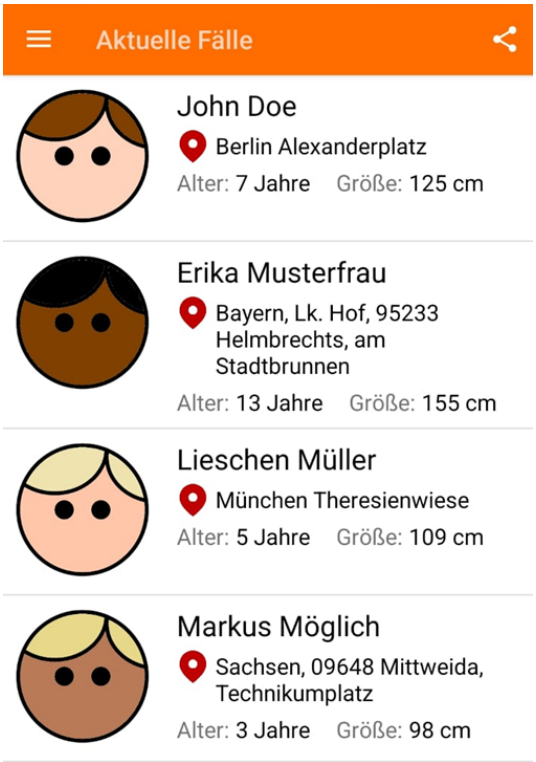


Abbildung 4.10: Screenshot der *Start-Destination* mit vier Beispielfällen (eigener Screenshot)

In Abbildung 4.10 wird deutlich, dass jeder Listeneintrag (*View*) nach dem gleichen Schema (siehe Abbildung 4.11) aufgebaut ist. Dieses Schema basiert auf einem *RelativeLayout* und ist in der Datei **listview_layout_case_overview.xml** festgelegt. Am linken Rand wurde ein *ImageView* eingefügt, in welchem ein Bild vom vermissten Kind angezeigt werden kann. Weiterhin wurden sechs *TextViews* für die wichtigsten Informationen über das Kind angelegt (Name, Ort, wo das Kind zuletzt gesehen wurde, Alter und Größe). Zwei *TextViews* sind mit nicht veränderbaren Strings (dunkleres Grau) gefüllt und vier mit durch den Fall beeinflussbaren Strings (helleres Grau). Außerdem wurde ein weiteres *ImageView* mit einem Stecknadelsymbol als Eye-Catcher vor das *TextView* des zuletzt gesehenen Ortes eingefügt. Das Stecknadelsymbol ist in der *drawable-Ressource* unter dem Namen **stecknadel.png** hinterlegt.

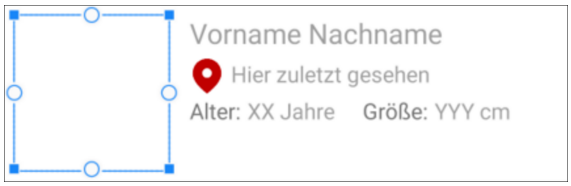


Abbildung 4.11: Darstellung des *View*-Schemas des *ListViews* in der Datei **fragment_current_cases.xml** (eigene Darstellung)

Durch das Berühren einer der Fälle in der Fallübersicht wird zu einer neuen *Activity* gesteuert, welche die Datei **activity_selected_case.xml** einbindet. Diese Ansicht zeigt detaillierte Fallinformationen zum jeweils ausgewählten Fall an. Die benötigten Informationen, welche über das Kind in der App enthalten sein sollten, wurden mithilfe von Carlo Schippers (AMBER Alert Europe) herausgearbeitet (siehe Anhang C). Dabei sollten grundlegende Informationen (Name und Alter) sowie körperliche Merkmale (Haarfarbe, Haarlänge, Augenfarbe, Größe usw.) angezeigt werden. Ebenso sollten die Umstände des Verschwindens dargestellt werden, z. B. wann und wo das Kind zuletzt gesehen wurde. Außerdem soll eine Möglichkeit geboten werden, wie Nutzer*innen Rückmeldungen zu Hinweisen oder zum Verbleib des Kindes geben können (z. B. Telefonnummer der zuständigen Polizei).
Die XML-Datei dieser *Activity* ist aus einem *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout* aufgebaut. Innerhalb dieses Hauptlayouts wurde ein *ImageView* für eine vergrößerte Ansicht des Bildes vom vermissten Kind eingefügt. Weiterhin sind 14 *TextViews* mit nicht veränderbaren Strings (grau und blau) und 14 mit fallabhängigen Strings (schwarz und grün) zu sehen. Die Inhalte der Strings sind entsprechende Fallinformationen, wobei die Texte für die fallabhängigen Strings über die *Realtime Database* geladen werden. Bei dem *TextView* mit der Telefonnummer der zuständigen Polizei wurde zusätzlich das Attribut „autoLink > phone“ aktiviert, sodass durch Berührung der Telefonnummer direkt die Telefon-System-App geöffnet wird. Außerdem wurden einige der *TextViews* innerhalb von insgesamt sieben weiteren *ConstraintLayouts*, welche Teil des Hauptlayouts sind, implementiert. Dadurch konnten die *TextViews* flexibler und somit optisch übersichtlicher angeordnet werden. Zusätzlich wurden die Dateien **stecknadel.png** sowie **uhr.png** aus der *drawable-Ressource* als Eye-Catcher vor die *TextViews*

„Zuletzt gesehen“ und „Vermisst seit“ eingefügt.
In Abbildung 4.12 wird ein Screenshot dieser *Activity* mit Fallinformationen zu einem der fiktiven Beispielfälle dargestellt.

Abbildung 4.12: Screenshot der *Activity* **activity_selected_case.xml** mit Fallinformationen zu einem ausgewählten Beispielfall (eigener Screenshot)

Wird innerhalb der *Activity* **activity_selected_case.xml** das *ImageView* mit dem Bild des vermissten Kindes berührt, wird eine neue *Activity* aufgerufen, welche die Datei **activity_show_picture.xml** einbindet. Diese Ansicht zeigt das Bild mithilfe eines *ConstraintLayouts* und *ImageViews* im Vollbildmodus mit schwarzem Hintergrund an.

Die Datei **fragment_child_missing_what_to_do.xml** definiert das Layout einer weiteren *Destination* des Navigationsgraphen, welche über die Berührung des Menüpunktes „Hilfe, mein Kind wird vermisst!“ im Navigationsmenü (siehe Abbildung 4.8a) aufgerufen wird. Die XML-Datei besteht aus einem *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout*. Am oberen Rand wurde ein *ImageView* mit der Datei **exclamation_mark.png** aus der *drawable-Ressource* als Eye-Catcher zur optischen Verschönerung eingefügt. Weiterhin sind 18 *TextViews* mit nicht veränderbaren Strings zu sehen, welche Texte zu Handlungsmaßnahmen enthalten, wenn das eigene Kind vermisst wird. Sechs der *TextViews* wurden zusätzlich innerhalb von insgesamt drei horizontalen *Li-*

nearLayouts implementiert, welche ebenso Teil des Hauptlayouts sind.
Neben den allgemeinen Handlungsmaßnahmen wurden ebenso *TextViews* mit wichtigen Telefonnummern und einem Link zu weiterführenden Informationen hinzugefügt. So wurde bei drei *TextViews* das Attribut „autoLink > phone“ aktiviert, um bei Berührung der jeweiligen Telefonnummer direkt zur Telefon-System-App zu gelangen und bei einem das Attribut „autoLink > web“, um bei Berührung des Links auf die entsprechende Internetseite in einem Browser zu gelangen.
In Abbildung 4.13 ist ein Screenshot des Fragments bzw. der *Destination* dargestellt. Dabei sind nicht alle verwendeten *View*-Objekte direkt zu sehen, jedoch kann aufgrund des implementierten *ScrollViews* innerhalb der Applikation gescrollt und somit alle *TextViews* sichtbar gemacht werden. Alle im Fragment verwendeten Texte werden im Anhang B.1 vollständig dargestellt.

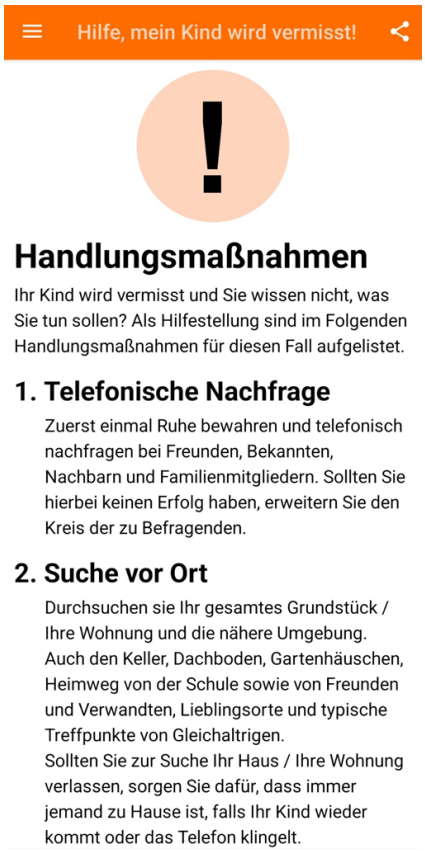


Abbildung 4.13: Screenshot der *Destination* **fragment_child_missing_what_to_do.xml** (eigener Screenshot)

In der Datei **fragment_about_amber.xml** wird das Aussehen der *Destination* definiert, welche allgemeine Informationen über das AMBER-Alert-System bereithält. Es wird durch die Berührung des Menüpunktes „Über AMBER“ im Navigationsmenü (siehe Abbildung 4.8a) aufgerufen und besteht aus einem *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout*. Innerhalb dieser *ViewGroup*-Objekte wurden neun *TextViews* mit nicht veränderbaren Strings implementiert, welche entsprechende Informationen zum Hintergrund des AMBER-Alerts anzeigen. Drei dieser *TextViews* enthalten Links

zu weiterführenden Internetseiten, weshalb bei diesen das Attribut „autoLink > web“ aktiviert wurde. Somit ist es möglich, bei Berührung einer dieser Links auf die entsprechende Internetseite in einem Browser zu gelangen.
In Abbildung 4.14 ist ein Screenshot dieses Fragmentes bzw. der *Destination* dargestellt.

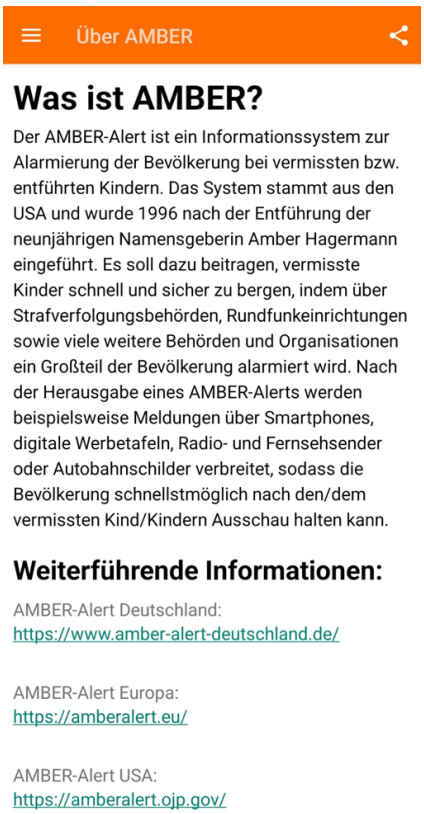


Abbildung 4.14: Screenshot der *Destination* **fragment_about_amber.xml** (eigener Screenshot)

Die XML-Datei **fragment_social_media.xml** definiert das Aussehen für die *Destination*, welche Informationen zur Social-Media-Präsenz des deutschen AMBER-Alerts bereithält. Das Fragment lässt sich über Berührung des Menüpunktes „Social Media“ im Navigationsmenü (siehe Abbildung 4.8a) anzeigen und ist aus einem *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout* aufgebaut. Darin sind zwei *TextViews* mit nicht veränderbaren Strings enthalten, welche allgemeine Informationstexte zur Social-Media-Präsenz bereitstellen. Weiterhin umfasst das Hauptlayout ein horizontales *LinearLayout* mit drei *ImageViews*. Diese *ImageViews* zeigen das Facebook[42]-, Instagram[43]- und Twitter[44]-Logo, welche in der *drawable-Ressource* als **logo_facebook.png**, **logo_instagram.png** sowie **logo_twitter.png** hinterlegt sind. Diese *ImageViews* dienen als Buttons zu gängigen Social-Media-Plattformen.
Ein Screenshot der *Destination* ist in Abbildung 4.15 zu sehen.

[42] `https://www.facebook.com/brand/resources/facebookapp/logo`
[43] `https://www.facebook.com/brand/resources/instagram/icons`
[44] `https://about.twitter.com/en/who-we-are/brand-toolkit`

Abbildung 4.15: Screenshot der *Destination* **fragment_social_media.xml** (eigener Screenshot)

Durch das Berühren des Menüpunktes „Über die App“ im Navigationsmenü (siehe Abbildung 4.8a) wird die Datei **fragment_about_app.xml** eingebunden. Diese beschreibt das Aussehen für die *Destination*, welche Informationen rund um die App bereithält. Sie umfasst ein *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout*, in welches ein *ImageView* und drei *TextViews* eingefügt wurden. Das *ImageView* ist am oberen Rand und dient mit der Datei **smartphone.png** aus der *drawable-Ressource* als Eye-Catcher. In den darunter befindlichen *TextViews* wird beschrieben, weshalb die App bei der Suche nach vermissten Kindern sinnvoll ist. Zudem wurde außerhalb des *LinearLayouts*, aber noch innerhalb des *ScrollViews*, ein weiteres *TextView* hinzugefügt, welches die aktuelle App-Version anzeigt.
Abbildung 4.16 zeigt einen Screenshot dieser *Destination*.

Abbildung 4.16: Screenshot der *Destination* **fragment_about_app.xml** (eigener Screenshot)

Um die Datenschutzerklärung anzeigen zu lassen, muss der Menüpunkt „Datenschutz“ im Navigationsmenü (siehe Abbildung 4.8a) berührt werden. Anschließend wird das in der Datei **fragment_privacy.xml** definierte Aussehen des Fragments angezeigt. Diese *Destination* ist aus einem *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout* aufgebaut. Innerhalb dieses Layouts sind ein *ImageView* und 46 *TextViews* zu finden. Das *ImageView* befindet sich am oberen Rand des Fragments und zeigt das Bild **lock.png** aus der *drawable-Ressource*. Das Bild dient wieder als Eye-Catcher. Die *TextViews* sind unter dem *ImageView* zu sehen und beinhalten die Datenschutzerklärung. Der Text der Datenschutzerklärung wurde mittels eRecht24[45] erstellt. Da im Rahmen der vorliegenden Arbeit die Applikation nur prototypisch implementiert wurde, sind einige der *TextViews* (z. B. „Hinweis zur verantwortlichen Stelle“) mit sog. Dummy-Daten gefüllt.
In Abbildung 4.17 ist ein Screenshot der Datenschutz-*Destination* dargestellt. Dabei sind nicht alle der verwendeten *View*-Objekte zu sehen, weshalb der gesamte Text der Datenschutzerklärung im Anhang B.2 abgebildet ist. Innerhalb der App kann aufgrund des verwendeten *ScrollViews* durch die App gescrollt und somit alle *View*-Objekte sichtbar gemacht werden.

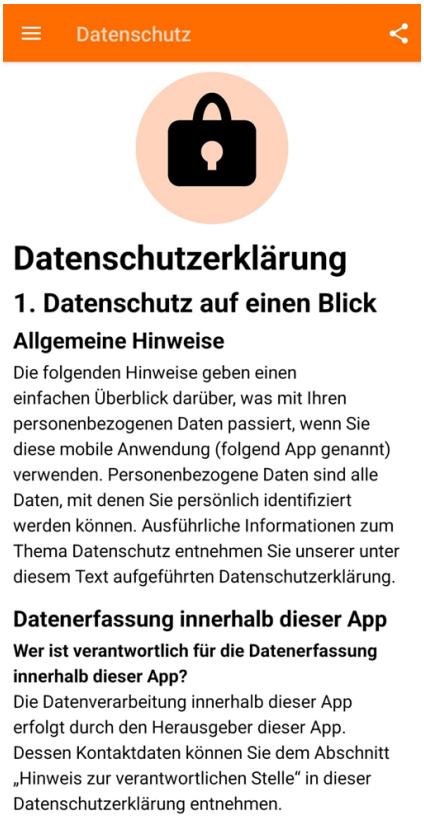


Abbildung 4.17: Screenshot der *Destination* **fragment_privacy.xml** (eigener Screenshot)

[45] `https://www.e-recht24.de`

Die letzte *Destination* lässt sich über den Menüpunkt „Impressum“ im Navigationsmenü (siehe Abbildung 4.8a) anzeigen. Dabei wird die XML-Datei **fragment_legal_notice.xml** eingebunden, welche das Impressum sichtbar macht. Die Datei umfasst ein *ConstraintLayout* mit *ScrollView* und vertikalem *LinearLayout*. Innerhalb des Layouts wurde am oberen Rand ein *ImageView* mit der *drawable-Ressource* **paragraph.png** als Eye-Catcher eingefügt. Darunter sind 28 *TextViews* mit dem ausgeschriebenen Impressum zu sehen. Der Haftungsausschluss des Impressums wurde dabei mittels eRecht24[46] erstellt. Zehn der *TextViews* befinden sich zusätzlich in insgesamt fünf verschiedenen horizontalen *LinearLayouts*, welche ebenso Teil des Hauptlayouts sind. Bei dem *TextView* mit der Telefonnummer wurde zusätzlich das Attribut „autoLink > phone“ aktiviert, um bei Berührung der Telefonnummer zur Telefon-System-App zu gelangen. Bei dem *TextView* mit der E-Mail-Adresse wurde das Attribut „autoLink > email“ aktiviert, um bei Berührung der E-Mail-Adresse zu einer E-Mail-App zu gelangen. Da es sich im Rahmen der vorliegenden Arbeit nur um eine prototypische Implementierung handelt, wurden einige der *TextViews* (z. B. „Kontakt“ und „Registereintrag“) mit Dummy-Daten gefüllt.
Abbildung 4.18 zeigt einen Screenshot des Impressums. Dabei sind die *View*-Objekte mit dem Haftungsausschluss nicht zu sehen. Jedoch kann durch das implementierte *ScrollView* innerhalb der Applikation gescrollt und somit die Texte angezeigt werden. Der gesamte Text des Impressums ist im Anhang B.3 dargestellt.

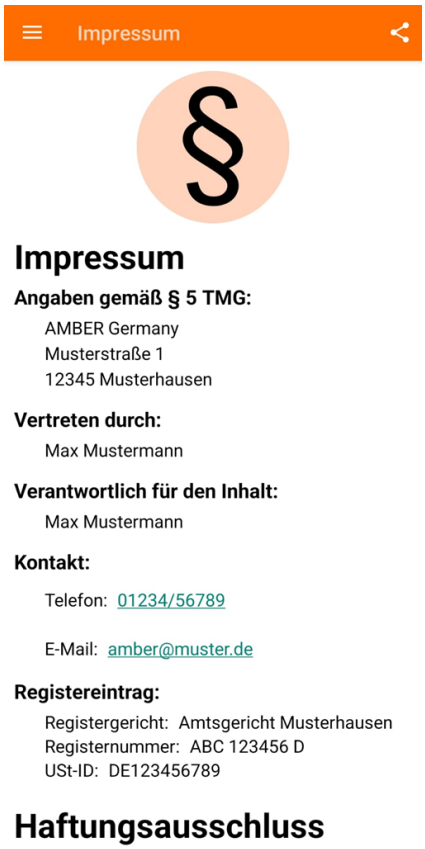


Abbildung 4.18: Screenshot der *Destination* **fragment_legal_notice.xml** (eigener Screenshot)

[46] `https://www.e-recht24.de/muster-disclaimer.html`

4.3.3 menu

Die *menu-Ressource* enthält nur die XML-Datei **navigation_menu.xml**. In dieser wird das Aussehen des Navigationsmenüs (siehe Abbildung 4.8a) ohne Header beschrieben. Es werden also die einzelnen Menüpunkte und ihre Reihenfolge im Navigationsmenü definiert. Neben einem Titel bekommen die Menüpunkte zusätzlich eine ID zugewiesen, welche den IDs der Elemente in der *navigation-Ressource* (siehe Abschnitt 4.3.5) entsprechen. Das stellt sicher, dass die Menüpunkte mit der richtigen *Destination* verbunden sind und somit bei Berührung eines Menüpunktes das richtige Fragment angezeigt wird.

4.3.4 mipmap

In Abbildung 4.19 sind die in der *mipmap-Ressource* enthaltenen Dateien abgebildet. In der *Ressource* sind verschiedene Versionen des Logos aus Abbildung 4.6 enthalten. Die Versionen unterscheiden sich zum einen in der Abrundung der Ecken und zum anderen in ihrer Pixeldichte. Das stellt sicher, dass das Icon auf verschiedenen Geräten mit unterschiedlichen Bildschirmauflösungen und Android-Versionen klar erkennbar ist. Die verschiedenen Versionen konnten per Rechtsklick auf „Project > app“ und anschließender Auswahl von „New > Image Asset“ automatisch erzeugt werden. Dabei wurden alle Voreinstellungen bis auf die Hintergrundfarbe beibehalten. Als Hintergrundfarbe wurde wieder der Orangeton aus Abbildung 4.6 gewählt.

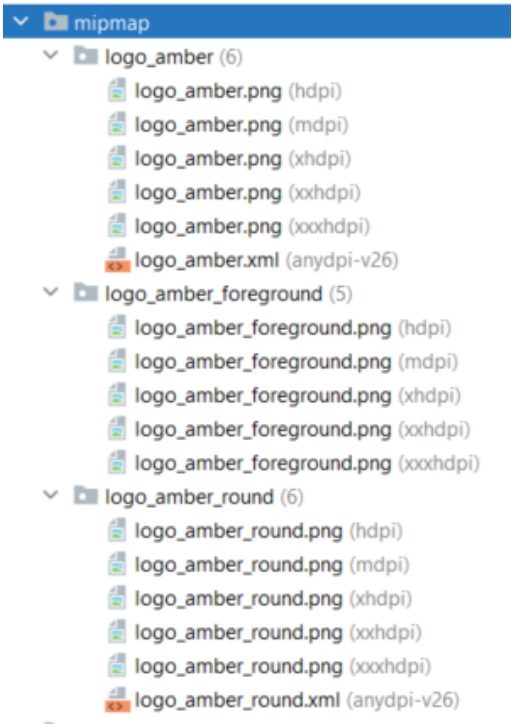


Abbildung 4.19: Screenshot der in der *mipmap-Ressource* enthaltenen Dateien (eigener Screenshot)

4.3.5 navigation

Die *navigation-Ressource* umfasst die Datei **mobile_navigation.xml**. Diese Datei dient als Navigationsgraph, welcher alle *Destinations* enthält, auf welche die Nutzer*innen über das Navigationsmenü zugreifen können. Hierfür wurde jeder *Destination* ein Fragment zugewiesen, welches den Nutzer*innen, je nach berührtem Menüpunkt, angezeigt wird.
In Abbildung 4.20 ist die Design-Übersicht des Graphen zu sehen. Dabei zeigt die blaue Markierung den festgelegten Navigationshost, der die *Destinations* ein- und auswechselt, je nachdem welcher Menüpunkt von den Nutzer*innen berührt wird. Die rote Umrandung listet alle festgelegten *Destinations* des Graphen auf. Bei den angezeigten Namen handelt es sich um die zugewiesenen IDs, welche den IDs der Elemente in der *menu-Ressource* entsprechen. So wird sichergestellt, dass die *Destinations* mit den richtigen Menüpunkten verbunden sind. Die grüne Markierung stellt die *Destinations* des Navigationsgraphen grafisch dar. Dabei wird deutlich, dass jede *Destination* einem Fragment entspricht. Das Haus-Symbol vor der *Destination* „nav_all_cases“ bedeutet, dass diese *Destination* die *Start-Destination* ist, also der erste Bildschirm, den die Nutzer*innen beim Öffnen der App sehen.

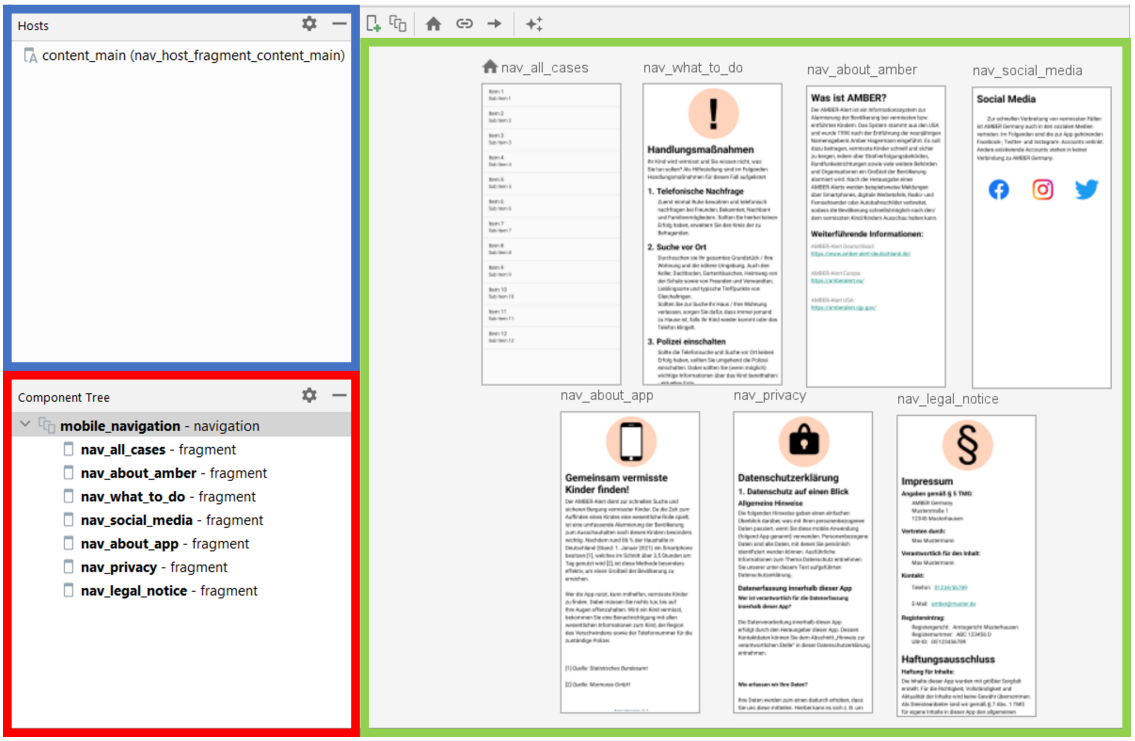


Abbildung 4.20: Darstellung der Design-Übersicht des Navigationsgraphen mit markiertem Host (blau) sowie markierter Auflistung (rot) und grafischer Darstellung (grün) der *Destinations* (eigene Darstellung)

4.3.6 values

Die letzte verwendete App-Ressource ist die *values-Ressource*. In dieser befinden sich verschiedene XML-Dateien, welche Daten unterschiedlicher Eigenschaften beinhalten. Abbildung 4.21 zeigt die in der *values-Ressource* enthaltenen Dateien.

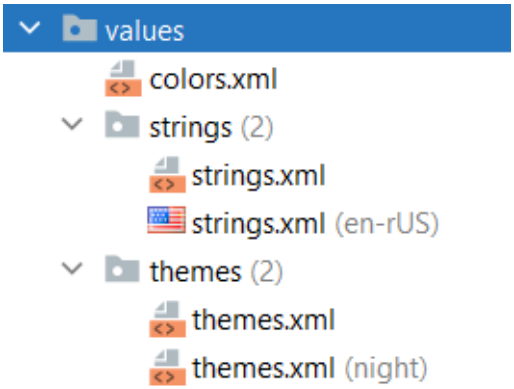


Abbildung 4.21: Screenshot der in der *values-Ressource* enthaltenen Dateien (eigener Screenshot)

Die Datei **colors.xml** umfasst die in der App verwendeten Farbwerte, z. B. den Wert „#FFFF6D00“. Bei diesem Wert handelt es sich um den im Rahmen der App verwendeten Orangeton (siehe Abbildung 4.6). Durch die Definition der Farbe an einem zentralen Ort kann der gleiche Farbton über eine ID an unterschiedlichen Stellen der App genutzt werden.
Das Unterverzeichnis „strings“ enthält zwei XML-Dateien. In der Datei **strings.xml** sind alle im Rahmen der App verwendeten Texte für die deutschsprachige Variante der App definiert. Dabei besitzt jeder String eine Ressourcen-ID, mit welcher er den entsprechenden *TextViews* im Layout der App zugeordnet werden kann. Die Datei **strings.xml (en-rUS)** umfasst die gleichen Strings aus der ersten XML-Datei, übersetzt ins Englische. Die englischen Strings verfügen dabei über die gleichen Ressourcen-IDs wie die deutschen Strings. Dadurch werden die Texte in der App automatisch in Englisch angezeigt, wenn Nutzer*innen die Sprache des Android-Systems auf Englisch gestellt haben. So ist die App ebenfalls von Personen nutzbar, die kein Deutsch sprechen bzw. lesen können.
Die *themes*-Dateien bestimmen das allgemeine Design, welches auf die gesamte App angewendet wird, z. B. Schrift- und Hintergrundfarben. In der Datei **themes.xml** wird das standardmäßig verwendete Design festgelegt, während in der Datei **themes.xml (night)** das Design für den *Dark Mode* definiert ist.

4.4 Java-Klassen

Neben der Definition des Layouts der Applikation müssen ebenso die Funktionalitäten zum Verarbeiten der Interaktivitäten durch die Benutzer*innen umgesetzt werden. Im Rahmen der vorliegenden Arbeit werden die Funktionalitäten mithilfe von Java sichergestellt. Hierfür wurden 13 Java-Klassen erstellt, wobei jede Klasse verschiedene Funktionalitäten implementiert. In Abbildung 4.22 sind der Pfad sowie die erstellten Java-Klassen dargestellt.

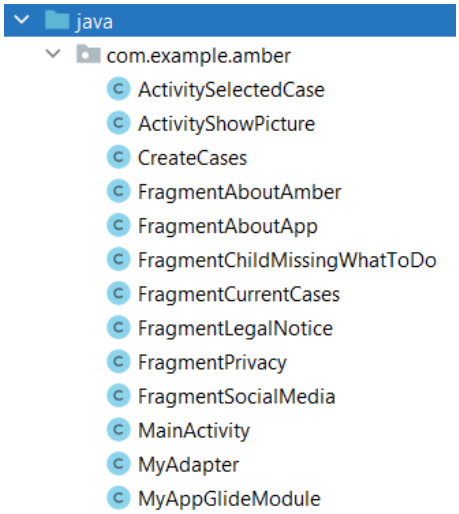


Abbildung 4.22: Screenshot der erstellten Java-Klassen (eigener Screenshot)

MainActivity

Die Datei **MainActivity.java** enthält den Quellcode für die *Activity*, welche zuerst geladen wird. Dabei wird der Großteil der Klasse in der *Callback*-Methode `onCreate()` implementiert:
Zu Beginn wird die Methode `createCases()` der Klasse **CreateCases** (siehe nächster Abschnitt) aufgerufen, um die in Firebase hinterlegten Fälle zu laden. Weiterhin werden die Fragmente, die Navigationsleiste sowie das Navigationsmenü in die *Activity* eingebunden. Außerdem werden die Navigationsleisten-Einstellungen konfiguriert. Dabei wurden über die IDs der *Destinations* (siehe Abschnitt 4.3.5) die sog. *Top-Level-Destinations* festgelegt. *Top-Level-Destinations* sind *Destinations* der höchsten Ebene, bei welchen kein sog. „*Up Button*“ bzw. keine Aufwärts-Schaltfläche in der Navigationsleiste anzeigt wird [77].
Anschließend wird die Funktionalität des in der Navigationsleiste eingefügten *Share-Buttons* implementiert. So kann durch Berührung des Buttons ein im Quellcode definierter Text geteilt werden[47]. Dieser Text könnte beispielsweise nach Veröffentlichung der App der Link zur App im Google Play Store sein. In Abbildung 4.23 wird der Startbildschirm der App nach Berührung des *Share-Buttons* dargestellt.

[47] `https://developer.android.com/training/sharing/send.html#send-text-content`

Außerhalb der `onCreate()`-Methode wird eine von Google bereitgestellte Methode implementiert, um das Navigationsmenü nutzen zu können[48].

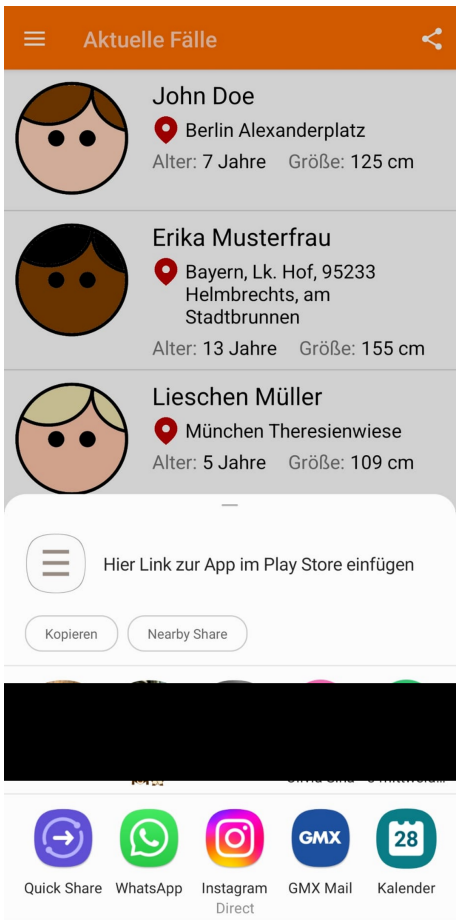


Abbildung 4.23: Screenshot des Startbildschirms der App nach Berührung des *Share-Buttons* mit geschwärzten Kontaktdaten (eigener Screenshot)

CreateCases

Die Klasse **CreateCases** implementiert den Firebase-Zugriff, um somit Fälle erstellen, ändern und löschen zu können. Hierfür werden zu Beginn die Firebase Referenzen für die *Realtime Database* und den *Cloud Storage* sowie 15 *ArrayLists* deklariert. In einer *ArrayList* sollen die eindeutigen Fallnamen (*Keys*) gespeichert werden und in den anderen 14 die entsprechenden Informationen zum Fall. Dabei hat jede Fallinformation eine eigene Nummer zugewiesen bekommen, mit welcher sie in Firebase abgespeichert ist. Über diese Nummern kann innerhalb der Klasse auf die jeweiligen Informationen zugegriffen werden. Welche Nummer zu welcher Fallinformation gehört, wird detailliert in Tabelle 5.1 im nächsten Kapitel dargestellt.
Die Klasse enthält nur die Methode `createCases()`, welche in Firebase Fälle erstellen, ändern und löschen kann. Hierzu muss zu Beginn der Zugriff auf die Firebase-Datenbank und den -*Storage* implementiert werden, was über folgende Code-Zeilen

[48] `https://developer.android.com/guide/navigation/navigation-ui#action_bar`

erfolgt. Die zweite Code-Zeile behandelt zusätzlich vorübergehende Netzwerkunterbrechungen des Smartphones, indem zwischengespeicherte Daten geladen werden, solange bis der*die Nutzer*in wieder online ist.

```
database = FirebaseDatabase.getInstance("https://amber-germany-
    default-rtdb.europe-west1.firebasedatabase.app/");
database.setPersistenceEnabled(true);
myDatabase = database.getReference();

storage = FirebaseStorage.getInstance();
myStorage = storage.getReference();
```

Nach der Implementierung des Firebase-Zugriffs, wird auf die Datenbank die Sortierfunktion `orderByChild` angewendet, um bei einer Datenbankabfrage die Ergebnisse nach einem bestimmten Objekt (sog. *Child*) zu sortieren. Hierbei werden die Daten nach dem Vermisst-Seit-Datum (Nummer 05) sortiert, um die Fälle nach Aktualität zu ordnen. Weiterhin wird der Datenbankreferenz ein `ChildEventListener()` hinzugefügt. Dieser behandelt die verschiedenen Datenbank-Ereignisse, also ob ein Objekt hinzugefügt, geändert oder gelöscht wurde, ob sich seine Priorität verändert hat oder ob der *Listener* fehlgeschlagen ist.
Wurde in der Datenbank ein Fall hinzugefügt, wird sein Fallname in der entsprechenden *ArrayList* an der Position 0 gespeichert. So wird sichergestellt, dass der aktuellste Fall am weitesten oben im *ListView* der *Start-Destination* angezeigt wird. Anschließend wird geprüft, ob in Firebase schon die benötigten Fallinformationen (z. B. Name, Alter, Größe usw.) zum Fall existieren. Trifft das zu, werden diese Fallinformationen in die jeweiligen Arrays an Position 0 geladen. Wurden noch keine Fallinformationen zum Fall in Firebase eingetragen, passiert nichts, da in der App kein Fall mit „leeren“ Informationen angezeigt werden soll.
Werden Informationen eines bereits angelegten Falls verändert, wird zu Beginn geprüft, ob der Fall noch bzw. schon alle benötigten Fallinformationen enthält, um keinen Fall mit „leeren“ Informationen in der App anzuzeigen. Existieren alle Fallinformationen, werden diese heruntergeladen und zwischengespeichert. Im Anschluss wird überprüft, ob in den *ArrayLists* schon etwas zum jeweiligen Fall eingetragen ist. Wurde noch nichts zum Fall eingetragen, werden die zwischengespeicherten Informationen an Position 0 der entsprechenden *ArrayLists* gespeichert. Andernfalls werden die Informationen an der entsprechenden Stelle der *ArrayLists* überschrieben.
Wird in der Datenbank ein Fall gelöscht, wird der Fallname aus der entsprechenden *ArrayList* herausgelöscht. Zusätzlich wird geprüft, ob der Fall die benötigten Fallinformationen enthielt. Trifft das zu, werden diese Informationen ebenfalls aus den entsprechenden *ArrayLists* gelöscht.
Nach Behandlung dieser Datenbank-Ereignisse wird jedes Mal der Adapter, welcher das *ListView* der Klasse **FragmentCurrentCases** füllt, aktualisiert.
Die Datenbank-Ereignisse, ob sich eine Priorität verändert hat oder ob der *Listener* fehlgeschlagen ist, wurden nicht behandelt.
Die Methode `createCases()` wird nur in der Klasse **MainActivity** aufgerufen.

FragmentCurrentCases

In der Datei **FragmentCurrentCases.java** ist der Quellcode für das (Start-)Fragment enthalten, welches die aktuellen Fälle in der App als geordnete Liste anzeigt. Hierbei werden alle Fälle dargestellt, welche in der Firebase-Datenbank mit Fallnamen und den benötigten Fallinformationen hinterlegt sind.
Zu Beginn wird ein Adapter der Klasse **MyAdapter** deklariert, welcher die Listeneinträge mit den entsprechenden Fallinformationen nach einem bestimmten Schema füllen soll. Anschließend wird in der *Callback*-Methode `onCreateView()` die Datei **fragment_current_cases.xml** aus der *layout-Ressource* und das darin enthaltene *ListView* eingebunden sowie der Adapter angewendet. Hierfür werden ihm die entsprechend benötigten *ArrayLists* der Klasse **CreateCases** übergeben. Die genaue Funktionsweise des Adapters wird im nächsten Abschnitt beschrieben.
Zum Schluss wird dem *ListView* ein `OnItemClickListner()` hinzugefügt, welcher das Ereignis behandelt, wenn einer der Listeneinträge berührt wird. Geschieht das, wird eine neue *Activity* der Klasse **ActivitySelectedCase** aufgerufen, welche detaillierte Fallinformationen zum berührten Fall angezeigt. Hierfür werden die Informationen zum entsprechenden Fall aus den *ArrayLists* der Klasse **CreateCases** übergeben.

MyAdapter

Die Klasse **MyAdapter** fungiert als Adapter. Dieser füllt die Listeneinträge des *ListViews* der Klasse **FragmentCurrentCases** nach einem bestimmten Schema. Es wird also festgelegt, wie und mit welchen Daten ein Listeneintrag gefüllt wird.
Bei dieser Klasse handelt es sich um einen abgewandelten *ArrayAdapter*[49], welcher die *ArrayLists* mit den entsprechend benötigten Fallinformationen übergeben bekommt.
In der *Callback*-Methode `getView()` wird das definierte Aussehen eines Listeneintrags aus der Datei **listview_layout_case_overview.xml** eingebunden. Anschließend werden die darin enthaltenen *TextViews* und das *ImageView* geladen sowie die *TextViews* mit den entsprechenden, zum jeweils gleichen Fall gehörenden Fallinformationen gefüllt.
Um das *ImageView* füllen zu können, wird „*FirebaseUI for Storage*“[50] und „*Glide*“[51] verwendet. Mithilfe dessen können Bilder schnell und einfach aus dem Firebase *Cloud Storage* heruntergeladen und in einem definierten *ImageView* angezeigt werden [78].
Hierfür mussten die folgenden Code-Zeilen in die **build.gradle-Datei auf App-Ebene** innerhalb der „dependencies“ hinzugefügt werden:

```
1  implementation 'com.firebaseui:firebase-ui-storage:8.0.1'
2  implementation 'com.github.bumptech.glide:glide:4.13.2'
3  annotationProcessor 'com.github.bumptech.glide:compiler:4.13.2'
```

[49] https://developer.android.com/reference/android/widget/ArrayAdapter
[50] https://github.com/firebase/FirebaseUI-Android/blob/master/storage/README.md
[51] https://github.com/bumptech/glide

Nach erfolgreichem Synchronisieren musste zusätzlich noch die Klasse **MyAppGlideModule** (siehe nächster Abschnitt) dem Projekt hinzugefügt werden, um *GlideApp* nutzen zu können. Anschließend können nun die Bilder der vermissten Kinder heruntergeladen und zu den jeweiligen Fällen angezeigt werden. Für die vorliegende App muss das Bild im *Cloud Storage* unter dem gleichen eindeutigen Fallnamen abgespeichert sein, wie der dazugehörige Fall in der Datenbank. Außerdem muss das Dateisuffix des Bildes `.jpg` oder `.png` sein. Sind beide Punkte erfüllt, kann das Bild in das *ImageView* geladen werden. Ist der Dateiname falsch oder die Datei endet nicht auf `.jpg` oder `.png`, wird der Fall trotzdem erzeugt und das *ImageView* bleibt leer.

MyAppGlideModule

Die Klasse **MyAppGlideModule** ist eine von Sam Stern et al. vorprogrammierte Klasse. Sie wird benötigt, um *GlideApp* nutzen zu können, also um Bilder aus dem Firebase *Cloud Storage* direkt in ein definiertes *ImageView* der App zu laden. [79] [80]

ActivitySelectedCase

In der Datei **ActivitySelectedCase.java** ist der Quellcode für die *Activity* enthalten, welche den von den Nutzer*innen ausgewählten Fall lädt und anzeigt. Dabei wird zuerst in der *Callback*-Methode `onCreate()` das in der *layout-Ressource* definierte Aussehen **activity_selected_case.xml** sowie die darin enthaltenen *TextViews* und das *ImageView* eingebunden. Anschließend werden die übergebenen Fallinformationen aus der Klasse **FragmentCurrentCases** geladen, zwischengespeichert und in die entsprechenden *TextViews* gefüllt. Weiterhin kann über den Fallnamen und mithilfe der *Glide*-Umgebung das Bild des vermissten Kindes zum jeweiligen Fall in das entsprechende *ImageView* geladen werden.
Dem *ImageView* wird zusätzlich ein `OnClickListener()` hinzugefügt, welcher dafür sorgt, dass das Bild größer angezeigt wird, wenn die Nutzer*innen dieses berühren. Das geschieht, indem eine neue *Activity* der Klasse **ActivityShowPicture** aufgerufen und der Fallname erneut übergeben wird.

ActivityShowPicture

Die Klasse **ActivityShowPicture** zeigt das Bild des jeweils vermissten Kindes in groß an, wenn in der *Activity* des ausgewählten Falls auf das Bild des Kindes geklickt wird. Hierfür wird zu Beginn in der *Callback*-Methode `onCreate()` das in der Datei **activity_show_picture.xml** definierte Aussehen und das darin enthaltene *ImageView* eingebunden. Anschließend wird der Fallname geladen sowie zwischengespeichert und über diesen sowie die *Glide*-Umgebung das Bild des vermissten Kindes zum entsprechenden Fall in das *ImageView* geladen und in voller Bildschirmbreite bzw. -höhe dargestellt.

FragmentChildMissingWhatToDo

Mithilfe der Klasse **FragmentChildMissingWhatToDo** wird die *layout-Ressource* **fragment_child_missing_what_to_do.xml** als entsprechende *Destination* in der *Callback*-Methode `onCreateView()` eingebunden. Hierdurch werden in der App Informationen darüber sichtbar gemacht, wie Personen zu handeln haben, wenn diese ihr eigenes Kind vermissen.

FragmentAboutAmber

Die Datei **FragmentAboutAmber.java** enthält den Quellcode der Klasse, welche allgemeine Informationen über den AMBER-Alert in der Applikation dargestellt. Hierfür wird in der *Callback*-Methode `onCreateView()` die *layout-Ressource* **fragment_about_amber.xml** als eine weitere *Destination* eingebunden.

FragmentSocialMedia

Die Klasse **FragmentSocialMedia** zeigt die *Destination* mit den Social-Media-Verknüpfungen an. Hierfür wird in der *Callback*-Methode `onCreateView()` die *layout-Ressource* **fragment_social_media.xml** eingebunden. Im Anschluss werden die *ImageViews* mit den Social-Media-Icons geladen und jedem dieser Icons wird ein `OnClickListener()` hinzugefügt. Die *Listener* sorgen dafür, dass über die Berührung eines der Icons, die dazu entsprechende Social-Media-Verknüpfung aufgerufen wird.
Im Folgenden ist der Aufbau des *Listeners* auf das Instagram-Icon dargestellt. In Code-Zeile 5 wird deutlich, dass bei Berührung des Icons keine zum deutschen AMBER-Alert gehörende Verknüpfung aufgerufen wird, sondern lediglich die offizielle Instagram-Webseite[52]. Grund dafür ist, dass es sich bei der vorliegenden App ausschließlich um einen Prototypen handelt. Wird jedoch ein entsprechender Account für die Social-Media-Plattform erstellt, kann der Link in Code-Zeile 5 mit dem Link zum erstellten Account getauscht werden.

```
1 imageViewInstagram.setOnClickListener(v -> {
2   Intent intent = new Intent();
3   intent.setAction(Intent.ACTION_VIEW);
4   intent.addCategory(Intent.CATEGORY_BROWSABLE);
5   intent.setData(Uri.parse("https://instagram.com"));
6   startActivity(intent);
7 });
```

Der Aufbau des Facebook- und Twitter-Icon-*Listeners* ist analog zum Instagram-Icon-*Listener*. Hierbei werden im Rahmen der vorliegenden Arbeit ebenfalls nur die offiziellen Webseiten der berührten Icons aufgerufen.

[52] `https://instagram.com`

FragmentAboutApp

Mittels der Klasse **FragmentAboutApp** wird die *Destination* mit den allgemeinen Informationen über die App dargestellt. Hierfür wird die entsprechende *layout-Ressource* **fragment_about_app.xml** in der *Callback*-Methode `onCreateView()` eingebunden.

FragmentPrivacy

Um die *Destination* mit der Datenschutzerklärung anzuzeigen, wird die Klasse **FragmentPrivacy** benötigt. Innerhalb dieser wird in der *Callback*-Methode `onCreateView()` die *layout-Ressource* **fragment_privacy.xml** eingebunden und somit die Datenschutzerklärung angezeigt.

FragmentLegalNotice

Innerhalb der *Callback*-Methode `onCreateView()` der Klasse **FragmentLegalNotice** wird die *layout-Ressource* **fragment_legal_notice.xml** eingebunden. Somit kann in der App die *Destination* mit dem Impressum sichtbar gemacht werden.

4.5 Testen der App

Nach der Implementierung der App muss diese auch getestet werden, um einen fehlerfreien Betrieb zu gewährleisten. Hierfür wurde nach der Implementierung ein Betatest durchgeführt. Im Rahmen dieses Tests wurde der Prototyp der App auf vier Android-Smartphones getestet, welche mit entsprechenden Eigenschaften in Tabelle 4.1 aufgelistet sind.

Tabelle 4.1: Zum Testen der App verwendete Geräte mit entsprechenden Geräteeigenschaften

Geräte-bezeichnung	**physisch / virtuell**	**Auflösung (in Pixel)**	**Zoll**	**Android-Version**
Samsung Galaxy A52	physisch	1.080 x 2.400	6,5	12.0
Samsung Galaxy A5 (2017)	physisch	1.080 x 1.920	5,2	8.0
Samsung Galaxy A3 (2016)	physisch	720 × 1.280	4,7	7.0
Nexus 5X (Google + LG)	virtuell	1.080 x 1.920	5,2	8.0

Um die App auf physischen Smartphones testen zu können, müssen die entsprechenden USB-Treiber auf dem Entwicklungs-PC installiert sein. In diesem Fall wurde für alle drei physische Smartphones der Samsung Android USB-Treiber für Windows[53] benötigt. Zusätzlich mussten auf den Smartphones der Entwicklermodus und USB-Debugging aktiviert werden. Anschließend konnte die App über eine USB-Verbindung zum Entwicklungs-PC auf den drei Smartphones ausgeführt werden.

Beobachtungen

Mit den zwei physischen Samsung Smartphones Galaxy A52 und Galaxy A3 konnte die App vollständig genutzt werden. Jedoch stürzt die App beim Drehen der Smartphones vom Portrait- in den Landscape-Modus und umgekehrt ab. Wird die App daraufhin wieder geöffnet, wird diese jedoch trotzdem im jeweiligen Modus dargestellt und die Funktionen werden dadurch nicht beeinträchtigt.
Beim Samsung Galaxy A3 konnte zusätzlich noch ein Layoutfehler in der *Activity* **activity_selected_case.xml** festgestellt werden. Hierbei haben sich die *TextViews* mit dem Alter und der Größe des vermissten Kindes überschnitten, was am zu kleinen Abstand zwischen den *TextViews* für die jeweilige Bildschirmbreite lag. Die Funktionen der App wurden hiervon nicht beeinträchtigt.

[53] `https://developer.samsung.com/android-usb-driver`

Auf dem Samsung Galaxy A5 stürzte die App ebenfalls beim Drehen des Smartphones ab, jedoch lies sich auch hier die App wieder öffnen und wurde anschließend im richtigen Format dargestellt. Weiterhin war der gleiche Layoutfehler wie beim Galaxy A3 festzustellen, was ebenso an einem zu kleinen Abstand zwischen den *TextViews* für die Bildbreite des Smartphones lag. Während hierdurch keine Funktionen eingeschränkt wurden, lies sich jedoch das Social-Media-Fragment nicht öffnen. Wurde versucht, dieses Fragment über das Menü zu öffnen, stürzte die App ab. Dies lag an einer zu großen Bildauflösung des Facebook-Icons und konnte mithilfe eines Icons mit geringerer Auflösung behoben werden. Alle weiteren Fragmente, *Activities* und Funktionen wurden vom Smartphone unterstützt.
Beim Nexus 5X traten die gleichen Fehler wie beim Samsung Galaxy A5 auf und wurden durch die gleichen Änderungen korrigiert.

Ergebnis

Durch das Feedback der Testpersonen konnten Fehler schneller und besser identifiziert werden. Die in der Betatestphase festgestellten Fehler der App sollten vor der Veröffentlichung behoben werden, wobei bereits Lösungsansätze beschrieben wurden.
Generell lässt sich feststellen, dass die App auf allen Testgeräten ausführbar war und die Funktionen bis auf eine Ausnahme nicht beeinträchtigt waren. Durch die Behebung der Layout- und Auflösungsfehler konnte die App schlussendlich auf allen Smartphones ohne Einschränkung der Funktionen genutzt werden. Das Abstürzen der App durch Bildschirmrotation wurde im Rahmen der vorliegenden Arbeit noch nicht korrigiert. Jedoch kann dieser Fehler durch Beachtung des *Activity Lifecycles*[54] gelöst werden.
Ebenfalls konnte durch das Testen auf fehlende Funktionen, welche sich Nutzer*innen wünschen, hingewiesen werden. Dazu gehört beispielsweise die Verknüpfung des zuletzt gesehenen Ortes des Kindes mit einer online Karte (z. B. Google Maps).
Wird die App über die *Google Play Console*[55] veröffentlicht, können zusätzlich die für die Anwender*innen nicht sichtbaren Fehler an die Konsole zur Analyse gesendet werden.

[54] `https://developer.android.com/guide/components/activities/activity-lifecycle`
[55] `https://developer.android.com/distribute/console`

5 Anleitung zum Anlegen eines Falls

In diesem Kapitel wird dargestellt, wie ein Fall eines vermissten Kindes angelegt und eine Benachrichtigung darüber an die Nutzer*innen der App versendet werden kann. Außerdem wird zusätzlich auf das Bearbeiten und Löschen eines Falls eingegangen.

1. Laden der Falldaten in die *Realtime Database*

Zu Beginn müssen die Falldaten in die *Realtime Database* geladen werden. Hierfür gibt es die im Folgenden erklärten zwei Möglichkeiten.

Variante 1:

Bei der ersten Variante können die Daten direkt online in die Firebase-Konsole geschrieben werden. Eine Übersicht der Konsole ist dabei in Abbildung 5.1 dargestellt.
Zuerst muss am linken Bildschirmrand **Realtime Database** gewählt werden. Anschließend können im mittleren Bereich über das **Plus-Symbol** neben der Datenbank ein oder mehrere Schlüssel-Werte-Paare hinzugefügt werden. Während das Wert-Feld leer bleiben kann, muss das Schlüssel-Feld immer gefüllt sein.

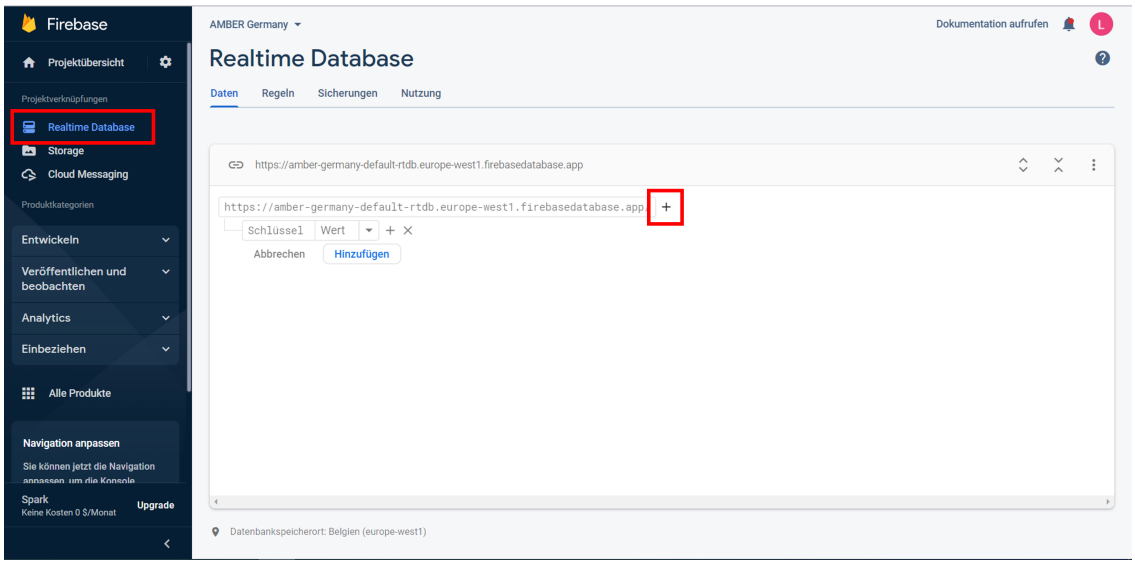


Abbildung 5.1: Screenshot der *Realtime Database* in der Firebase-Konsole mit rot markiertem Plus-Symbol, über welches der Datenbank Daten hinzugefügt werden können (eigener Screenshot)

Über diese Schlüssel-Werte-Paare der ersten Ebene können die einzelnen Fälle über eindeutige Fallnamen definiert werden. Dabei wird der Fallname in das **Schlüssel**-Feld geschrieben, während das Wert-Feld leer bleibt (siehe Abbildung 5.2).

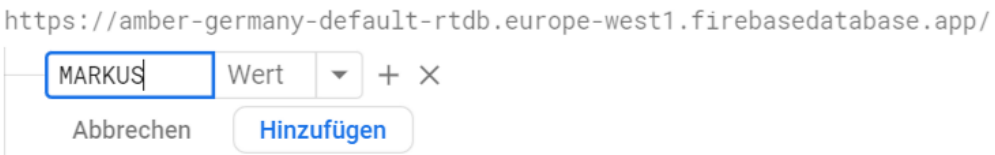


Abbildung 5.2: Screenshot vom Anlegen eines Falls in der Firebase-Konsole, indem der Fallname („MARKUS“) in ein Schlüssel-Feld der ersten Ebene geschrieben wurde (eigener Screenshot)

Anschließend können über das **Plus-Symbol** neben dem Fallnamen Schlüssel-Werte-Paare der zweiten Ebene hinzugefügt werden (siehe Abbildung 5.3). In diesen werden die entsprechend benötigten Fallinformationen zum jeweiligen Fall definiert.

Abbildung 5.3: Screenshot vom rot markierten Plus-Symbol neben dem Fallnamen, über welches Schlüssel-Werte-Paare der zweiten Ebene hinzugefügt werden können (eigener Screenshot)

Die Fallinformationen werden mithilfe von eindeutig festgelegten Nummer abgespeichert. Welche Nummer welcher Fallinformation entspricht, wird in der folgenden Tabelle 5.1 beschrieben.

Tabelle 5.1: Benötigte Fallinformationen für die in Firebase erstellten Fälle mit einer Nummer, über welche auf die entsprechende Information zugegriffen werden kann

Nummer	Fallinformation
01	Name
02	Ort, wo zuletzt gesehen
03	Alter
04	Größe
05	vermisst seit
06	Augenfarbe
07	Haarfarbe
08	Haarlänge
09	Geschlecht
10	zuletzt getragene Kleidung
11	auffällige Wiedererkennungsmerkmale
12	Informationen über den*die Täter*innen
13	weitere Informationen
14	Telefonnummer der zuständigen Polizei

Die Nummern werden dabei in die **Schlüssel**-Felder geschrieben und in die **Wert**-Felder die dazu gehörigen Fallinformationen. In Abbildung 5.4 wird ein Beispielfall mit allen 14 ausgefüllten Fallinformationen dargestellt.

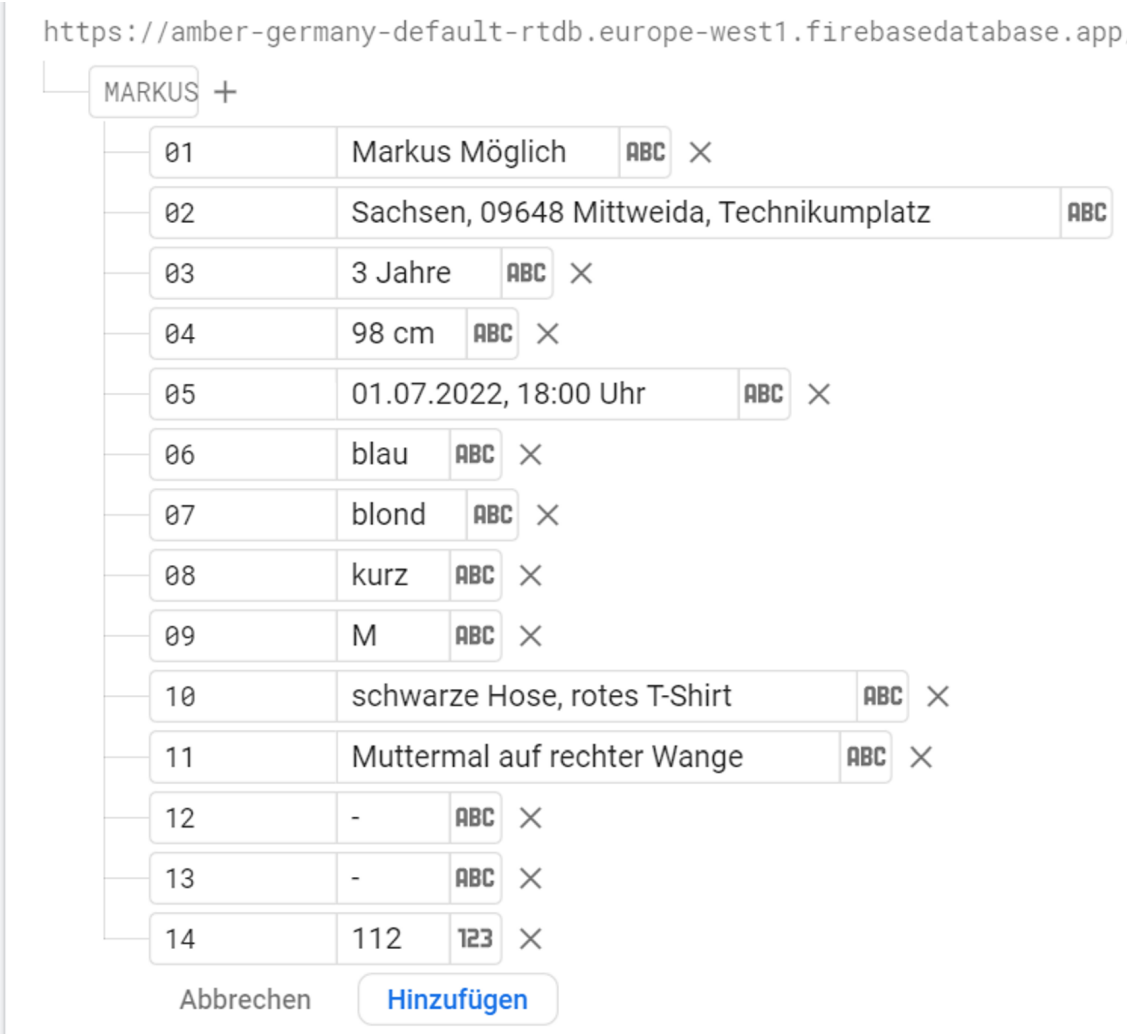


Abbildung 5.4: Screenshot eines in der Firebase-Konsole angelegten Beispielfalls mit allen benötigten Fallinformationen (eigener Screenshot)

Sind alle Felder ausgefüllt, kann der Fall über den Button **Hinzufügen** erstellt und somit in der App angezeigt werden.
Zum Zwischenspeichern kann ebenso während des Ausfüllens der Hinzufügen-Button gedrückt werden, auch wenn noch nicht alle Felder vollständig sind. Trifft das zu, wird der Fall noch nicht in der App angezeigt, sondern erst dann, wenn alle 14 untergeordneten Schlüssel-Werte-Paare des Falls gefüllt sind.
Liegen einige der entsprechend benötigten Fallinformationen nicht vor, kann in diese Felder ein Leerzeichen oder „-“ eingefügt werden (siehe Abbildung 5.4: Nummer 12 und 13).

Variante 2:

Die zweite Variante um Falldaten in die *Realtime Database* zu laden, besteht darin, die Daten lokal am Computer in eine JSON-Datei zu schreiben und diese anschließend in die Firebase-Konsole zu importieren. Im Folgenden ist der Aufbau einer solchen JSON-Datei mit den gleichen Beispiel-Falldaten aus Abbildung 5.4 zu sehen.

```
{
  "MARKUS": {
    "01": "Markus Möglich",
    "02": "Sachsen, 09648 Mittweida, Technikumplatz",
    "03": "3 Jahre",
    "04": "98 cm",
    "05": "01.07.2022, 18:00 Uhr",
    "06": "blau",
    "07": "blond",
    "08": "kurz",
    "09": "M",
    "10": "schwarze Hose, rotes T-Shirt",
    "11": "Muttermal auf rechter Wange",
    "12": "-",
    "13": "-",
    "14": "112"
  }
}
```

Wurde eine JSON-Datei mit Falldaten erstellt, kann diese in die Firebase-Konsole geladen werden. Hierfür muss über die drei senkrechten Punkte am rechten Bildschirmrand das **Datenbank-Menü** ausgewählt werden (siehe Abbildung 5.5). Im Anschluss kann über **Daten aus JSON-Datei importieren** die JSON-Datei in die *Realtime Database* importiert und die angelegten Fälle in der App angezeigt werden.

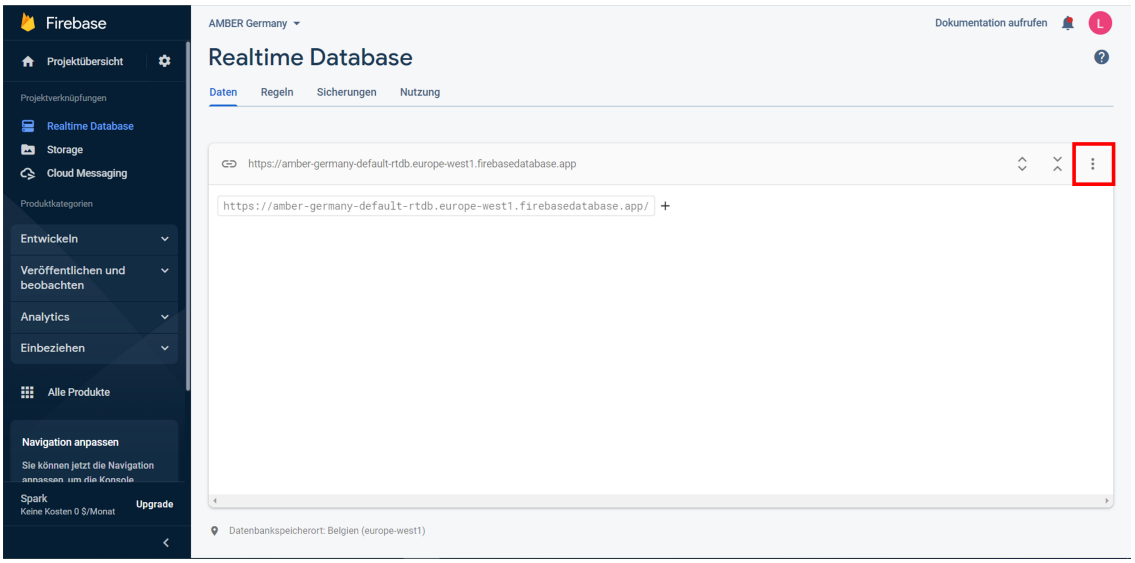


Abbildung 5.5: Screenshot der *Realtime Database* in der Firebase-Konsole mit rot markierten Menü-Punkten (eigener Screenshot)

2. Laden des Bildes in den *Cloud Storage*

Nach dem Laden der Falldaten in die Datenbank wird der Fall in der App sichtbar. Jedoch wird der Fall noch ohne ein Bild des vermissten Kindes dargestellt (siehe Abbildung 5.6).

Abbildung 5.6: Screenshot des Beispielfalls am App-Startbildschirm, nachdem die Beispiel-Falldaten in die *Realtime Database* geladen wurden (eigener Screenshot)

Um ein Bild zu einem Fall hinzuzufügen, muss die Firebase-Konsole benutzt werden. In Abbildung 5.7 ist eine Übersicht der Konsole dargestellt.
Zu Beginn muss am linken Bildschirmrand **Storage** gewählt werden. Im Anschluss kann am rechten Bildschirmrand über den Button **Datei hochladen** ein Bild in den Firebase *Cloud Storage* geladen werden.

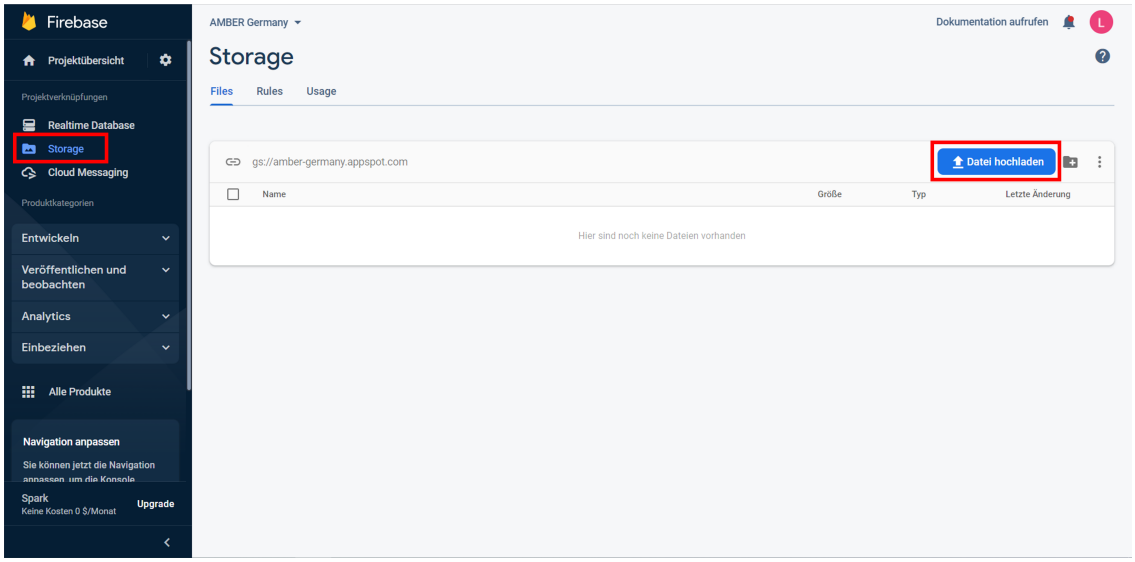


Abbildung 5.7: Screenshot des *Cloud Storage* in der Firebase-Konsole mit rot markiertem Button zum Hochladen von Bildern (eigener Screenshot)

Um eine Verknüpfung zwischen dem Bild und den zugehörigen Falldaten in der Datenbank herzustellen, muss das Bild unter dem gleichen eindeutigen Fallnamen abgespeichert sein, wie der dazugehörige Fall in der *Realtime Database*. Im Rahmen des vorliegenden Beispiels wäre das der Name „MARKUS“. Außerdem muss es sich bei dem Bild um eine JPG- oder PNG- Datei handeln. Sind beide Punkte erfüllt, wird das Bild zum Fall in der App sichtbar.

3. Benachrichtigungen über neu angelegte Fälle an die Nutzer*innen der App senden

Wurde ein neuer Fall zu einem vermissten Kind angelegt, sollten zuletzt noch die Nutzer*innen der App darüber informiert werden. Verwenden Nutzer*innen die App, während ein neuer Fall angelegt wird, wird ihnen dieser Fall direkt unter „Aktuelle Fälle“ angezeigt. Jedoch sollten auch die Nutzer*innen über einen neuen Fall informiert werden, welche die App nicht im gleichen Moment verwenden. Dies ist mithilfe des Firebase *Cloud Messaging* realisierbar, wobei es zwei Möglichkeiten gibt, eine Benachrichtigung über die Firebase-Konsole zu verfassen.

Variante 1:

Bei der ersten Variante handelt es sich um die Erstnutzung von *Cloud Messaging*, also wenn zuvor noch keine Benachrichtigung versendet wurde. Eine Übersicht der Firebase-Konsole ist in Abbildung 5.8 dargestellt.
Zu Beginn muss innerhalb der Konsole am linken Bildschirmrand **Cloud Messaging** ausgewählt werden. Anschließend kann über den Button **Send your first message** eine Benachrichtigung erstellt werden.

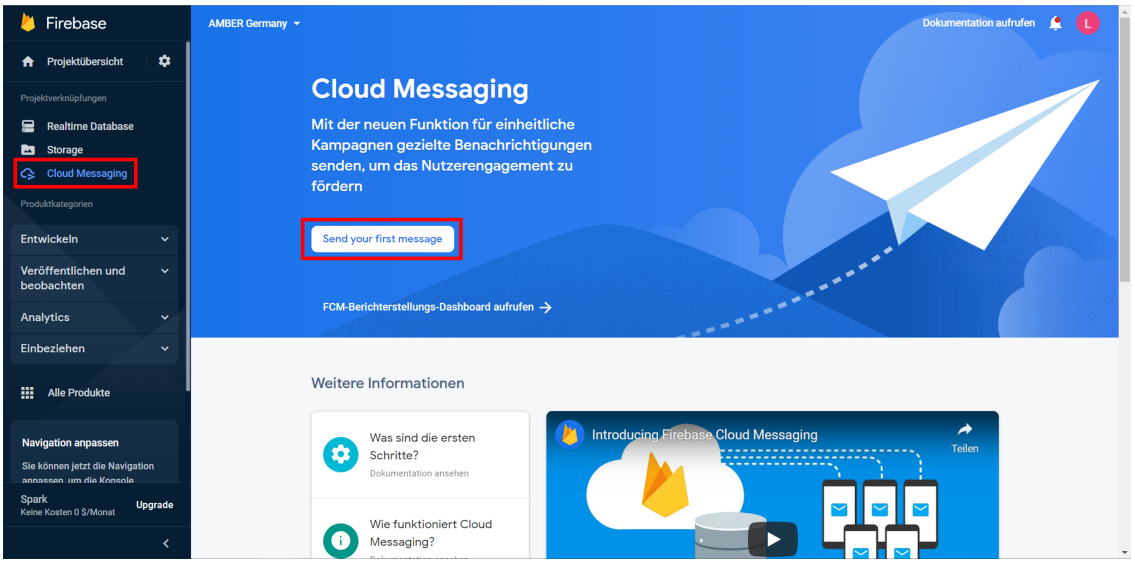
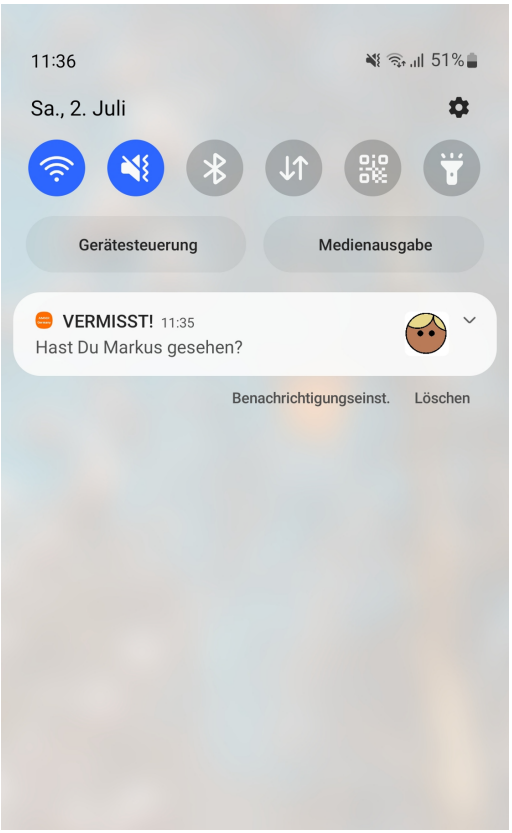


Abbildung 5.8: Screenshot des *Cloud Messaging* in der Firebase-Konsole mit rot markiertem Button zum Erstellen einer Benachrichtigung bei Erstnutzung (eigener Screenshot)

Beim Verfassen der Benachrichtigung können verschiedene Benachrichtigungseinstellungen getroffen werden. So können z. B. ein Titel, der eigentliche Benachrichtigungstext und ein Bild festgelegt werden. Im vorliegenden Beispiel wurde als Titel „VERMISST!“ und als Benachrichtigungstext „Hast Du Markus gesehen?“ gewählt. Als Benachrichtigungsbild wurde der Link des entsprechenden Bildes „MARKUS“ aus dem

Cloud Storage eingefügt.
Anschließend kann das **Ziel** der Benachrichtigung gewählt werden, wobei die erstellte *AMBER Germany* App ausgewählt wurde. So wird sichergestellt, dass alle Nutzer*innen der App die Benachrichtigung bekommen.
Unter **Zeitplan** kann der Zeitpunkt geplant werden, wann die Benachrichtigung an die Nutzer*innen geschickt wird. Wobei zur Alarmierung bei vermissten Kindern immer der Zeitpunkt „Jetzt“ gewählt werden sollte, um die Nutzer*innen so schnell wie möglich zu informieren.
Außerdem können optional ***Conversion*-Ereignisse** (*Google-Analytics*-Optionen) sowie **zusätzliche Optionen** eingestellt werden.
Zuletzt kann über den Button **Überprüfung** die Benachrichtigung veröffentlicht und somit an die Nutzer*innen der App gesendet werden.
In Abbildung 5.9 wird die gesendete Benachrichtigung angezeigt. Links ist die Standardansicht der Benachrichtigung und rechts die erweiterte Benachrichtigungsansicht dargestellt. Die erweiterte Ansicht ist zu sehen, wenn die Standardansicht auf dem Bildschirm nach unten gezogen wird. Der Unterschied der beiden Ansichten ist, dass bei der erweiterten Ansicht das Bild des vermissten Kindes größer dargestellt wird. Klickt ein*e Nutzer*in nun auf die Benachrichtigung, wird die App geöffnet und er*sie sieht den Startbildschirm der App mit der Übersicht aller aktuellen Fälle inkl. des neuen Falls.



(a) Standardbenachrichtigungsansicht

(b) erweiterte Benachrichtigungsansicht

Abbildung 5.9: Screenshot der Standardbenachrichtigungsansicht (a) und der erweiterten Benachrichtigungsansicht (b) des erstellten Beispielfalls (eigene Screenshots)

Variante 2:

Die zweite Variante zum Erstellen einer Benachrichtigung muss dann genutzt werden, wenn zuvor schon mindestens eine Benachrichtigung versendet wurde. Eine Übersicht dieser Konsolen-Ansicht ist in Abbildung 5.10 dargestellt.
Zu Beginn muss innerhalb der Firebase-Konsole am linken Bildschirmrand **Messaging** und danach am rechten Bildschirmrand der Button **Neue Kampagne** gewählt werden. Im Anschluss kann über **Benachrichtigung** eine neue Benachrichtigung erstellt werden.

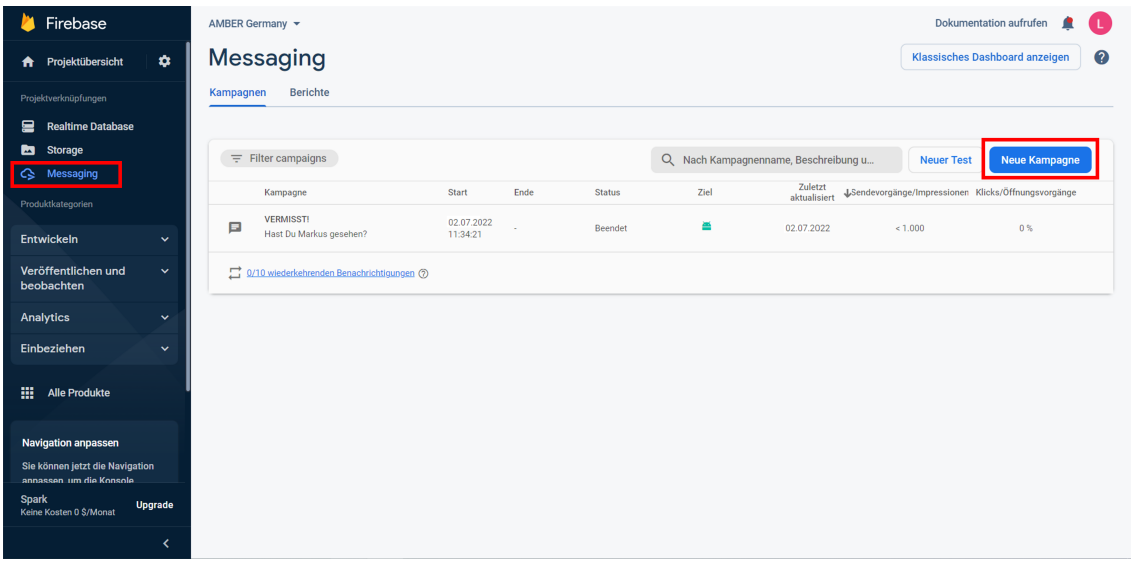


Abbildung 5.10: Screenshot des *Cloud Messaging* in der Firebase-Konsole mit rot markiertem Button zum Erstellen einer neuen Benachrichtigung (eigener Screenshot)

Die Benachrichtigung selbst wird analog zur ersten Variante verfasst und veröffentlicht. Die beiden Varianten unterscheiden sich im Wesentlichen nur dahingehen, ob bereits eine Benachrichtigung erstellt wurde und wie die Firebase-Konsole dementsprechend dargestellt wird.

5.1 Zusatz: Bearbeiten eines Falls

Wurde ein Fall angelegt, muss dieser auch bearbeitet werden können. Hierfür gibt es wieder zwei Möglichkeiten, welche im Folgenden erklärt werden.

Variante 1:

Bei Variante 1 können die Daten innerhalb der Firebase-Konsole direkt über die **Wert**-Felder der *Realtime Database* verändert und mit Enter bestätigt werden. Anschließend werden die abgeänderten Daten zum jeweiligen Fall direkt in der App angezeigt.

Variante 2:

Bei der zweiten Variante kann die aktuellste JSON-Datei aus der *Realtime Database* exportiert, entsprechend abgeändert und wieder importiert werden. Hierbei werden ebenfalls die abgeänderten Daten direkt in der App dargestellt.

5.2 Zusatz: Löschen eines Falls

Muss ein Fall wieder gelöscht werden, z. B. weil das vermisste Kind aufgefunden wurde, gibt es hierfür nur eine Möglichkeit. In der *Realtime Database* der Firebase-Konsole muss neben dem erstellten Fall das **Mülleimer**-Symbol gedrückt (siehe Abbildung 5.11) und die Löschung anschließend über den Button **Löschen** bestätigt werden.

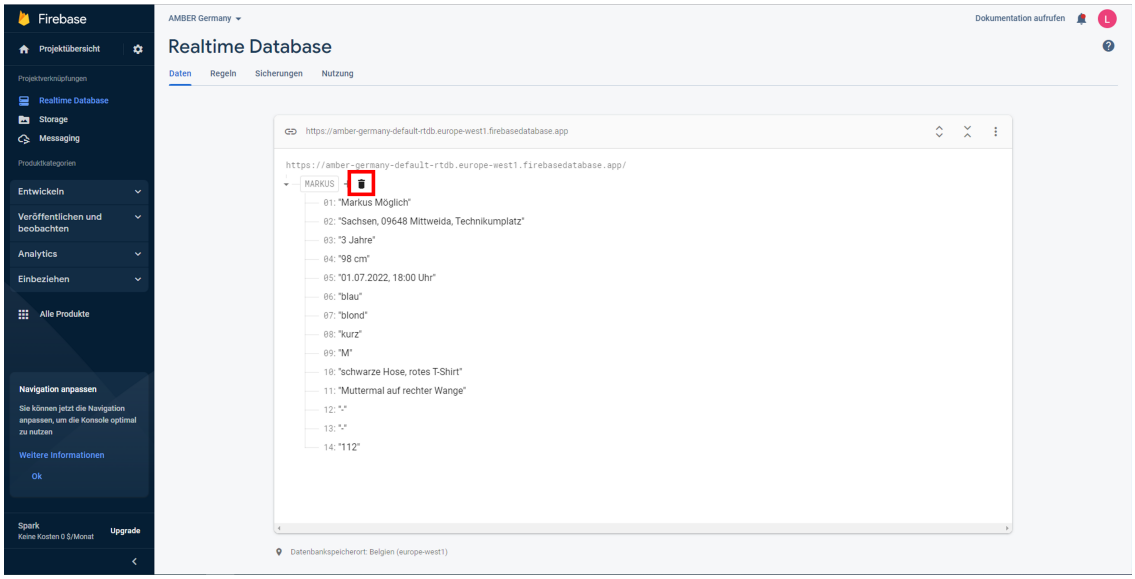


Abbildung 5.11: Screenshot der *Realtime Database* in der Firebase-Konsole mit rot markiertem Button zum Löschen eines Falls (eigener Screenshot)

6 Diskussion

Im letzten Kapitel werden zu Beginn die vorherigen Kapitel zusammengefasst. Anschließend wird die im Rahmen der vorliegenden Arbeit implementierte App zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland mit den bereits existierenden Apps verglichen. Weiterhin wird auf die Einbindung von Vermisstenfällen in bereits bekannte Warn-Apps und auf das sog. Cell Broadcast eingegangen. Zum Schluss wird ein Fazit gezogen sowie ein Ausblick gegeben, wie in Zukunft mit der entwickelten App weitergearbeitet werden kann.
Die im Rahmen der vorliegenden Arbeit entwickelte App wird im Folgenden „AMBER Germany“ genannt.

6.1 Zusammenfassung

6.1.1 Grundlagen

AMBER-Alert

In Kapitel 2 wird zu Beginn auf das AMBER-Alert-System eingegangen. Das System stammt aus den USA und ist eine Kooperation zwischen Strafverfolgungsbehörden, Rundfunkeinrichtungen sowie Verkehrsbehörden. Es wurde 1996 zur schnellen Suche und sicheren Bergung von vermissten bzw. entführten Kindern eingeführt. Wird ein AMBER-Alert über ein vermisstes Kind von einer Strafverfolgungsbehörde herausgegeben, wird diese Meldung anschließend über verschiedene Wege verbreitet, um einen Großteil der Bevölkerung zu erreichen. Dies geschieht z. B. über das Radio und Fernsehen, Autobahnschilder oder auch Smartphones. Dabei steht vor allem eine ortsgebundene Alarmierung im Fokus. [2] [6] [8]
Neben den USA wird ein derartiges System in mehr als 30 weiteren Ländern benutzt [2]. So gibt es beispielsweise in Europa die Organisation „*AMBER Alert Europe*“, welche die Suche nach vermissten Kindern in ganz Europa sowie die Länder bei der Entwicklung eines eigenen AMBER-Alert-Systems unterstützt [17]. Das in Deutschland bestehende AMBER-Alert-System wird von der NGO „Initiative Vermisste Kinder“ betrieben, wobei es beim deutschen System weniger Möglichkeiten zur Verbreitung von Vermisstenmeldungen gibt als beim staatlich betriebenen System aus den USA [7].

Android-App-Entwicklung

Anschließend an die Erklärung des AMBER-Alert-Systems folgt die Darstellung der Grundlagen zum Entwickeln einer Applikation für das mobile Betriebssystem Android. Dabei handelt es sich um eine native App, welche auf der quelloffenen Plattform Android beruht. Android bietet dementsprechend mehr als nur ein Betriebssystem, z. B. Treiber, Bibliotheken und System-Apps [28]. Neben der Android-Architektur, den App-Komponenten sowie wichtigen zusätzlichen Dateien und Ressourcen, welche zum Entwickeln einer Android-App benötigt werden, wird auf die Software Android Studio eingegangen. Android Studio ist die offizielle IDE zum Entwickeln von Android-Apps und bietet eine einheitliche Umgebung zum Entwickeln, Codevorlagen sowie Möglichkeiten zum Testen der implementierten App. [35]
Abschließend wird auf die App-Entwicklungsplattform Firebase, die Sprachen Java und XML sowie das Datenaustauschformat JSON näher eingegangen. Für die App „AMBER Germany“ wurde Firebase als Backend-Infrastruktur, JSON zum Anlegen und Bearbeiten von Fällen in Firebase sowie Java und XML zum Entwickeln der App genutzt.

6.1.2 Konzeptioneller Entwurf

Im 3. Kapitel wird ein mögliches Konzept einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland entworfen. Zu Beginn wird darauf eingegangen, warum eine Android-Applikation hierfür sinnvoll wäre. Dabei lies sich feststellen, dass über 85 % der Haushalte in Deutschland ein Smartphone besitzen [5], welches im Schnitt zwischen 3,5 und 4 Stunden genutzt wird [69]. Außerdem beträgt der Marktanteil des Betriebssystems Android in Deutschland rund 70 % [70], was deutlich zeigt, dass mittels einer Android-Applikation ein Großteil der Menschen in Deutschland erreicht werden kann.
Im Anschluss wurde der IST-Zustand einer solchen App in Deutschland genauer betrachtet. Hierfür wurden die vier Applikationen „Deutschland Findet Euch“, „Vermisst in Thüringen“, „hessenWARN“ und „ChildRescue“ analysiert. Bei dieser Analyse wurde deutlich, dass es in Deutschland nur zwei tatsächlich noch aktiv betriebene Apps zur Suche nach vermissten Personen gibt. Diese stellen jedoch ausschließlich Vermisstenfälle aus den Bundesländern Thüringen sowie Hessen bereit und enthalten ebenso Meldungen von vermissten Erwachsenen.
Aus diesem Grund wurde anschließend der gewünschte SOLL-Zustand einer deutschlandweiten App zur Alarmierung der Bevölkerung bei vermissten Kindern definiert. Dabei wird sich zum einen auf die funktionalen Anforderungen und zum anderen auf nichtfunktionale Anforderungen konzentriert. Zusätzlich wird eine erste Struktur der benötigten *Activities* und Fragmente für die App aufgezeigt.

6.1.3 Implementierung

In Kapitel 4 wird die Umsetzung des zuvor entworfenen Konzeptes aus Kapitel 3 dargestellt. Dabei wurde zu Beginn auf das Setup von Android Studio eingegangen, wobei zum Programmieren die Sprache Java gewählt wurde. Anschließend wurde Android Studio mit Firebase verknüpft, da diese Plattform als Backend-Infrastruktur dient. Hierfür wurde die *Realtime Database*, der *Cloud Storage* und das *Cloud Messaging* konfiguriert.
Nach dem Setup wurde das Aussehen der Applikation in den App-Ressourcen definiert. Hierbei wurden die in der App verwendeten Bilder, das Layout der *Activities* und Fragmente, das Aussehen des benötigten Menüs, das App-Icon, der Navigationsgraph sowie die verwendeten Farben, Strings und *Themes* festgelegt.
Im Anschluss wurden in den Java-Klassen die Funktionalitäten der App zum Verarbeiten der Interaktivitäten durch die Benutzer*innen implementiert.
Abschließend folgt ein Einblick in die Testphase. In dieser wurde im Rahmen eines Betatests die App auf vier Smartphones mit unterschiedlichen Geräteeigenschaften getestet, die Beobachtungen dokumentiert und das daraus folgende Ergebnis dargestellt.

6.1.4 Anleitung zum Anlegen eines Falls

Im 5. Kapitel wird die Anleitung zum Anlegen eines neuen Vermisstenfalls erläutert. Da Firebase als Backend-Infrastruktur konfiguriert wurde, können die Fälle über diese Plattform verwaltet werden. Zuerst müssen die Falldaten des vermissten Kindes in die *Realtime Database* und ein entsprechendes Bild von diesem Kind in den *Cloud Storage* geladen werden. Anschließend kann über das *Cloud Messaging* eine Benachrichtigung über diesen neuen Fall an die Nutzer*innen der App gesendet werden.
Zusätzlich wird in dem Kapitel erklärt, wie ein bereits existierender Fall bearbeitet und gelöscht werden kann.

6.2 Vergleich mit anderen Apps

Bereits in Abschnitt 3.2 wurde auf den IST-Zustand einer Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland eingegangen. Dabei wurden vier für diesen Fall entwickelte Apps genauer analysiert, welche im Folgenden der App „AMBER Germany“ gegenübergestellt werden.

Deutschland Findet Euch

Während die App „AMBER Germany“ nur für das Betriebssystem Android entworfen wurde, war die App „Deutschland Findet Euch“ nur für iOS-Geräte verfügbar. Jedoch wurde bereits in Abschnitt 3.1 erörtert, dass iOS einen geringeren Marktanteil als Android in Deutschland besitzt, weshalb mit dieser App nicht so viele Menschen erreicht werden könnten, wie mit der entwickelten „AMBER Germany“ App. Außerdem ist die App „Deutschland Findet Euch“ mittlerweile (Stand: September 2022) nicht mehr im App Store erhältlich.

Vermisst in Thüringen

Die App „Vermisst in Thüringen“ ist ebenso wie die App „AMBER Germany“ ausschließlich für Android verfügbar. Jedoch soll „AMBER Germany“ lediglich Fälle von vermissten Minderjährigen zur Verfügung stellen, während in der App „Vermisst in Thüringen“ ebenso Fälle von vermissten Erwachsenen dargestellt werden. Ein weiterer Unterschied ist, dass die Applikation „Vermisst in Thüringen“ nur Fälle von vermissten Personen aus einem Bundesland enthält. Einerseits ist die App somit zwar regionaler gebunden, andererseits wird bei allen anderen Vermisstenfällen in Deutschland die Bevölkerung nicht mittels dieser App alarmiert. Selbst wenn jedes Bundesland eine eigene App für die Vermisstenfälle im jeweiligen Bundesland besitzen würde, wäre es komplizierter, den Großteil der Bevölkerung über diesen Weg zu erreichen. Wohnen Personen beispielsweise in der Nähe von Bundesländergrenzen oder fahren in den Urlaub, müssten sich diese mehrere Apps installieren, um über alle Fälle in ihrer Nähe informiert zu werden.

hessenWARN

Die Applikation „hessenWARN“ ist für die Betriebssysteme Android und iOS verfügbar. Hierdurch können mehr Personen erreicht werden als mit der App „AMBER Germany“. Außerdem stellt die App „hessenWARN“ ebenso Fälle von vermissten Erwachsenen zur Verfügung, während in der App „AMBER Germany“ nur Fälle von vermissten Minderjährigen angezeigt werden sollen. Jedoch ist ein Nachteil der App „hessenWARN“, dass sie nur Fälle von vermissten Personen aus einem Bundesland enthält. So ist sie zwar wie

die App „Vermisst in Thüringen“ regionaler gebunden, jedoch wird bei allen andren Vermisstenfällen in Deutschland die Bevölkerung nicht mittels dieser App informiert.

ChildRescue

Die vierte Vergleichs-App „ChildRescue“ ist sowohl für Android als auch für iOS verfügbar, wodurch mehr Personen zu erreichen sind als mit der App „AMBER Germany“. Außerdem enthält die App „ChildRescue“ Vermisstenfälle von Kindern aus ganz Europa, wodurch Personen auch an den Ländergrenzen oder bei Grenzübergängen die Möglichkeit haben, auf Vermisstenfälle aus diesen Ländern aufmerksam zu werden. Jedoch wird an den App-Bewertungen im Google Play Store deutlich, dass die App teilweise nicht funktioniert. Im Gegensatz dazu wurde die App „AMBER Germany“ getestet (siehe Abschnitt 4.5), um einen fehlerfreien Betrieb gewährleisten zu können. Außerdem soll sie ausschließlich für Vermisstenfälle von Kindern innerhalb Deutschlands zur Verfügung stehen.
Weiterhin lässt sich an den App-Bewertungen der App „ChildRescue“ erkennen, dass viele Optionen nicht in allen europäischen Sprachen erhältlich sind. Da die App „AMBER Germany“ ausschließlich Fälle aus Deutschland anzeigen soll, ist die App zum einen in deutscher Sprache verfügbar. Zum anderen ist sie ebenso in Englisch verfügbar, wenn die Sprache des Android-Gerätes im System auf Englisch gestellt ist, um so noch mehr Menschen mit der App erreichen zu können.
Abschließend fällt anhand des Zuletzt-Aktualisiert-Datums im Google Play Store auf, dass die Applikation „ChildRescue“ seit Projektende im Jahr 2020 scheinbar nicht mehr aktualisiert wurde, während die App „AMBER Germany“ erst im Rahmen dieser Arbeit entwickelt wurde.

Zusammenfassung

Der Vergleich der vorangegangenen Apps zeigt deutlich, dass es in Deutschland bisher (Stand: September 2022) keine funktionsfähige deutschlandweite App zur Alarmierung der Bevölkerung bei vermissten Kindern gibt. Mit „AMBER Germany“ wurde eine Möglichkeit geschaffen, potenziell einen Großteil der Bevölkerung in Deutschland über vermisste Minderjährige zu informieren, sie aktiv an der Suche zu beteiligen und so die Kinder und Jugendlichen sicher zu bergen.

6.3 Nutzung bekannter Warn-Apps

In Kapitel 3 wurde bereits die Frage beantwortet, warum eine Android-Applikation zur Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland sinnvoll ist. Allerdings könnte bei einigen Nutzer*innen die folgende neue Frage aufkommen:

- Warum können Vermisstenfälle nicht in bereits bekannte Warn-Apps mit aufgenommen werden?

Die zwei bekanntesten deutschlandweiten Warn-Apps sind **NINA**[56] vom Bundesamt für Bevölkerungsschutz und Katastrophenhilfe und **KATWARN**[57] vom Fraunhofer Institut für Offene Kommunikationssysteme FOKUS. Über diese Apps soll die Bevölkerung über Gefahrensituationen innerhalb Deutschlands informiert werden, wobei diese Warnmeldungen ebenso ortsbezogen sein können. [81] [82]
Da diese Apps bereits etabliert sind und mit mehreren Millionen Downloads sowohl im Google Play Store als auch App Store erhältlich sind, würde es sich anbieten, hierüber Fälle von vermissten Kindern und Jugendlichen zu verbreiten.
In den Jahren 2018 / 2019 wurde dieser Versuch in Hessen umgesetzt. Hier wurde über die App KATWARN nach vermissten Personen gesucht, sowohl nach Minderjährigen als auch nach Erwachsenen. Jedoch gab es hierfür zahlreiche Kritik von Nutzer*innen der App, welche drohten, sich aufgrund dessen von der App wieder abzumelden. Diese Nutzer*innen wollten über die App keine Benachrichtigungen von vermissten Personen erhalten, sondern nur vor großen Gefahren, wie beispielsweise schweren Unwettern oder anderen lebensbedrohlichen Situationen gewarnt werden. [83]
Aus diesem Grund wurde **hessenWARN**[58] aus KATWARN weiterentwickelt (siehe Abschnitt 3.2). Über diese App erhalten die Nutzer*innen weiterhin alle Meldungen, welche sie ebenso über KATWARN erhalten haben. Ebenfalls enthält diese App Zusatzinformationen, z. B. Vermisstenmeldungen. So können sich Personen aussuchen, welche App sie installieren bzw. welche Warnmeldungen sie erhalten möchten. [74]
Jedoch ist diese weiterentwickelte App mit Vermisstenmeldungen nur für das Bundesland Hessen erhältlich, wodurch sich die bereits erwähnte Problematik ergibt, dass die App zwar regional gebunden ist, sich jedoch nur auf Vermisstenfälle innerhalb Hessens konzentriert wird und somit Fälle in anderen Bundesländern nicht beachtet werden.
Das Vorangegangene zeigt, dass eine Alarmierung der Bevölkerung bei vermissten Kindern in Deutschland über bereits bekannte Warn-Apps möglich ist, jedoch von den Nutzer*innen solcher Apps kritisch gesehen wird. Aus diesem Grund ist eine eigene App, welche nur für Meldungen von Vermisstenfällen zuständig ist, sinnvoll.

[56] https://www.bbk.bund.de/DE/Warnung-Vorsorge/Warn-App-NINA/NINA-Download/nina-download_node.html
[57] https://www.katwarn.de/anmeldung-app.php
[58] https://innen.hessen.de/Sicherheit/hessenWARN

6.4 Cell Broadcast

Eine weitere sehr effektive Möglichkeit, die Bevölkerung über vermisste Kinder zu informieren, wäre mithilfe des sog. *Cell Broadcasts*. Mittels Cell Broadcast können Meldungen an alle in einer bestimmten Funkzelle befindlichen Mobilfunkendgeräte gesendet werden. Somit können genau die Personen erreicht werden, welche von der Meldung unmittelbar betroffen sind. [84]
Dieses System kann beispielsweise zur Gefahrenwarnung genutzt werden und einen regional begrenzten Bereich der Bevölkerung alarmieren, z. B. vor einem drohenden Unwetter oder einer Hochwasserkatastrophe. Jedoch könnte dieses System neben der Warnung von Gefahrensituationen ebenso zur Herausgabe von Vermisstenmeldungen verwendet werden. Somit könnten immer die Personen erreicht werden, welche sich unmittelbar in der Nähe des Verschwindeortes des Kindes aufhalten.
Ein weiterer Vorteil des Cell Broadcasts besteht darin, dass der benötigte Datenverkehr nicht durch ein erhöhtes Aufkommen an Mobilfunkgesprächen beeinflusst wird. Erschwerend kommt allerdings hinzu, dass hiermit nur Textnachrichten versendet werden können und somit kein Bild des vermissten Kindes übertragen werden kann. Außerdem befindet sich dieses System in Deutschland aktuell in der Testphase und die Einführung ist noch nicht abgeschlossen (Stand: September 2022). [84]
Cell Broadcast soll voraussichtlich 2023 einsatzbereit sein. Jedoch steht noch nicht fest, für welche Art von Meldungen das System genutzt wird. Desgleichen ist es möglich, dass auch hier die Bevölkerung den Meldungen kritisch gegenübersteht, wenn sie statt ausschließlich über Warnungen von Gefahrensituationen ebenso über vermisste Kinder alarmiert wird.

6.5 Fazit und Ausblick

Wird ein Kind als vermisst gemeldet, spielt vor allem die Zeit, um dieses Kind wiederzufinden, eine entscheidende Rolle (siehe Abschnitt 2.1). Aus diesem Grund sollte schnellstmöglich ein Großteil der Bevölkerung alarmiert werden, sobald ein Kind als vermisst gemeldet wird, sodass diese Personen bei der Suche mithelfen können. Jedoch wird beim Betrachten der vorliegenden Arbeit deutlich, dass es in Deutschland bisher keine effektive Möglichkeit gibt, einen Großteil der Bevölkerung bei vermissten Kindern zu alarmieren. Obwohl eine Alarmierung über Smartphones eine Erfolg versprechende Idee darstellt, gibt es hierfür bisher keine geeignete Umsetzung.
Bereits entwickelte Apps sind auf einzelne Bundesländer begrenzt, besitzen Funktionsfehler oder sind nicht mehr verfügbar. Ebenso wird die Einbindung von Vermisstenfällen in bereits bekannte Warn-Apps als kritisch betrachtet. Weiterhin wäre Cell Broadcast eine ideale Möglichkeit, um die Bevölkerung über ihre Smartphones zu erreichen. Allerdings ist dieses System in Deutschland noch nicht einsatzbereit und es steht nicht fest, ob es überhaupt für Vermisstenmeldungen genutzt werden soll. Wird das System jedoch zukünftig auch für Vermisstenmeldungen verwendet, ist es möglich, dass die Nachrichtenempfänger*innen diesen Alarmierungen ebenso kritisch gegenüberstehen wie bei der Einbindung von Vermisstenfällen in bereits bekannte Warn-Apps.
Die im Rahmen der Arbeit entwickelte App „AMBER Germany“ stellt einen Prototypen dar, mit welchem es möglich ist, einen Großteil der Bevölkerung in Deutschland bei vermissten Kindern zu alarmieren. Dabei soll die Bevölkerung ausschließlich über vermisste Kinder und Jugendliche informiert werden, um Personen somit an der Suche nach den Vermissten zu beteiligen. Die App wurde zwar nur für Android-Geräte entwickelt, jedoch ist dieses Betriebssystem das mit Abstand am weitesten verbreitete in Deutschland (siehe Abschnitt 3.1). Außerdem wurde die App dafür entwickelt, die Bevölkerung über Vermisstenfälle von Minderjährigen aus ganz Deutschland zu alarmieren und sich nicht auf nur eine bestimmte Region (z. B. ein Bundesland) zu beschränken. Hierdurch soll gewährleistet werden, dass jedes vermisste Kind in Deutschland die Möglichkeit hat, dass so viele Personen wie möglich nach diesem Kind Ausschau halten.

6.5.1 App-Veröffentlichung

Bei der App „AMBER Germany“ handelt es sich ausschließlich um eine prototypische Implementierung. Deshalb wurde die App im Rahmen der vorliegenden Arbeit nicht zum Download bereitgestellt. Dies bedeutet jedoch nicht, dass die App in Zukunft nicht noch veröffentlicht werden kann. Die Erkenntnisse, dass die Zeit beim Wiederauffinden von vermissten Kindern und daraus resultierende Alarmierungen über Smartphones entscheidende Rollen spielen, sind unumstritten. Aus diesem Grund sollte zur schnellen Suche und sicheren Bergung von vermissten Kindern und Jugendlichen die App der Bevölkerung zum Download zur Verfügung gestellt werden. Dabei sollte die Bezugsquelle der App vertrauenswürdig sein, d. h. dass die App nur über offizielle App-Stores ange-

boten werden sollte. Mithilfe der *Google Play Console* ist es möglich, die App schnell und einfach im Google Play Store zu veröffentlichen [85].
Da die App bisher allerdings nur im kleinen Rahmen getestet wurde, ist es zudem sinnvoll, diese vor Veröffentlichung im größeren Umfang zu testen.

6.5.2 App-Verantwortlichkeit

Bevor die App veröffentlicht wird, sollten außerdem die Verantwortlichkeiten der Applikation geklärt werden. Dabei gibt es zwei Möglichkeiten, wer für die Erstellung und Bearbeitung von denen in der App angezeigten Vermisstenfällen zuständig ist: die Polizei oder eine NGO.
Das aktuell in Deutschland bestehende AMBER-Alert System wird bereits von der NGO „Initiative Vermisste Kinder“ betrieben (siehe Abschnitt 2.1.3). Aus diesem Grund wäre es sinnvoll, die App in die Zuständigkeit dieser NGO zu legen, um die bisherigen Möglichkeiten des Alarmierungssystems zu erweitern. So könnten sich die Erziehungsberechtigten des vermissten Kindes an die Initiative wenden und mit deren Hilfe eine Vermisstenmeldung über die App verbreiten lassen.
Jedoch bietet die Applikation im Verantwortungsbereich einer NGO einige Nachteile, welche durch die Zuständigkeit der Polizei nicht bestehen würden. Die Polizei kann beispielsweise nachprüfen, ob die persönlichen Daten des Kindes korrekt sind, das Kind tatsächlich vermisst wird und der Fall generell veröffentlicht werden sollte (siehe Anhang C).
Ist die Polizei für die App zuständig, stellt sich außerdem die Frage, ob es eine zentrale Stelle zum Anlegen, Bearbeiten und Löschen der Vermisstenfälle in der App geben sollte oder ob jede Polizeidienststelle eigene Zugangsdaten zur Firebase-Konsole erhält. Besitzt jede Stelle ihre eigenen Zugangsdaten, kann direkt nach der Vermisstenmeldung ohne großen Zeitverzug ein Fall von der zuständigen Polizei angelegt werden. Allerdings können hierdurch Daten von anderen Fällen von jedem mit den Zugangsdaten verändert werden (absichtlich und unabsichtlich). Der Vorteil einer zentralen Stelle ist, dass nur eine begrenzte Anzahl an Personen die Zugangsdaten zur Firebase-Konsole hat und somit die Gefahr einer absichtlichen und unabsichtlichen Falldatenänderung geringer wäre. Außerdem würde es somit einen definierten Personenkreis geben, welcher sich in die Firebase-Konsole und die damit verbundene Fall-Verwaltung einarbeiten kann, womit Flüchtigkeitsfehler durch Unerfahrene vermieden werden können. Jedoch muss diese Zentralstelle auch nachts erreichbar sein, da Kinder und Jugendliche zu jeder Tageszeit als vermisst gemeldet werden können.

6.5.3 Weiterentwicklung der App

Nach der Klärung der App-Verantwortlichkeiten und der Veröffentlichung der App sollte diese stets weiterentwickelt werden. Dabei sollten zum einen Rückmeldungen von den Nutzer*innen beachtet werden, welche Informationen und Funktionen sich diese in der App wünschen. Zum anderen ist es sinnvoll, sich ebenso mit Experten und Expertinnen im Bereich Kindesentführung in Verbindung zu setzten, um zu ermitteln, welche weiteren Informationen zum vermissten Kind zusätzlich noch in die App eingebaut werden sollten.
Ebenfalls ist es wichtig, neu erscheinende Android-Versionen zu analysieren und den Programmcode entsprechend anzupassen.
Neue App-Versionen wiederum müssen vor der Veröffentlichung umfangreich getestet werden, damit weiterhin ein fehlerfreier Betrieb gewährleistet werden kann.
Zusätzlich zur Android-App kann zukünftig ebenso eine App für das Betriebssystem iOS entwickelt werden, um so noch mehr Menschen zu erreichen.

6.5.4 AMBER-Alert - mehr als nur eine App

Das ursprüngliche AMBER-Alert System aus den USA zeigt, dass eine Vermisstenmeldung mehr als nur eine Alarmierung über Smartphones bedarf. Zwar kann somit ein potenzieller Großteil der Bevölkerung erreicht werden, jedoch gibt es noch weitere Möglichkeiten, die Bevölkerung schnell und effizient auf vermisste Minderjährige aufmerksam zu machen. Beispielsweise sollten bei einer neuen Vermisstenmeldung wie beim amerikanischen EAS Radio- und Fernsehprogramme direkt unterbrochen und auf die Fälle mit umfassenden Informationen zum Kind und den möglichen Täter*innen aufmerksam gemacht werden. Zusätzlich sollten Informationen zum möglichen Täter*innen-Fahrzeug auf digitalen Autobahnschildern angezeigt werden. So können z. B. auch Personen beim Autofahren alarmiert werden, während sie keine Möglichkeit haben, ihr Smartphone zu benutzen.
Ebenso sollten Social-Media-Profile für die gängigsten Social-Media-Plattformen eingerichtet werden. So können auch Personen über Vermisstenmeldungen informiert werden, welche die App nicht installiert haben. Die Möglichkeit, die App mit Social-Media-Profilen zu verknüpfen, wurde bereits bei der App-Entwicklung geschaffen. So können in den Programmcode an entsprechender Stelle die jeweiligen Links zu den Accounts eingefügt und diese Profile über die entsprechende *Destination* „Social Media“ aufgerufen werden.
Ein bereits bekannter Kooperationspartner der „Initiative Vermisste Kinder“ zur Verbreitung von Vermisstenmeldungen ist Ströer, welcher digitale Werbetafeln zur Verfügung stellt (siehe Abschnitt 2.1.3).
Weiterhin sollten Hotels, Post- und Paketdienste sowie Stadt- und Bauarbeiter in die Suche über spezielle Schnittstellen direkt mit eingebunden werden. Werden Kinder oder Jugendliche als vermisst gemeldet, müssen die Mitarbeiter der zuvor genannten Berei-

che direkte Meldungen mit Informationen zum Fall auf ihre Systeme erhalten, um bei der Suche effektiv mithelfen zu können. So können z. B. Hotelmitarbeiter*innen darauf achten, ob der*die Täter*in mit dem Kind in ein Hotel eincheckt oder die Paketdienstmitarbeiter*innen, ob sie beim Austragen der Pakete das Kind oder den*die Täter*in wiedererkennen.

6.5.5 Cold Cases

Abschließend stellt sich die Frage, ob in die App ebenso ungeklärte Fälle (sog. *Cold Cases*) mit aufgenommen werden sollten. Da die Fälle nicht akut sind, müssen die Nutzer*innen der App auch nicht direkt darüber benachrichtigt werden. Dies könnte von ihnen als kritisch betrachtet werden, da kein schneller Handlungsbedarf besteht. Außerdem könnten Nutzer*innen bei übermäßig vielen AMBER-Alerts desensibilisiert werden. Jedoch könnte ein zusätzlicher Bereich in die App implementiert werden, in welchem ausschließlich Cold Cases angezeigt werden, sodass diese Fälle nicht in Vergessenheit geraten.

Anhang A: Android Studio Dateien

Im beiliegenden ZIP-Archiv befindet sich das entwickelte Android-Studio-Projekt. Teil hiervon sind z. B. die entwickelten Java-Klassen, die App-Ressourcen sowie die zum Erstellen der App benötigten Gradle-Build-Dateien.
Die entsprechenden Dateien befinden sich unter folgenden Pfad:

`MA_Bunzel_Android_Studio > AMBER`

Anhang B: Übersicht der in der App verwendeten Texte

B.1 fragment_child_missing_what_to_do.xml

Handlungsmaßnahmen

Ihr Kind wird vermisst und Sie wissen nicht, was Sie tun sollen? Als Hilfestellung sind im Folgenden Handlungsmaßnahmen für diesen Fall aufgelistet.

1. Telefonische Nachfrage

Zuerst einmal Ruhe bewahren und telefonisch nachfragen bei Freunden, Bekannten, Nachbarn und Familienmitgliedern. Sollten Sie hierbei keinen Erfolg haben, erweitern Sie den Kreis der zu Befragenden.

2. Suche vor Ort

Durchsuchen sie Ihr gesamtes Grundstück / Ihre Wohnung und die nähere Umgebung. Auch den Keller, Dachboden, Gartenhäuschen, Heimweg von der Schule sowie von Freunden und Verwandten, Lieblingsorte und typische Treffpunkte von Gleichaltrigen.
Sollten Sie zur Suche Ihr Haus / Ihre Wohnung verlassen, sorgen Sie dafür, dass immer jemand zu Hause ist, falls Ihr Kind wieder kommt oder das Telefon klingelt.

3. Polizei einschalten

Sollte die Telefonsuche und Suche vor Ort keinen Erfolg haben, sollten Sie umgehend die Polizei einschalten. Dabei sollten Sie (wenn möglich) wichtige Informationen über das Kind bereithalten:

- aktuelles Foto
- Geburtsdatum / Alter
- äußere Merkmale (z. B. Haar- und Augenfarbe, Größe)
- zuletzt getragene Kleidung
- auffällige Wiedererkennungsmerkmale (z. B. auffälliges Muttermal, Brille, Zahnspange)

Wichtige Telefonnummern

Euronotruf: **112**
Polizei: **110**
Hotline für vermisste Kinder: **116000**
(Die Hotline ersetzt nicht den polizeilichen Notruf unter der 110 bzw. 112)

Weiterführende Informationen:

https://www.initiative-vermisste-kinder.de/ich-benötige-hilfe/

B.2 fragment_privacy.xml

Datenschutzerklärung

1. Datenschutz auf einen Blick

Allgemeine Hinweise

Die folgenden Hinweise geben einen einfachen Überblick darüber, was mit Ihren personenbezogenen Daten passiert, wenn Sie diese mobile Anwendung (folgend App genannt) verwenden. Personenbezogene Daten sind alle Daten, mit denen Sie persönlich identifiziert werden können. Ausführliche Informationen zum Thema Datenschutz entnehmen Sie unserer unter diesem Text aufgeführten Datenschutzerklärung.

Datenerfassung innerhalb dieser App

Wer ist verantwortlich für die Datenerfassung innerhalb dieser App?
Die Datenverarbeitung innerhalb dieser App erfolgt durch den Herausgeber dieser App. Dessen Kontaktdaten können Sie dem Abschnitt „Hinweis zur verantwortlichen Stelle“ in dieser Datenschutzerklärung entnehmen.

Wie erfassen wir Ihre Daten?
Ihre Daten werden zum einen dadurch erhoben, dass Sie uns diese mitteilen. Hierbei kann es sich z. B. um Daten handeln, die Sie direkt in die App eingeben.

Andere Daten werden automatisch beim Verwenden der App durch unsere IT-Systeme erfasst. Das sind vor allem technische Daten (z. B. Mobiles Endgerät, Betriebssystem oder Uhrzeit Nutzung). Die Erfassung dieser Daten erfolgt automatisch, sobald Sie diese App benutzen.

Wofür nutzen wir Ihre Daten?
Ein Teil der Daten wird erhoben, um eine fehlerfreie Bereitstellung der App zu gewährleisten. Andere Daten können zur Analyse Ihres Nutzerverhaltens verwendet werden.

Welche Rechte haben Sie bezüglich Ihrer Daten?
Sie haben jederzeit das Recht, unentgeltlich Auskunft über Herkunft, Empfänger und Zweck Ihrer gespeicherten personenbezogenen Daten zu erhalten. Sie haben außerdem ein Recht, die Berichtigung oder Löschung dieser Daten zu verlangen. Wenn Sie eine Einwilligung zur Datenverarbeitung erteilt haben, können Sie diese Einwilligung jederzeit für die Zukunft widerrufen. Außerdem haben Sie das Recht, unter bestimmten Umständen die Einschränkung der Verarbeitung Ihrer personenbezogenen Daten zu verlangen. Des Weiteren steht Ihnen ein Beschwerderecht bei der zuständigen Aufsichtsbehörde zu. Hierzu sowie zu weiteren Fragen zum Thema Datenschutz können Sie sich jederzeit an uns wenden.

Analyse-Tools und Tools von Drittanbietern

Beim Besuch dieser App kann Ihr Surf-Verhalten statistisch ausgewertet werden. Das geschieht vor allem mit sogenannten Analyseprogrammen.
Detaillierte Informationen zu diesen Analyseprogrammen finden Sie in der folgenden Datenschutzerklärung.

2. Allgemeine Hinweise und Pflichtinformationen

Datenschutz

Die Betreiber dieser App nehmen den Schutz Ihrer persönlichen Daten sehr ernst. Wir behandeln Ihre personenbezogenen Daten vertraulich und entsprechend den gesetzlichen Datenschutzvorschriften sowie dieser Datenschutzerklärung.

Wenn Sie diese App benutzen, werden verschiedene personenbezogene Daten erhoben. Personenbezogene Daten sind Daten, mit denen Sie persönlich identifiziert werden können. Die vorliegende Datenschutzerklärung erläutert, welche Daten wir erheben und wofür wir sie nutzen. Sie erläutert auch, wie und zu welchem Zweck das geschieht.

Wir weisen daraufhin, dass die Datenübertragung im Internet (z. B. bei der Kommunikation per E-Mail) Sicherheitslücken aufweisen kann. Ein lückenloser Schutz der Daten vor dem Zugriff durch Dritte ist nicht möglich.

Hinweis zur verantwortlichen Stelle

Die verantwortliche Stelle für die Datenverarbeitung in dieser App ist:

AMBER Germany
Musterstraße 1
12345 Musterhausen

Telefon: 01234/56789
E-Mail: amber@muster.de

Verantwortliche Stelle ist die natürliche oder juristische Person, die allein oder gemeinsam mit anderen über die Zwecke und Mittel der Verarbeitung von personenbezogenen Daten (z. B. Namen, E-Mail-Adressen o. Ä.) entscheidet.

Speicherdauer

Soweit innerhalb dieser Datenschutzerklärung keine speziellere Speicherdauer genannt wurde, verbleiben Ihre personenbezogenen Daten bei uns, bis der Zweck für die Datenverarbeitung entfällt. Wenn Sie ein berechtigtes Löschersuchen geltend machen oder eine Einwilligung zur Datenverarbeitung widerrufen, werden Ihre Daten gelöscht, sofern wir keine anderen rechtlich zulässigen Gründe für die Speicherung Ihrer personenbezogenen Daten haben (z. B. steuer- oder handelsrechtliche Aufbewahrungsfristen); im letztgenannten Fall erfolgt die Löschung nach Fortfall dieser Gründe.

Allgemeine Hinweise zu den Rechtsgrundlagen der Datenverarbeitung innerhalb dieser App

Sofern Sie in die Datenverarbeitung eingewilligt haben, verarbeiten wir Ihre personenbezogenen Daten auf Grundlage von Art. 6 Abs. 1 lit. a DSGVO bzw. Art. 9 Abs. 2 lit. a DSGVO, sofern besondere Datenkategorien nach Art. 9 Abs. 1 DSGVO verarbeitet werden. Im Falle einer ausdrücklichen Einwilligung in die Übertragung personenbezogener Daten in Drittstaaten erfolgt die Datenverarbeitung außerdem auf Grundlage von Art. 49 Abs. 1 lit. a DSGVO. Sofern Sie in die Speicherung von Cookies oder in den Zugriff auf Informationen in Ihr Endgerät (z. B. via Device-Fingerprinting) eingewilligt haben, erfolgt die Datenverarbeitung zusätzlich auf Grundlage von § 25 Abs. 1 TTDSG. Die Einwilligung ist jederzeit widerrufbar. Sind Ihre Daten zur Vertragserfüllung oder zur Durchführung vorvertraglicher Maßnahmen erforderlich, verarbeiten wir Ihre Daten auf Grundlage des Art. 6 Abs. 1 lit. b DSGVO. Des Weiteren verarbeiten wir Ihre Daten, sofern diese zur Erfüllung einer rechtlichen Verpflichtung erforderlich sind auf Grundlage von Art. 6 Abs. 1 lit. c DSGVO. Die Datenverarbeitung kann ferner auf Grundlage unseres berechtigten Interesses nach Art. 6 Abs. 1 lit. f DSGVO erfolgen. Über die jeweils im Einzelfall einschlägigen Rechtsgrundlagen wird in den folgenden Absätzen dieser Datenschutzerklärung informiert.

Widerruf Ihrer Einwilligung zur Datenverarbeitung

Viele Datenverarbeitungsvorgänge sind nur mit Ihrer ausdrücklichen Einwilligung möglich. Sie können eine bereits erteilte Einwilligung jederzeit widerrufen. Die Rechtmäßigkeit der bis zum Widerruf erfolgten Datenverarbeitung bleibt vom Widerruf unberührt.

Widerspruchsrecht gegen die Datenerhebung in besonderen Fällen sowie gegen Direktwerbung (Art. 21 DSGVO)

WENN DIE DATENVERARBEITUNG AUF GRUNDLAGE VON ART. 6 ABS. 1 LIT. E ODER F DSGVO ERFOLGT, HABEN SIE JEDERZEIT DAS RECHT, AUS GRÜNDEN, DIE SICH AUS IHRER BESONDEREN SITUATION ERGEBEN, GEGEN DIE VERARBEITUNG IHRER PERSONENBEZOGENEN DATEN WIDERSPRUCH EINZULEGEN; DIES GILT AUCH FÜR EIN AUF DIESE BESTIMMUNGEN GESTÜTZTES PROFILING. DIE JEWEILIGE RECHTSGRUNDLAGE, AUF DENEN EINE VERARBEITUNG BERUHT, ENTNEHMEN SIE DIESER DATENSCHUTZERKLÄRUNG. WENN SIE WIDERSPRUCH EINLEGEN, WERDEN WIR IHRE BETROFFENEN PERSONENBEZOGENEN DATEN NICHT MEHR VERARBEITEN, ES SEI DENN, WIR KÖNNEN ZWINGENDE SCHUTZWÜRDIGE GRÜNDE FÜR DIE VERARBEITUNG NACHWEISEN, DIE IHRE INTERESSEN, RECHTE UND FREIHEITEN ÜBERWIEGEN ODER DIE VERARBEITUNG DIENT DER GELTENDMACHUNG, AUSÜBUNG ODER VERTEIDIGUNG VON RECHTSANSPRÜCHEN (WIDERSPRUCH NACH ART. 21 ABS. 1 DSGVO).

WERDEN IHRE PERSONENBEZOGENEN DATEN VERARBEITET, UM DIREKTWERBUNG ZU BETREIBEN, SO HABEN SIE DAS RECHT, JEDERZEIT WIDERSPRUCH

GEGEN DIE VERARBEITUNG SIE BETREFFENDER PERSONENBEZOGENER DATEN ZUM ZWECKE DERARTIGER WERBUNG EINZULEGEN; DIES GILT AUCH FÜR DAS PROFILING, SOWEIT ES MIT SOLCHER DIREKTWERBUNG IN VERBINDUNG STEHT. WENN SIE WIDERSPRECHEN, WERDEN IHRE PERSONENBEZOGENEN DATEN ANSCHLIESSEND NICHT MEHR ZUM ZWECKE DER DIREKTWERBUNG VERWENDET (WIDERSPRUCH NACH ART. 21 ABS. 2 DSGVO).

Beschwerderecht bei der zuständigen Aufsichtsbehörde

Im Falle von Verstößen gegen die DSGVO steht den Betroffenen ein Beschwerderecht bei einer Aufsichtsbehörde, insbesondere in dem Mitgliedstaat ihres gewöhnlichen Aufenthalts, ihres Arbeitsplatzes oder des Orts des mutmaßlichen Verstoßes zu. Das Beschwerderecht besteht unbeschadet anderweitiger verwaltungsrechtlicher oder gerichtlicher Rechtsbehelfe.

Recht auf Datenübertragbarkeit

Sie haben das Recht, Daten, die wir auf Grundlage Ihrer Einwilligung oder in Erfüllung eines Vertrags automatisiert verarbeiten, an sich oder an einen Dritten in einem gängigen, maschinenlesbaren Format aushändigen zu lassen. Sofern Sie die direkte Übertragung der Daten an einen anderen Verantwortlichen verlangen, erfolgt dies nur, soweit es technisch machbar ist.

Auskunft, Löschung und Berichtigung

Sie haben im Rahmen der geltenden gesetzlichen Bestimmungen jederzeit das Recht auf unentgeltliche Auskunft über Ihre gespeicherten personenbezogenen Daten, deren Herkunft und Empfänger und den Zweck der Datenverarbeitung und ggf. ein Recht auf Berichtigung oder Löschung dieser Daten. Hierzu sowie zu weiteren Fragen zum Thema personenbezogene Daten können Sie sich jederzeit an uns wenden.

Recht auf Einschränkung der Verarbeitung

Sie haben das Recht, die Einschränkung der Verarbeitung Ihrer personenbezogenen Daten zu verlangen. Hierzu können Sie sich jederzeit an uns wenden. Das Recht auf Einschränkung der Verarbeitung besteht in folgenden Fällen:

- Wenn Sie die Richtigkeit Ihrer bei uns gespeicherten personenbezogenen Daten bestreiten, benötigen wir in der Regel Zeit, um dies zu überprüfen. Für die Dauer der Prüfung haben Sie das Recht, die Einschränkung der Verarbeitung Ihrer personenbezogenen Daten zu verlangen.

- Wenn die Verarbeitung Ihrer personenbezogenen Daten unrechtmäßig geschah/geschieht, können Sie statt der Löschung die Einschränkung der Datenverarbeitung verlangen.

- Wenn wir Ihre personenbezogenen Daten nicht mehr benötigen, Sie sie jedoch zur

Ausübung, Verteidigung oder Geltendmachung von Rechtsansprüchen benötigen, haben Sie das Recht, statt der Löschung die Einschränkung der Verarbeitung Ihrer personenbezogenen Daten zu verlangen.

- Wenn Sie einen Widerspruch nach Art. 21 Abs. 1 DSGVO eingelegt haben, muss eine Abwägung zwischen Ihren und unseren Interessen vorgenommen werden. Solange noch nicht feststeht, wessen Interessen überwiegen, haben Sie das Recht, die Einschränkung der Verarbeitung Ihrer personenbezogenen Daten zu verlangen.

Wenn Sie die Verarbeitung Ihrer personenbezogenen Daten eingeschränkt haben, dürfen diese Daten – von ihrer Speicherung abgesehen – nur mit Ihrer Einwilligung oder zur Geltendmachung, Ausübung oder Verteidigung von Rechtsansprüchen oder zum Schutz der Rechte einer anderen natürlichen oder juristischen Person oder aus Gründen eines wichtigen öffentlichen Interesses der Europäischen Union oder eines Mitgliedstaats verarbeitet werden.

3. Datenerfassung in dieser App

Anfrage per E-Mail oder Telefon

Wenn Sie uns per E-Mail oder Telefon kontaktieren, wird Ihre Anfrage inklusive aller daraus hervorgehenden personenbezogenen Daten (Name, Anfrage) zum Zwecke der Bearbeitung Ihres Anliegens bei uns gespeichert und verarbeitet. Diese Daten geben wir nicht ohne Ihre Einwilligung weiter.

Die Verarbeitung dieser Daten erfolgt auf Grundlage von Art. 6 Abs. 1 lit. b DSGVO, sofern Ihre Anfrage mit der Erfüllung eines Vertrags zusammenhängt oder zur Durchführung vorvertraglicher Maßnahmen erforderlich ist. In allen übrigen Fällen beruht die Verarbeitung auf unserem berechtigten Interesse an der effektiven Bearbeitung der an uns gerichteten Anfragen (Art. 6 Abs. 1 lit. f DSGVO) oder auf Ihrer Einwilligung (Art. 6 Abs. 1 lit. a DSGVO) sofern diese abgefragt wurde; die Einwilligung ist jederzeit widerrufbar.

Die von Ihnen an uns per Kontaktanfragen übersandten Daten verbleiben bei uns, bis Sie uns zur Löschung auffordern, Ihre Einwilligung zur Speicherung widerrufen oder der Zweck für die Datenspeicherung entfällt (z. B. nach abgeschlossener Bearbeitung Ihres Anliegens). Zwingende gesetzliche Bestimmungen – insbesondere gesetzliche Aufbewahrungsfristen – bleiben unberührt.

Anwendungsdaten

Die App übermittelt während der Kommunikation mit unserem Server automatisch Daten, welche vom Server automatisch in sogenannten Log-Dateien gespeichert werden. Diese Daten sind:

- Art des mobilen Endgerätes

- Verwendetes Betriebssystem
- Verwendete Sprache
- Technische Informationen über das verwendete Endgerät
- Datum und Uhrzeit der Anfrage
- IP-Adresse
- Mobilfunknummer und Gerätenummer
- Aktuelle Geokoordinaten des mobilen Endgerätes
- Benutzernamen, sowie Vor- und Familiennamen, Telefonnummern und E-Mail-Adressen von Mitarbeitern

Dies erfolgt auf Grundlage von Art. 6 Abs. 1 lit. b DSGVO, der die Verarbeitung von Daten zur Erfüllung eines Vertrags oder vorvertraglicher Maßnahmen gestattet.

Adressdaten auf Ihrem Endgerät

Wenn Sie Informationen aus den Apps an Empfänger versenden wollen, greifen unsere Apps auf das Adressbuch zu. Die aus dem Adressbuch selektierten Daten werden nur zur Adressierung und Versendung der gewählten Informationen verwendet und nicht weiter gespeichert. Dabei handelt es sich um:

- Vollständiger Name
- Mobiltelefonnummer
- E-Mail-Adresse

Dies erfolgt auf Grundlage von Art. 6 Abs. 1 lit. b DSGVO, der die Verarbeitung von Daten zur Erfüllung eines Vertrags oder vorvertraglicher Maßnahmen gestattet.

Nutzung von Google Analytics for Firebase

Google Analytics für Firebase oder Firebase Analytics ist ein Analyse-Dienst von Google LLC. Weitere Informationen zur Verwendung von Daten bei Google sind in der Partner-Richtlinie von Google einsehbar. Firebase Analytics kann Daten mit anderen von Firebase bereitgestellten Tools wie Crash Reporting, Authentication, Remote Config oder Notifications gemeinsam nutzen. Der Nutzer kann diese Datenschutzerklärung überprüfen, um eine ausführliche Erläuterung zu den anderen vom Eigentümer verwendeten Tools zu finden.
Diese Anwendung verwendet Identifikatoren für mobile Geräte (einschließlich Android Advertising ID bzw. Advertising Identifier für iOS) und Cookie-ähnliche Technologien für die Ausführung des Google Analytics for Firebase-Dienstes.
Nutzer können sich über die entsprechenden Geräteeinstellungen mobiler Geräte von bestimmten Firebase-Funktionen abmelden, wie etwa über die Werbeeinstellungen für mobile Geräte, oder indem sie gegebenenfalls den Anweisungen anderer Abschnitte dieser Datenschutzrichtlinie bezüglich Firebase folgen.
Erhobene personenbezogene Daten: Cookie, eindeutige Gerätekennzeichnung für Wer-

bung (z. B. Google-Werbe-ID oder IDFA) und Nutzungsdaten.
Verarbeitungsort: Vereinigte Staaten, Irland.

Nutzung von Firebase Realtime Database

Firebase Realtime Database ist ein Webhosting und Backend Dienst, bereitgestellt von Google LLC. Erhobene personenbezogene Daten: Nutzungsdaten; verschiedene Datenarten, wie in der Datenschutzerklärung des Dienstes beschrieben.
Verarbeitungsort: Vereinigte Staaten, Irland.

Nutzung von Firebase Cloud Storage

Firebase Cloud Storage ist ein Webhosting Dienst, bereitgestellt von Google LLC. Erhobene personenbezogene Daten: Nutzungsdaten; verschiedene Datenarten, wie in der Datenschutzerklärung des Dienstes beschrieben.
Verarbeitungsort: Vereinigte Staaten, Irland.

Nutzung von Firebase Cloud Messaging

Firebase Cloud Messaging ist ein Dienst für den Nachrichtenversand, bereitgestellt von Google LLC. Firebase Cloud Messaging ermöglicht dem Eigentümer, über Plattformen wie Android, iOS und das Web Nachrichten und Benachrichtigungen an Nutzer zu senden. Nachrichten können an einzelne Geräte, Gerätegruppen, zu bestimmten Themen oder an bestimmte Benutzersegmente gesendet werden. Erhobene personenbezogene Daten: verschiedene Datenarten, wie in der Datenschutzerklärung des Dienstes beschrieben.
Verarbeitungsort: Vereinigte Staaten, Irland.

Quelle:
https://www.e-recht24.de

B.3 fragment_legal_notice.xml

Impressum

Angaben gemäß § 5 TMG:

AMBER Germany
Musterstraße 1
12345 Musterhausen

Vertreten durch:

Max Mustermann

Kontakt:

Telefon: 01234/56789
E-Mail: amber@muster.de

Verantwortlich für den Inhalt:

Max Mustermann

Registereintrag:

Registergericht: Amtsgericht Musterhausen
Registernummer: ABC 123456 D
USt-ID: DE123456789

Haftungsausschluss

Haftung für Inhalte:

Die Inhalte dieser App wurden mit größter Sorgfalt erstellt. Für die Richtigkeit, Vollständigkeit und Aktualität der Inhalte wird keine Gewähr übernommen. Als Diensteanbieter sind wir gemäß § 7 Abs. 1 TMG für eigene Inhalte in dieser App den allgemeinen Gesetzen verantwortlich. Nach §§ 8 bis 10 TMG sind wir als Diensteanbieter jedoch nicht verpflichtet, übermittelte oder gespeicherte fremde Informationen zu überwachen oder nach Umständen zu forschen, die auf eine rechtswidrige Tätigkeit hinweisen.
Verpflichtungen zur Entfernung oder Sperrung der Nutzung von Informationen nach den allgemeinen Gesetzen bleiben hiervon unberührt. Eine diesbezügliche Haftung ist jedoch erst ab dem Zeitpunkt der Kenntnis einer konkreten Rechtsverletzung möglich. Bei Bekanntwerden von entsprechenden Rechtsverletzungen werden wir diese Inhalte umgehend entfernen.

Haftung für Links:

Unser Angebot enthält Links zu externen Websites Dritter, auf deren Inhalte wir keinen Einfluss haben. Deshalb können wir für diese fremden Inhalte auch keine Gewähr übernehmen. Für die Inhalte der verlinkten Seiten ist stets der jeweilige Anbieter oder Betreiber der Seiten verantwortlich. Die verlinkten Seiten wurden zum Zeitpunkt der

Verlinkung auf mögliche Rechtsverstöße überprüft. Rechtswidrige Inhalte waren zum Zeitpunkt der Verlinkung nicht erkennbar.
Eine permanente inhaltliche Kontrolle der verlinkten Seiten ist jedoch ohne konkrete Anhaltspunkte einer Rechtsverletzung nicht zumutbar. Bei Bekanntwerden von Rechtsverletzungen werden wir derartige Links umgehend entfernen.

Urheberrecht:

Die durch die Seitenbetreiber erstellten Inhalte und Werke auf diesen Seiten unterliegen dem deutschen Urheberrecht. Die Vervielfältigung, Bearbeitung, Verbreitung und jede Art der Verwertung außerhalb der Grenzen des Urheberrechtes bedürfen der schriftlichen Zustimmung des jeweiligen Autors bzw. Erstellers. Downloads und Kopien dieser Seite sind nur für den privaten, nicht kommerziellen Gebrauch gestattet.
Soweit die Inhalte auf dieser Seite nicht vom Betreiber erstellt wurden, werden die Urheberrechte Dritter beachtet. Insbesondere werden Inhalte Dritter als solche gekennzeichnet. Sollten Sie trotzdem auf eine Urheberrechtsverletzung aufmerksam werden, bitten wir um einen entsprechenden Hinweis. Bei Bekanntwerden von Rechtsverletzungen werden wir derartige Inhalte umgehend entfernen.

Quelle Haftungsausschluss:
https://www.e-recht24.de/muster-disclaimer.html

Anhang C: E-Mail Korrespondenz mit C. Schippers (AMBER Alert Europe) vom 12. Mai 2022

„Just as some context for my answers and suggestions, I'm a recently retired police officer with 30 years of experience in missing persons/children cases and have been involved in developing the AMBER Alert system, and the AMBER Alert app, in The Netherlands from the beginning. I've put in writing things that came to my mind, but I'm sure I have overlooked some aspects, so please feel free to let me know if you have any additional questions.

1. Decide who should be the app's owner and user. Are you thinking of an organization like Deutschland Findet Euch or other non-profit (NGO), or could it be a police force? This is important for the following reasons:
   a. Police can verify that the child's personal information is correct;
   b. Police can investigate that the child is actually missing and the case meets their criteria for publication;
   c. Police has access to information in their systems about potential previous events related to the missing child;
   d. Police may be able to determine that using social media would/could be endangering the child or might be counterproductive;
   e. Most of this information cannot be shared by police, unless an NGO has a contract with police about this kind of cooperation;
   f. The parents of the missing child will have to give their written consent to publish their child's information and photograph, to police as well as an NGO.
2. Determine what the criteria should be to show a missing child to the public using the app. Should all missing children be put in there? If not, what would these criteria be?
3. Consider what information about the child should be in the app. The basic information covers the child's name (first and family name, or just first name?), the child's age (and/or date of birth) and a recent (portait like) photograph.
4. A physical description of the child is also needed. Think of hair color, hair length and model, color of the eyes, length, and such. Please have a look at the Dutch Police website at Aysha Opdam | politie.nl, to see what information could be used.
5. A brief description of the circumstances of the disappearance will be the motivation for the public to pay attention.
6. For both 5 and 6 you need to realize that the information provided may not harm the child or the child's interests. Information that is "too personal" needs to be avoided. Think of a medical condition, the intention to commit suicide or other

problems the child may have.

7. As photographs of the missing child are rarely so good that they can be used without editing, it would be a major plus if the app would offer some simple tools to do this, like cropping (to get other people or objects out of the picture) and resizing;
8. Consider your action once the child has been located and the feedback you, the app, will give to the public. It is not complicated to inform the public that the child has been found alive and well, but what if the child is deceased.
9. The app could potentially provide a link to a website (police or NGO) with more information, more pictures and such. With a lot of app users and in high profile cases such a link may generate substantial volumes of data traffic to such a site.
10. Do you want the app to allow people to register themselves, for instance for their location information, allowing you to geographically target them for an alert?“

Literaturverzeichnis

[1] Bundeskriminalamt. Die polizeiliche Bearbeitung von Vermisstenfällen in Deutschland. Online verfügbar unter https://www.bka.de/DE/UnsereAufgaben/Ermittlungsunterstuetzung/BearbeitungVermisstenfaelle/bearbeitungVermisstenfaelle.html, zuletzt geprüft am 09.09.2022, 2022.

[2] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. Frequently Asked Questions. Online verfügbar unter https://amberalert.ojp.gov/about/faqs, zuletzt geprüft am 04.04.2022, 2019.

[3] Federal Emergency Management Agency. Emergency Alert System. U.S. Department of Homeland Security. Online verfügbar unter https://www.fema.gov/emergency-managers/practitioners/integrated-public-alert-warning-system/public/emergency-alert-system, zuletzt geprüft am 04.04.2022, 2022.

[4] Federal Emergency Management Agency. Wireless Emergency Alerts. U.S. Department of Homeland Security. Online verfügbar unter https://www.fema.gov/emergency-managers/practitioners/integrated-public-alert-warning-system/public/wireless-emergency-alerts, zuletzt geprüft am 04.04.2022, 2020.

[5] Statistisches Bundesamt. Daten aus den Laufenden Wirtschaftsrechnungen (LWR) zur Ausstattung privater Haushalte mit Informationstechnik. Online verfügbar unter https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Einkommen-Konsum-Lebensbedingungen/Ausstattung-Gebrauchsgueter/Tabellen/a-infotechnik-d-lwr.html, zuletzt geprüft am 11.05.2022, 2021.

[6] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. About AMBER Alert. Online verfügbar unter https://amberalert.ojp.gov/about, zuletzt geprüft am 04.04.2022, 2019.

[7] Initiative Vermisste Kinder. AMBER Alert. Online verfügbar unter https://www.amber-alert-deutschland.de/, zuletzt geprüft am 27.09.2022, o.D.

[8] Federal Emergency Management Agency. General Public. U.S. Department of Homeland Security. Online verfügbar unter https://www.fema.gov/emergency-managers/practitioners/integrated-public-alert-warning-system/public, zuletzt geprüft am 04.04.2022, 2021.

[9] Katherine M. Brown, Robert D. Kappel, Joseph G. Weis, and Marvin E. Skeen. Case management for missing children homicide investigations: Report II, Online verfügbar unter https://www.researchgate.net/publication/265602370_Case_management_for_missing_children_homicide_investigations_Report_II, zuletzt geprüft am 05.04.2022, 2006.

[10] U.S. Department of Justice, Office of Justice Programs, Office of Juvenile Justice and Delinquency Prevention. AMBER Alert Best Practices, Online verfügbar unter https://ojjdp.ojp.gov/sites/g/files/xyckuh176/files/pubs/252759.pdf, zuletzt geprüft am 06.04.2022, 2019.

[11] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. Guidelines for Issuing AMBER Alerts. Online verfügbar unter https://amberalert.ojp.gov/about/guidelines-for-issuing-alerts, zuletzt geprüft am 04.04.2022, o.D.

[12] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. Secondary Distribution of AMBER Alerts. Online verfügbar unter https://amberalert.ojp.gov/resources/secondary-distribution-amber-alerts, zuletzt geprüft am 05.04.2022, 2019.

[13] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. AMBER Alert Secondary Distributors. Online verfügbar unter https://amberalert.ojp.gov/resources/amber-alert-secondary-distributors, zuletzt geprüft am 05.04.2022, 2019.

[14] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. What Happens When Law Enforcement Issues an AMBER Alert - flowchart. Online verfügbar unter https://amberalert.ojp.gov/media/image/711, zuletzt geprüft am 05.04.2022, o.D.

[15] National Center for Missing & Exploited Children®. 2020 AMBER Alert Report: Analysis of AMBER Alert Cases in 2020. Online verfügbar unter https://www.missingkids.org/content/dam/missingkids/pdfs/amber/2020%20Annual%20AMBER%20Alert%20Report%20and%20Map_FINAL.pdf, zuletzt geprüft am 04.04.2022, 2021.

[16] U.S. Department of Justice, Office of Justice Programs, AMBER Alert. Statistics. Online verfügbar unter https://amberalert.ojp.gov/statistics, zuletzt geprüft am 06.04.2022, 2019.

[17] AMBER Alert Europe. AMBER Alert Europe Foundation. Online verfügbar unter https://www.amberalert.eu/amber-alert-europe/, zuletzt geprüft am 11.04.2022, 2022.

[18] AMBER Alert Europe. AMBER Alert Europe. Online verfügbar unter https://www.amberalert.eu/, zuletzt geprüft am 11.04.2022, 2022.

[19] AMBER Alert Europe. Our work: ACTIVITIES AND PROJECTS Online verfügbar unter https://www.amberalert.eu/projects, zuletzt geprüft am 28.09.2022, 2022.

[20] Ministère de la Justice, Alerte-enlèvement. Le dispositif Alerte enlèvement. Online verfügbar unter http://www.alerte-enlevement.gouv.fr/, zuletzt geprüft am 11.04.2022, 2021.

[21] AMBER Alert Luxemburg. Differenzierungsmerkmale & Begriffe. Online verfügbar unter https://www.amberalert.lu/de/differenzierungsmerkmale-begriffe/, zuletzt geprüft am 11.04.2022, 2016.

[22] AMBER Alert Europe. AMBER Alert in your country. Online verfügbar unter https://www.amberalert.eu/amber-alert-in-your-country/, zuletzt geprüft am 11.04.2022, 2022.

[23] Initiative Vermisste Kinder. Initiative Vermisste Kinder. Online verfügbar unter https://www.initiative-vermisste-kinder.de/, zuletzt geprüft am 12.04.2022, o.D.

[24] Initiative Vermisste Kinder. ÜBER UNS. Online verfügbar unter https://www.initiative-vermisste-kinder.de/%C3%BCber-uns/, zuletzt geprüft am 12.04.2022, o.D.

[25] Christian Aichele and Marius Schönberger. *App-Entwicklung – effizient und erfolgreich*. Springer Fachmedien Wiesbaden, 2016.

[26] StatCounter. Mobile Operating System Market Share Worldwide: Mar 2022. Online verfügbar unter https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202203-202203-bar, zuletzt geprüft am 13.04.2022, 2022.

[27] Google. Die Geschichte von Android - Google. Online verfügbar unter https://about.google/intl/ALL_de/stories/geschichte-android/, zuletzt geprüft am 13.04.2022, o.D.

[28] Google Developers, Android for Developers. Platform Architecture. Online verfügbar unter https://developer.android.com/guide/platform, zuletzt geprüft am 12.04.2022, 2021.

[29] Google Developers, Android for Developers. Android Releases. Online verfügbar unter https://developer.android.com/about/versions, zuletzt geprüft am 13.04.2022, 2022.

[30] Google Developers, Android for Developers. Features and APIs Overview. Online verfügbar unter https://developer.android.com/about/versions/12/features, zuletzt geprüft am 13.04.2022, 2022.

[31] Google Developers, Android for Developers. Device compatibility overview. Online verfügbar unter https://developer.android.com/guide/practices/compatibility, zuletzt geprüft am 13.04.2022, 2022.

[32] Android Open Source Project. System and kernel security. Online verfügbar unter https://source.android.com/docs/security/overview/kernel-security, zuletzt geprüft am 29.09.2022, 2022.

[33] Google Developers, Android for Developers. Application Fundamentals. Online verfügbar unter https://developer.android.com/guide/components/fundamentals, zuletzt geprüft am 02.05.2022, 2021.

[34] Google Developers, Android for Developers. Migrate to Android Studio. Online verfügbar unter https://developer.android.com/studio/intro/migrate, zuletzt geprüft am 27.04.2022, 2022.

[35] Google Developers, Android for Developers. Meet Android Studio. Online verfügbar unter https://developer.android.com/studio/intro, zuletzt geprüft am 27.04.2022, 2022.

[36] Google Developers, Android for Developers. Introduction to Activities. Online verfügbar unter https://developer.android.com/guide/components/activities/intro-activities, zuletzt geprüft am 28.04.2022, 2021.

[37] Google Developers, Android for Developers. Fragments . Online verfügbar unter https://developer.android.com/guide/fragments, zuletzt geprüft am 28.04.2022, 2021.

[38] Google Developers, Android for Developers. The Activity Lifecycle. Online verfügbar unter https://developer.android.com/guide/components/activities/activity-lifecycle, zuletzt geprüft am 19.08.2022, 2021.

[39] Google Developers, Android for Developers. Fragment. Online verfügbar unter https://developer.android.com/reference/android/app/Fragment, zuletzt geprüft am 19.08.2022, 2022.

[40] Google Developers, Android for Developers. Broadcasts overview . Online verfügbar unter https://developer.android.com/guide/components/broadcasts, zuletzt geprüft am 02.05.2022, 2022.

[41] Google Developers, Android for Developers. App resources overview. Online verfügbar unter https://developer.android.com/guide/topics/resources/providing-resources, zuletzt geprüft am 26.04.2022, 2022.

[42] Google Developers, Android for Developers. Get started with the Navigation component. Online verfügbar unter https://developer.android.com/guide/navigation/navigation-getting-started, zuletzt geprüft am 26.04.2022, 2022.

[43] Google Developers, Android for Developers. Layouts. Online verfügbar unter https://developer.android.com/guide/topics/ui/declaring-layout, zuletzt geprüft am 02.05.2022, 2021.

[44] Google Developers, Android for Developers. ImageView. Online verfügbar unter https://developer.android.com/reference/android/widget/ImageView, zuletzt geprüft am 24.05.2022, 2022.

[45] Google Developers, Android for Developers. ListView. Online verfügbar unter https://developer.android.com/reference/android/widget/ListView, zuletzt geprüft am 24.05.2022, 2022.

[46] Google Developers, Android for Developers. NavigationView. Online verfügbar unter https://developer.android.com/reference/com/google/android/material/navigation/NavigationView, zuletzt geprüft am 24.05.2022, 2022.

[47] Google Developers, Android for Developers. TextView. Online verfügbar unter https://developer.android.com/reference/android/widget/TextView, zuletzt geprüft am 24.05.2022, 2022.

[48] Google Developers, Android for Developers. AppBarLayout. Online verfügbar unter https://developer.android.com/reference/com/google/android/material/appbar/AppBarLayout, zuletzt geprüft am 24.05.2022, 2022.

[49] Google Developers, Android for Developers. ConstraintLayout. Online verfügbar unter https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout, zuletzt geprüft am 24.05.2022, 2022.

[50] Google Developers, Android for Developers. CoordinatorLayout. Online verfügbar unter https://developer.android.com/reference/androidx/

coordinatorlayout/widget/CoordinatorLayout, zuletzt geprüft am 24.05.2022, 2022.

[51] Google Developers, Android for Developers. DrawerLayout. Online verfügbar unter https://developer.android.com/reference/androidx/drawerlayout/widget/DrawerLayout, zuletzt geprüft am 24.05.2022, 2022.

[52] Google Developers, Android for Developers. FrameLayout. Online verfügbar unter https://developer.android.com/reference/android/widget/FrameLayout, zuletzt geprüft am 24.05.2022, 2022.

[53] Google Developers, Android for Developers. LinearLayout. Online verfügbar unter https://developer.android.com/reference/android/widget/LinearLayout, zuletzt geprüft am 24.05.2022, 2022.

[54] Google Developers, Android for Developers. RelativeLayout. Online verfügbar unter https://developer.android.com/reference/android/widget/RelativeLayout, zuletzt geprüft am 24.05.2022, 2022.

[55] Google Developers, Android for Developers. ScrollView. Online verfügbar unter https://developer.android.com/reference/android/widget/ScrollView, zuletzt geprüft am 24.05.2022, 2022.

[56] Google Developers, Android for Developers. Toolbar. Online verfügbar unter https://developer.android.com/reference/android/widget/Toolbar, zuletzt geprüft am 24.05.2022, 2022.

[57] Google Developers, Android for Developers. AdapterView.OnItemClickListener. Online verfügbar unter https://developer.android.com/reference/android/widget/AdapterView.OnItemClickListener, zuletzt geprüft am 20.08.2022, 2022.

[58] Google Developers, Android for Developers. Input events overview. Online verfügbar unter https://developer.android.com/guide/topics/ui/ui-events, zuletzt geprüft am 20.08.2022, 2021.

[59] Google Developers, Firebase. Make your app the best it can be. Online verfügbar unter https://firebase.google.com/, zuletzt geprüft am 23.05.2022, o.D.

[60] Google Developers, Firebase. Products / Build: Accelerate and scale app development without managing infrastructure. Online verfügbar unter https://firebase.google.com/products-build, zuletzt geprüft am 23.05.2022, o.D.

[61] Google Developers, Firebase. Products / Release & Monitor: Release with con-

fidence and monitor performance and stability. Online verfügbar unter https://firebase.google.com/products-release, zuletzt geprüft am 23.05.2022, o.D.

[62] Google Developers, Firebase. Products / Engage: Boost engagement with rich analytics, A/B testing, and messaging campaigns. Online verfügbar unter https://firebase.google.com/products-engage, zuletzt geprüft am 23.05.2022, o.D.

[63] Peter Späth and Jeff Friesen. *Learn Java for Android Development*. Apress, 2020.

[64] Oracle. Java Downloads. Online verfügbar unter https://www.oracle.com/java/technologies/downloads/, zuletzt geprüft am 04.08.2022, 2021.

[65] Google Developers, Android for Developers. Android's Kotlin-first approach. Online verfügbar unter https://developer.android.com/kotlin/first, zuletzt geprüft am 04.05.2022, 2021.

[66] Google Developers, Android for Developers. Kotlin on Android FAQ. Online verfügbar unter https://developer.android.com/kotlin/faq, zuletzt geprüft am 04.05.2022, 2020.

[67] Pierre Carbonnelle. PYPL PopularitY of Programming Language. Online verfügbar unter https://pypl.github.io/PYPL.html, zuletzt geprüft am 04.05.2022, 2022.

[68] Jeff Friesen. *Java XML and JSON*. Apress, 2019.

[69] Murmuras GmbH. Smartphone behavior during the Corona pandemic - Real app usage data from Germany. Online verfügbar unter https://murmuras.com/en/blog/app-usage-and-corona, zuletzt geprüft am 11.05.2022, 2020.

[70] StatCounter. Mobile Operating System Market share Germany. Online verfügbar unter https://gs.statcounter.com/os-market-share/mobile/germany/#monthly-202204-202204-bar, zuletzt geprüft am 11.05.2022, 2022.

[71] nhb studios. „Deutschland Findet Euch“ - Initiative vermisste Kinder. YouTube. Online verfügbar unter https://www.youtube.com/watch?v=6BWSNAK4LAs, zuletzt geprüft am 26.05.2022, 2011.

[72] Seven.One Entertainment Group. Umfrage zu „Welche Geräte mit Bildschirm sind in Ihrem Haushalt vorhanden?“, zititert nach de.statista.com. Online verfügbar unter https://de.statista.com/statistik/daten/studie/201970/umfrage/

verbreitung-von-bildschirm-medien-in-deutschen-haushalten/, zuletzt geprüft am 25.08.2022, 2011.

[73] Andreas Kirsten. Vermisst in Thüringen (Version 6.631) [Mobile app]. Google Play. Online verfügbar unter https://play.google.com/store/apps/details?id=com.Tobit.android.Slitte7180706553&hl=gsw&gl=US, zuletzt geprüft am 26.08.2022, 2021.

[74] Hessisches Ministerium des Innern und für Sport. hessenWARN. Online verfügbar unter https://innen.hessen.de/Sicherheit/hessenWARN, zuletzt geprüft am 05.09.2022, o.D.

[75] ChildRescue. ChildRescue: Collective Awareness Platform for Missing Children Investigation and Rescue. Online verfügbar unter https://www.childrescue.eu/, zuletzt geprüft am 25.08.2022, 2022.

[76] Google Developers, Android for Developers. Projects overview. Online verfügbar unter https://developer.android.com/studio/projects, zuletzt geprüft am 03.08.2022, 2021.

[77] Google Developers, Android for Developers. Update UI components with NavigationUI. Online verfügbar unter https://developer.android.com/guide/navigation/navigation-ui, zuletzt geprüft am 19.08.2022, 2021.

[78] Google Developers, Android for Developers. Download files with Cloud Storage on Android. Online verfügbar unter https://firebase.google.com/docs/storage/android/download-files, zuletzt geprüft am 13.08.2022, 2022.

[79] Nathan Wharry Sam Stern, Alex Saveau and Meir Fischer. FirebaseUI-Android/README.md: FirebaseUI for Storage. Online verfügbar unter https://github.com/firebase/FirebaseUI-Android/blob/master/storage/README.md, zuletzt geprüft am 13.08.2022, 2018.

[80] Sam Stern and Alex Saveau. FirebaseUI-Android/MyAppGlideModule.java. Online verfügbar unter https://github.com/firebase/FirebaseUI-Android/blob/master/app/src/main/java/com/firebase/uidemo/storage/MyAppGlideModule.java, zuletzt geprüft am 13.08.2022, 2019.

[81] Bundesamt für Bevölkerungsschutz und Katastrophenhilfe. Warn-App NINA. Online verfügbar unter https://www.bbk.bund.de/DE/Warnung-Vorsorge/Warn-App-NINA/warn-app-nina_node.html, zuletzt geprüft am 05.09.2022, o.D.

[82] Fraunhofer Institut für Offene Kommunikationssysteme FOKUS. Das Warnsystem.

Online verfügbar unter https://www.katwarn.de/warnsystem.php, zuletzt geprüft am 05.09.2022, o.D.

[83] Vermisstensuche per Katwarn-App. Frankfurter Neue Presse. Online verfügbar unter https://www.fnp.de/hessen/vermisstensuche-katwarn-app-10378303.html, zuletzt geprüft am 05.09.2022, 2018.

[84] Bundesamt für Bevölkerungsschutz und Katastrophenhilfe. Cell Broadcast. Online verfügbar unter https://www.bbk.bund.de/DE/Warnung-Vorsorge/Warnung-in-Deutschland/Warnmittel/Cell-Broadcast/cell-broadcast_node.html, zuletzt geprüft am 06.09.2022, o.D.

[85] Google Developers, Android for Developers. Google Play. Online verfügbar unter https://developer.android.com/distribute, zuletzt geprüft am 07.09.2022, o.D.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 4. Oktober 2022