



IT-HTL YBBS AN DER DONAU
HÖHERE TECHNISCHE LEHRANSTALT
FÜR INFORMATIONSTECHNOLOGIE
AUSBILDUNGSSCHWERPUNKT MEDIENTECHNIK



DIPLOMARBEIT

BambiGuard Die Rehkitzrettungsapp

Ausgeführt im Schuljahr 2021/22 von:

Maximilian Strobl	5AHITM
Clemens Losbichler	5AHITM
Felix Teufel	5AHITM

Betreuer/Betreuerin:

DI Dr. Hildegard Rumetshofer
Markus Meyerhofer, MSc

Ybbs an der Donau, am 21.03.2022

Abgabevermerk:

Datum:

Betreuer:

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Ybbs an der Donau, am 04.04.2022

Verfasser/Verfasserinnen:

Maximilian Strobl

Clemens Losbichler

Felix Teufel

Kurzfassung der Diplomarbeit/Abstract

	HÖHERE TECHNISCHE BUNDESLEHRANSTALT Ybbs/Donau
	Fachrichtung: Informationstechnologie Ausbildungsschwerpunkt: Medientechnik

DIPLOMARBEIT DOKUMENTATION

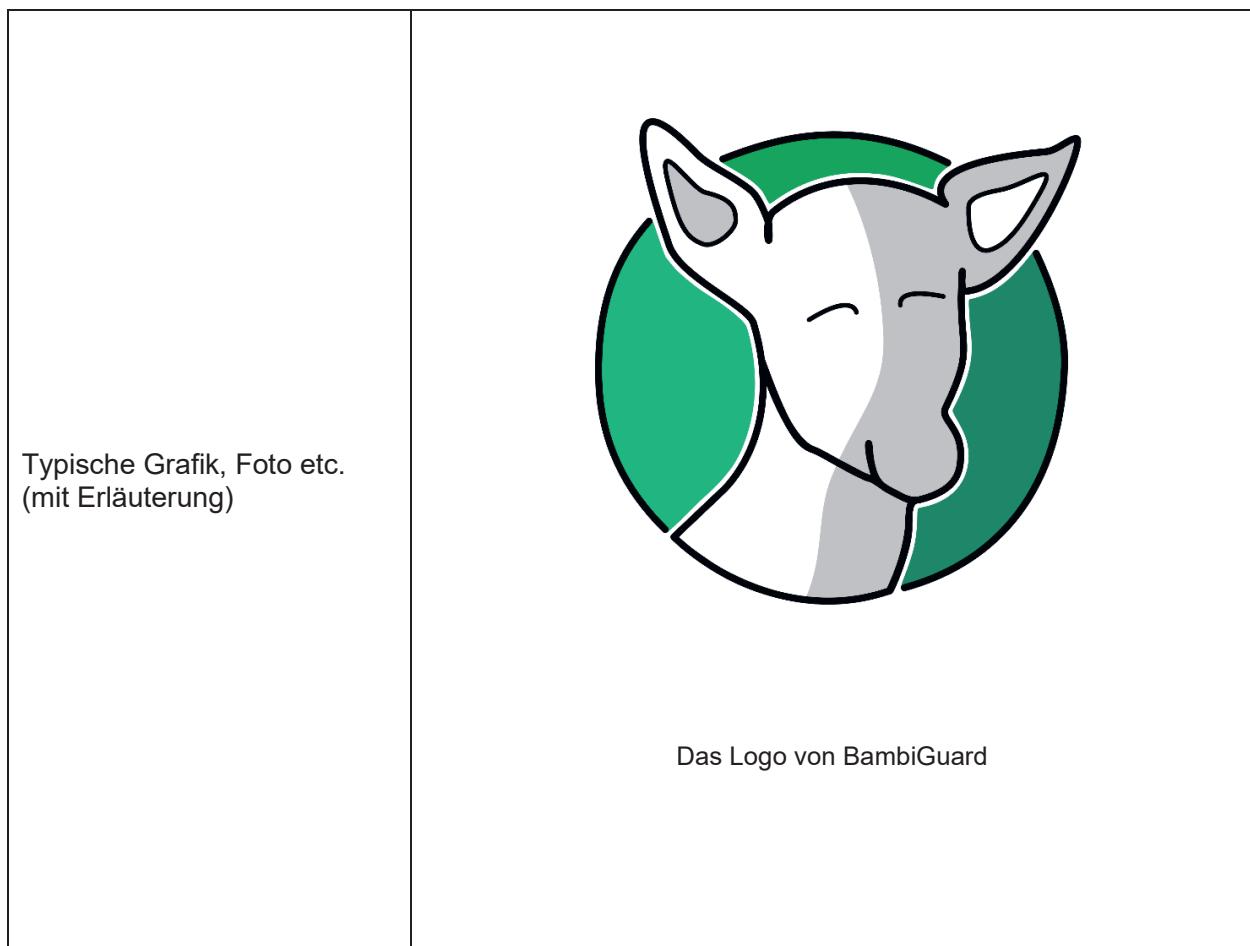
Namen der Verfasser/innen	Maximilian Strobl Felix Teufel Clemens Losbichler
Jahrgang Schuljahr	5AHITM 2021/22
Thema der Diplomarbeit	Erkennung und Rettung von Rehkitzen in Feldern mithilfe einer Drohne mit Wärmebildkamera gesteuert von einer Smartphone App.
Kooperationspartner	Verein für Förderung von Forschung für Innovation (VFI)

Aufgabenstellung	Die Rettung von Rehkitzen vor dem Tod durch Mäharbeiten erfolgt, wenn überhaupt in den meisten Fällen händisch, also durch Absuchen des Feldes oder Vergrämung der Tiere. Um die Effektivität und Effizienz zu steigern, soll ein System entwickelt werden, welches durch den Einsatz einer Drohne mit montierter Wärmebildkamera Wildtiere automatisch erkennt und Helfer mittels einer Smartphone App zum Fundort leitet.
------------------	--

Realisierung	Die App wurde für die Betriebssysteme Android und iOS nativ, also ohne Framework entwickelt. Eine Drohne des Herstellers DJI wird mithilfe des mitgelieferten SDK gesteuert sowie dessen Bilddaten ausgelesen. Die Drohne trägt eine Wärmebildkamera, mit welcher die Wildtiere in einem Feld gut erkennbar sind. Ein Algorithmus zur Erkennung der Tiere wurde in C++ mit der Bibliothek OpenCV geschrieben und in die App eingebunden. Helfer mit deren Smartphones und der gleichen App werden in den Flug eingebunden und erhalten bei einem Fund die Koordinaten des Tieres. Sie werden wie mit einem Kompass zum Fundort gelotst. Die Kommunikation erfolgt mit Websockets über das Internet und einen zentralen Server. Dazu ergänzend sind eine Website und diverse Infomaterialien zur Diplomarbeit und dem Themengebiet erstellt worden, welche Interessierte und Betroffene aufmerksam machen sollen.
--------------	--

Ergebnisse	Als Ergebnis liegt eine Android und iOS App vor, welche mit der Drohne kommunizieren, die Flugplanung durchführen, Helfer lotsen und die Drohne steuern kann. Zusätzlich besteht eine Website, welche die Diplomarbeit genauer beschreibt und das Team vorstellt. Druckmaterealien wie etwa Flyer oder Visitenkarten wurden erstellt und sind für den Druck aufbereitet worden. Zwei Social-Media-Kanäle auf Facebook und Instagram wurden mit eigens erstellten Inhalten gefüllt.
------------	---

htl bildung mit zukunft	HÖHERE TECHNISCHE BUNDESLEHRANSTALT Ybbs/Donau
Fachrichtung: Ausbildungsschwerpunkt:	Informationstechnologie Medientechnik



Teilnahme an Wettbewerben, Auszeichnungen	Bosch Technik fürs Leben Energy Globe Award Jugend Innovativ #35 Mostviertler Schul-Innovativpreis Sustainability Award FH OÖ Campus Wels 2022
--	--

Möglichkeiten der Einsichtnahme in die Arbeit	Archiv der Schule
--	-------------------

Approbation (Datum / Unterschrift)	Prüfer/Prüferin	Direktor/Direktorin Abteilungsvorstand/Abteilungsvorständin
---------------------------------------	-----------------	--

	College of Information Technology Ybbs/Donau
Department: Educational focus:	Information Technology Media Technology

DIPLOMA THESIS DOCUMENTATION

Author(s)	Maximilian Strobl Felix Teufel Clemens Losbichler
Form Academic year	5AHITM 2021/22
Topic	Recognition and Rescue of fawns in fields of grass with the help of a thermal camera drone controlled by a smartphone app.
Co-operation Partners	Verein für Förderung von Forschung für Innovation (VFI)

Assignment of Tasks	The rescue of fawns from death by mowing is, if at all, mostly done manually, i.e., by searching the field or scaring away the animals. To increase effectiveness and efficiency, a system is going to be developed that automatically detects wild animals by using a thermal imaging drone and guides helpers to the location using a smartphone app.
---------------------	---

Realisation	The app was developed natively for the Android and iOS operating systems. A drone from the manufacturer DJI is being controlled using the included SDK and its image data is being read out by the application. The drone carries a thermal imaging camera which detects wild animals in the field. An algorithm for recognizing the animals was written in C++ with the OpenCV library and integrated into the app. Helpers with their smartphones and the same app are integrated into the flight and receive the coordinates of the animal when it is found. They are guided to the location of the animal, like with a compass. Communication takes place with websockets via the Internet and a central server. In addition, a website and various information materials on the thesis and the subject area have been created to draw the attention of interested parties and those affected.
-------------	--

Results	The result is an app for Android and iOS that communicates with the drone, controls it, acts as a compass for the helpers and leads through the flight planning. In addition, there is a website that describes the thesis in more detail and introduces the team. Print materials, such as flyers and business cards, have been created and are ready to print. Social media channels on Facebook and Instagram have been filled with content specific to the theme.
---------	---

htl bildung mit zukunft	College of Information Technology Ybbs/Donau
Department: Educational focus:	Information Technology Media Technology

Illustrative Graph, Photo (incl. explanation)	 <p>The BambiGuard Logo</p>
--	--

Participation in Competitions Awards	Bosch Technik fürs Leben Energy Globe Award Jugend Innovativ #35 Mostviertler Schul-Innovativpreis Sustainability Award FH OÖ Campus Wels 2022
---	--

Accessibility of Diploma Thesis	Archive of the school
------------------------------------	-----------------------

Approval (Date / Sign)	Examiner	Head of College / Department
---------------------------	----------	------------------------------

Inhaltsverzeichnis

Kurzfassung der Diplomarbeit/Abstract	C
Inhaltsverzeichnis	I
Danksagungen	II
1. Einleitung	1
1.1 Ausgangslage	1
1.2 Problemstellung	1
1.3 Persönlicher Standpunkt.	2
1.4 Individuelle Themenstellungen	3
1.5 Vorgehensweise	3
2. Grundlagen und Methoden	4
2.1 Etablierte Lösungsansätze	4
2.1.1 Absuchen des Mähfeldes	4
2.1.2 Vergrämung	4
2.1.3 Einsatz von Drohnen	4
2.2 Drohne	5
2.2.1 Vergleich verschiedener Drohnenmodelle	5
2.2.2 Rechtlicher Rahmen.	7
2.2.3 Tests des Modells DJI Mavic 2 Enterprise Dual	8
2.3 Mobile Applikationen	10
2.3.1 Flutter.	12
2.3.2 Xamarin.	13
2.3.3 React Native	13
2.3.4 Native Apps	14
2.3.5 Conclusio	15
2.4 Bilderkennung.	15
2.4.1 Bildverarbeitung- und erkennung mit OpenCV.	16
2.4.2 Bilderkennung durch maschinelles Lernen mit TensorFlow	22
2.4.3 Conclusio	25
2.5 User Interface Design	25
2.5.1 Unterschiede zwischen User Interface und User Experience	25
2.5.2 Grundprinzipien und Gesetzmäßigkeiten	27
2.6 Web-Frameworks	33
2.6.1 Vergleich verschiedener Web-Frameworks.	33

2.6.2 Conclusio	35
2.7 Gestaltgesetze	36
2.8 Werkzeuge und Wissen	39
2.8.1 Software.	39
2.8.2 Projektmanagement.	46
3. Ergebnisdokumentation	48
3.1 Einleitung	48
3.1.1 Was ist BambiGuard?	48
3.1.2 Themenabgrenzung	48
3.1.3 Entwicklungsprozesse	49
3.2 Software-Architektur	49
3.2.1 Anforderungen	49
3.2.2 Tech-Stack	49
3.2.3 Schnittstellen	53
3.3 App-Design	54
3.3.1 Modell	55
3.3.2 Wireframe.	59
3.3.3 Mockup und Prototyp	65
3.4 Bilderkennung.	70
3.4.1 Schritte des Algorithmus	70
3.4.2 Einrichtung der Entwicklungsumgebung	71
3.4.3 Programmierung	73
3.4.4 Photogrammetrie	75
3.4.5 Debugging und Testen	77
3.5 App für Android.	78
3.5.1 Entwicklungsumgebung Android Studio	78
3.5.2 Informationshierarchie	79
3.5.3 Benutzeroberfläche	80
3.5.4 Flugplanung mit Mapbox.	87
3.5.5 Debugging und Testen mit der Drohne	98
3.5.6 Arbeiten mit des DJI SDK	100
3.5.7 Bilderkennung und NDK.	108
3.5.8 Implementierung des Websockets	113
3.6 App für iOS	117
3.6.1 Entwicklungsumgebung Xcode	117

3.6.2 Benutzeroberfläche	118
3.6.3 Arbeiten mit dem DJI SDK.	122
3.7 Node.js Server.	124
3.7.1 Node.js	124
3.7.2 BambiGuard Implementierung	125
3.7.3 Distribution.	127
3.8 Repräsentation	129
3.8.1 Design der Marke BambiGuard.	129
3.8.2 Website	134
3.8.3 Werbevideo	146
3.8.4 Infopaket	147
3.8.5 Social Media	154
4. Evaluierung	156
4.1 Projektmanagement	156
4.2 Planung und Konzeption	156
4.3 Verbesserung des Systems BambiGuard	157
4.4 Repräsentation	157
5. Abbildungsverzeichnis	159
6. Tabellenverzeichnis	163
7. Programmausdruckverzeichnis	164
8. Anhang	167
8.1 Verfasserverzeichnis	167
8.2 Beigelegtes Speichermedium.	167
8.3 Projektstrukturplan	168
8.4 Meilensteine.	169
8.5 Projektphasen	170
8.6 GANTT-Diagramm.	171
8.7 Stundenlisten	172
9. Quellen und Literatur	174

Danksagungen

Wir wollen uns an dieser Stelle bei allen Menschen bedanken, die uns bei der Erstellung dieser Diplomarbeit unterstützt haben. Die Unterstützung und das konstruktive Feedback der Betreuer und Professoren waren nicht selbstverständlich und wir waren und sind für jeden Input dankbar.

Einen wichtigen Beitrag leisteten Frau Professor Rumetshofer und Herr Professor Meyerhofer, indem sie stets ein Verständnis für unsere Fragestellungen zeigten und uns mit Hilfestellung auf den richtigen Weg führten. Mit ihrem umfassenden fachlichen Wissen waren sie eine entscheidende Stütze für die Umsetzung der Arbeit.

Besonderer Dank kommt Frau Professor Rohrmüller zu, die uns immer wieder Zuversicht gegeben hat, auch in schwierigen Phasen nicht aufzugeben und unser Ziel konsequent weiterzuverfolgen. Viel Stress und Arbeit haben unsere Arbeit an der Diplomarbeit oft nicht einfach gemacht und ein aufheiternder Spruch war in diesen Zeiten oft der benötigte Motivationsstoß.

Ein großer Dank geht auch an jene Lehrkräfte, welche uns bei den Anmeldungen für verschiedene Wettbewerbe geholfen haben und uns auf diese aufmerksam machten.

Unseren Klassenkollegen danken wir für den Zusammenhalt und die Motivation. Der Teamgeist wurde dadurch sehr gestärkt und wir haben uns immer gegenseitig geholfen, um die Ziele gemeinsam erreichen zu können.

Zu guter Letzt sei auch Maximilian Strobls Vater gedankt, ohne welchem die Diplomarbeit so nicht stattfinden hätte können. Er hat uns die finanzielle Unterstützung gegeben und eine geeignete Drohne für unser Projekt zur Verfügung gestellt.

1. Einleitung

1.1 Ausgangslage

Für viele Wildtiere, darunter Rehkitze und Junghasen, bietet die Wiese im Frühjahr einen Rückzugsort und einen Schutz vor Fressfeinden. Durch das Mähen der Wiesen mit immer größer werdenden Mähwerken und immer höheren Geschwindigkeiten werden schätzungsweise jährlich bis zu 25.000 Jungtiere getötet¹. Allem voran ist das Problem, dass die frühen Erntetermine von Grünland in die Brut- und Setzzeit vieler Wildtiere fallen. Als wäre der Schaden an der Natur und deren Artenvielfalt nicht schon genug, werden dadurch auch die Nutztiere, welche durch tote Wildtiere mit Bakterien verunreinigtes Futter fressen, gefährdet. Die Nutztiere können durch das gebildete Nervengift (Botulinumtoxin) an schweren Krankheiten leiden und darüber hinaus sogar sterben.

Im schlimmsten Fall könnte einem Landwirt durch den Mähtod eines Rehkitzes nicht nur ein wirtschaftlicher und gesundheitlicher Schaden seiner Tiere entstehen, sondern auch eine strafrechtliche Verfolgung im Sinne des österreichischen Tierschutzgesetzes drohen.

1.2 Problemstellung

Mähtode von Rehkitzen verursachen immer wieder Diskussionen über Schuldzuweisungen dieser Vorfälle. Aufgrund dessen stellt sich die Frage, wer nun die Verantwortung eines solchen Vorfalls tatsächlich trägt, der Landwirt oder der Jäger?

Grundsätzlich liegt die Verschuldung eines Mähtodes beim Fahrzeugbetreiber des Mähwerks, welcher dafür zu sorgen hat, dass weder Personen- oder Sachschaden durch diesen Betrieb entsteht. Den Jäger trifft hierbei nur eine Mitwirkungspflicht (Hegepflicht – festgelegt im Landesjagdgesetz des jeweiligen Bundeslandes), das bedeutet, dass der Jäger dem Bauern die möglichen Mittel zur Verfügung stellen muss, welche der Vergrämung oder Vertreibung des Wildes dienen (Blinklicht, Plastiktüten, etc.) Verursacher und damit primärer Schadenserzeuger ist somit der Landwirt, der für das Absuchen seiner Ländereien verantwortlich ist.

1 Vgl. Tir 10.03.2022

Im österreichischen Tierschutzgesetz befinden sich zumindest drei Paragraphen, welche auf die Mähtode abzielen könnten²:

- § 5. (1) "Es ist verboten, einem Tier ungerechtfertigt Schmerzen, Leiden oder Schäden zuzufügen oder es in schwere Angst zu versetzen."
- § 6. (1) "Es ist verboten, Tiere ohne vernünftigen Grund zu töten."
- § 9. "Wer ein Tier erkennbar verletzt oder in Gefahr gebracht hat, hat, soweit ihm dies zumutbar ist, dem Tier die erforderliche Hilfe zu leisten oder, wenn das nicht möglich ist, eine solche Hilfeleistung zu veranlassen."

Wie ein österreichisches Gericht diesen Tatbestand in einen plausiblen Strafbestand interpretieren würde, muss sich erst bei gegebenem Anlassfall einer Anklage zeigen. Deutsche Gerichte sehen auf Grundlage des deutschen Tierschutzgesetzes sehr wohl eine Deckung zwischen Tat- und Strafbestand und verurteilten bereits einige Landwirte zu Geldstrafen bis zu mittleren vierstelligen Beträgen.³

1.3 Persönlicher Standpunkt

Unser Grundgedanke bei der Auswahl unseres Themas war immer von einer anwendbaren pragmatischen Idee geleitet. Für uns kam keine Themenstellung in Frage, welche nach der Fertigstellung unserer Diplomarbeit keinen Nutzen oder Erleichterung für jemanden bringt. Des Weiteren wollen wir unsere Fachkenntnisse im Bereich Corporate Design/Identity und Appentwicklung stärken.

Maximilian ist Jäger und ist mit dieser Idee auf unser Projektteam zugekommen. Anfangs war die Idee nur eine Arbeitserleichterung der Jäger und Erhöhung der Erfolgsquote beim Suchen von Rehkitzen, doch durch Beratung mit dem Projektteam wurde erweiternd auf eine Bewusstseinsbildung der Landwirte und Jäger durch Projektmarketing aufmerksam gemacht. Nach und nach wurde auch der Nutzungsbereich eingegrenzt, sodass der Landwirt im Endeffekt besseres und gesünderes Futtermittel für seine Tiere erreicht. Schlussendlich fügten sich unsere Interessensbereiche nahtlos in die Aufgabenstellung ein.

Da wir alle drei im ländlichen Raum wohnen, haben wir bereits einen Zugang zu dieser Thematik. Jeder von uns kennt in seiner Verwandschaft mindestens einen Landwirt, der bereits Probleme beim Mähen aufgrund von Kitzen hatte. Durch diesen Kontext, ist uns die Thematik des Problems nicht fern und wir wollen die derzeitige Situation verbessern.

Die Idee zur Verbesserung der Benutzerfreundlichkeit beim Absuchen eines Feldes mittels einer Drohne stammt von Maximilians Vater. Er besitzt keine ausgezeichneten Kenntnisse in der Drohnensteuerung. Daher ist es ihm derzeit nicht möglich, Felder abzusuchen. Mit diesem Problem ist er nicht alleine,

2 Vgl. Rec. 10.03.2022

3 Vgl. Wag 10.03.2022

viele Landwirte und Jäger würden von einem automatischen Abflug ihrer Felder profitieren. Das Ziel ist, die beteiligten Personen durch Automatisierung des Prozesses in Bezug auf die Flugsteuerung und das Erkennen der Wildtiere zu entlasten.

1.4 Individuelle Themenstellungen

Maximilian Strobl hat die Aufgabenstellung, die Front-end Konzeption und Entwicklung der Bambi-Guard App zu gestalten. Er ist verantwortlich für das Design, die Aufteilung und die Benutzererfahrung sowie die iOS-Version der Applikation. Weiters erstellt Maximilian die BambiGuard Website im Zuge der Repräsentation der Diplomarbeit.

Clemens Losbichler ist verantwortlich für die Programmierung des Algorithmus zur Bilderkennung und des Back-ends der Smartphone App. Außerdem beschäftigt er sich mit der Entwicklung der Android-Version der BambiGuard App.

Felix Teufel hat den Zuständigkeitsbereich Corporate Design und Corporate Identity. Dazu gehören Infomaterialien, das Logo, ein Werbevideo und Social-Media-Kanäle mit dem Ziel Bewusstsein und Aufmerksamkeit zu generieren.

1.5 Vorgehensweise

Im ersten Inhaltskapitel „2. Grundlagen und Methoden“ der vorliegenden Arbeit sind für die Realisierung der Aufgabenstellung relevante Technologien und Konzepte beschrieben. Es wird genauer auf die Recherche verschiedener Drohnenmodelle und deren Vergleich eingegangen. Außerdem beschäftigt sich dieser Teil mit der Auswahl der Methoden, mit welchen die App, die Bilderkennung und die Website programmiert wurden. Neben den technischen Vorgehensweisen geht dieses Kapitel noch auf diverse gestalterische Grundlagen ein.

Das Kapitel „3. Ergebnisdokumentation“ umfasst detaillierte Schilderungen des Arbeitsprozesses und der einzelnen Arbeitsschritte im Zuge der Erfüllung der Aufgabenstellung. In der Dokumentation angegebene Programmzeilen sind Auschnitte aus der praktischen Arbeit. Alle relevanten und referenzierten Dateien sind im abgegebenen Speichermedium zu finden.

Ein Verzeichnis darüber, welches Teammitglied welchen Abschnitt verfasst hat, ist im Anhang zu finden, in dem sich auch für das Projektmanagement relevante Dateien, Grafiken und Tabellen befinden. Die beigelegte DVD enthält neben den Projektergebnissen mit allen Programmdateien auch ausführlichere Dokumente der Planung und des Controllings. Der Inhalt und die Struktur der DVD sind im Anhang tabellarisch beschrieben.

2. Grundlagen und Methoden

2.1 Etablierte Lösungsansätze

Es existieren einige gängige Methoden, welche von Jägern, Landwirten und Helfern eingesetzt werden und mit denen der Mähtod zumindest teilweise vermieden werden kann.

2.1.1 Absuchen des Mähfeldes

Die einfachste und naheliegendste Methode ist das Feld oder die Wiese nach Tieren abzusuchen. Dabei geht man, am besten mit einigen Helfern, den Bereich systematisch ab und bringt alle gefundenen Rehkitze in Sicherheit, indem man sie neben dem Feld ablegt. Diese Methode ist nicht sehr effektiv und dauert lange. Außerdem sieht man unter dem Gras versteckte Tiere beim Vorbeigehen nur schlecht, weswegen häufig einige Tiere nicht entdeckt werden. Deswegen ist es bei dieser Methode ratsam, einen Spürhund mit sich zu führen, um die Tiere besser erkennen zu können. Ein weiteres Hilfsmittel, welches die Effektivität steigert, ist der Infrarot-Wildretter. Dieser ist ein ausziehbarer Stab, an welchem mehrere Infrarotsensoren befestigt sind. Wird ein Tier durch seine Wärmesignatur erkannt, so schlägt das Gerät Alarm. Hierbei ist es jedoch wichtig, das Feld in den frühen Morgenstunden abzusuchen, da Sonnenschein einen Fehlalarm hervorrufen kann.

2.1.2 Vergrämung

Vergrämung bedeutet, Tiere mittels Irritation zu verscheuchen. Verwendet werden laute **Geräusche**, künstliches **Licht** oder auch Anmähen der Wiese am Vorabend. Diese Aktionen müssen unmittelbar am Tag der Mahd oder am Vortag geschehen, da das Wild sich nach einiger Zeit an die Veränderung gewöhnt und wieder in den ursprünglichen Bereich zurückkehrt. Es können mit geringen Kosten und geringem Aufwand Rehkitze und teilweise Feldhasen vor dem Mähtod gerettet werden.

2.1.3 Einsatz von Drohnen

Der manuelle Drohnenflug mit optischer Kamera ist eine sehr effektive Methode zum Finden von Tieren. Man überfliegt das Feld und sucht die **Echtzeitaufnahme** der Kamera nach Auffälligkeiten ab. Zusätzlich kann eine Wärmebildkamera für effizienteres Suchen verwendet werden. Ebenso wie bei der Suche mit dem Infrarot-Wildretter ist es ratsam, früh morgens die Suche zu starten, da Kahlstellen oder Maulwurfshügel fälschlicherweise als Wildtiere erkannt werden könnten.

Der manuelle Drohnenflug ist demnach die effektivste und effizienteste Methode im Vergleich zu den anderen. Man braucht dafür eine Drohne, einen Piloten und einige Helfer, welche einerseits die Tiere aus dem Feld tragen und andererseits bei der Erkennung im der Rehkitze Kamerabild helfen.

2.2 Drohne

2.2.1 Vergleich verschiedener Drohnenmodelle

Der Markt des Verkaufs von Drohnen ist sehr jung und hat erst in den letzten 10-20 Jahren an Reichweite und Produkttiefe gewonnen. Beispielsweise wurde das Unternehmen DJI (*Da-Jiang Innovations Science and Technology Co., Ltd*), für dessen Drohnenmodell BambiGuard entwickelt wird, erst 2006 gegründet. Dadurch sind die Produktauswahl und Entscheidungsmöglichkeiten auf einen überschaubaren Rahmen eingeschränkt. Für die Diplomarbeit wurden einige der etablierten Marken und Modelle recherchiert und verglichen.

Parrot Anafi Thermal

Die Parrot Anafi Thermal Drohne verfügt über einen Infrarot Sensor, welcher Temperaturen im Bereich von -10°C bis $+400^{\circ}\text{C}$ unterscheiden kann und eine Auflösung von 160x120 Pixeln hat. Zusätzlich besitzt sie einen RGB Farbsensor mit einer Auflösung von 3840x2160 Pixeln. Es können beide Videosignale miteinander verrechnet und angezeigt werden, um somit ein deutlicheres Bild zu erhalten.

Das Kameramodul besitzt einen Einstellungswinkel von $+/ -90^{\circ}$ und einen dreifachen digitalen Zoom. Die maximale Entfernung zur Fernsteuerung darf 4km nicht überschreiten. Die Drohne wiegt 386 Gramm und ist im ausgeklappten Zustand 153x152x116mm groß.¹

DJI Matrice 300 RTK (mit Zenmuse H20T Sensor)

Die DJI Matrice 300 RTK Zenmuse H20T verfügt über drei Kameras und einen Laserentfernungsmaßstab. Verbaut ist eine Zoomkamera (Auflösung: 3480x2160 Pixel), eine Weitwinkelkamera (Auflösung: 1920x1080 Pixel) und eine Wärmebildkamera (Auflösung: 640x512 Pixel).

Das Kameramodul besitzt einen horizontalen Einstellungswinkel von $+/ -330^{\circ}$ und einen vertikalen Einstellungswinkel von $-132,5^{\circ}$ bis $+42,5^{\circ}$. Das Kameramodul verfügt zusätzlich auch noch über eine eigene Zoomkamera, welche optischen Zoom ermöglicht. Die Drohne wiegt 828 Gramm und ist 167x135x161 mm groß. Sie kann nicht zusammengeklappt werden.²

Autel Evo II Dual

Die Autel Evo II Dual verfügt über eine Wärmebildkamera (Auflösung: 640x512 Pixel) und eine optische Kamera (Auflösung: 5472x3076 Pixel). Die Normalbildkamera besitzt einen dreifachen optischen Zoom, dieser kann jedoch digital auf das 8-fache vergrößert werden.

1 Vgl. Par 20.02.2022

2 Vgl. DJI Zen 20.02.2022

Die Drohne besitzt eine Höchstgeschwindigkeit von 45 km/h und eine maximale Flugdauer von 38 Minuten, zusätzlich beträgt die maximale Datenübertragungsdistanz 9 km. Sie wiegt 1150 Gramm und besitzt eine Spannweite von 397mm, kann jedoch auch für den leichteren Transport zusammengeklappt werden.³

DJI Mavic 2 Enterprise Dual

Die Mavic II Enterprise Dual verfügt über eine Wärmebildkamera (Auflösung: 160x120 Pixel) und eine optische Kamera (Auflösung: 3840x2160 Pixel). Das Kameramodul kann sich horizontal um -100° bis +100° und vertikal um -135° bis +45° drehen. Die Videosignale können ebenfalls für ein detaillierteres Bild miteinander verrechnet werden.

Die Drohne besitzt eine maximale Flugdauer von 31 Minuten bei konstant 25 km/h. Die maximale Fluggeschwindigkeit beträgt 72 km/h. Sie wiegt 899 Gramm und ist im ausgeklappten Zustand ohne Zusatzmodule 322x242x84 mm groß.

An die Fernsteuerung, welche über ein Display zum Anzeigen von Echtzeitdaten, die Steuerungsknöpfe und Knöpfe verfügt, wird ein Smartphone mittels Kabel angeschlossen. Um die Drohne zu bedienen, lädt man am Smartphone die vom Hersteller entwickelte App namens „DJI Pilot“ aus dem Play Store (Android) oder aus dem App Store (iPhone) herunter. Mit dieser kann man entweder manuell fliegen oder Missionen (z.B. Wegpunkte oder Follow Me) planen und starten. Die Kommunikation zwischen Drohne und Fernsteuerung erfolgt über 2,4 GHz und 5,8 GHz; der beste Übertragungskanal wird automatisch gewählt.⁴

Conclusio

	Parrot Anafi Thermal	DJI Matrice 300 RTK (mit Zenmuse H20T Sensor)	DJI Mavic 2 Enterprise Dual	Autel EVO II Dual
Auflösung Wärmesensor (Pixel)	160x120	640x512	160x120	640x512 oder 32x256
Genauigkeit	± 5 °C	± 2 °C	± 5 °C	± 3 °C
Akkulaufzeit	35 min	43 min	31 min	38 min
Kosten	2 280 €	> 15 000 €	2 899 €	7 000 – 1 000 €

Tabelle 1: Datenvergleich verschiedener Drohnenmodelle

3 Vgl. Aut 20.02.2022

4 Vgl. Dji Mav 20.02.2022

Die Diplomarbeit verfügt über ein begrenztes Budget, weswegen die Modelle DJI Matrice 300 RTK und Autel EVO II Dual ausscheiden, da diese zu kostenintensiv wären. Es wird auch davon ausgegangen, dass Landwirte oder Vereine meist nicht genügend Geld besitzen, um sich ein so teures Modell leisten zu können.

Die Entscheidung fiel auf die DJI Mavic 2 Enterprise Dual, weil sie sich einerseits gleich wie die Parrot Anafi Thermal im Rahmen des Budgets befindet und über annähernd dieselben Spezifikationen verfügt. Sie besitzt jedoch ein umfangreiches SDK für Desktop (Windows) und Mobile (iOS, Android) mit Dokumentationen, Demo Projekten und Tutorials. Überdies haben Felix Teufel und Maximilian Strobl Erfahrungen mit Drohnen der Marke DJI in Schulprojekten und im privaten Gebrauch gemacht.

2.2.2 Rechtlicher Rahmen

Der rechtliche Rahmen für Drohnenflüge wird von der EASA (*European Union Aviation Safety Agency*) für den gesamten europäischen Raum festgelegt. Abhängig vom Fluggerät, also des Gewichts und der Ladung, und vom Einsatzort werden Drohnen (genauer „UAS“ – *Unmanned Aircraft System*, deutsch: Unbemanntes Luftfahrzeugsystem) unterschiedlich kategorisiert und klassifiziert. In Österreich sorgt die Luftfahrtbehörde *AustroControl* für Regulierungen unterhalb der EU-Richtlinien.

Limitierungen

Ausgehend von dem verwendeten Drohnenmodell DJI Mavic 2 Enterprise Dual (UAS-Klasse C1) und dem Einsatz auf Feldern und Wiesen, fällt man in die Kategorie „Open“, Unterkategorie „A3“. Es sind folgende Limitierungen für diese Einstufungen festgelegt:

- Mindestes 150 m Sicherheitsabstand zu unbeteiligten Personen, Siedlungsgebieten, Industrieanlagen, etc.
- Maximal 25 kg Abfluggewicht
- Maximal 120 m über dem Boden
- Permanenter direkter Sichtkontakt zum Fluggerät
- Kein Transport von Gefahrengut oder Menschen

Voraussetzungen

Nach dem Lernen über ein Online-Training, bestehend aus video- und textbasiertem Kursmaterial und dem positiven Absolvieren einer Online-Prüfung (60 min) wird ein Zertifikat von AustroControl ausgestellt. Dieses erhält man per Mail als PDF, dieses muss man bei Flügen vorweisen können. Eine Registrierungen des Piloten und der Drohne bei der zuständigen Luftfahrtbehörde sind notwendig (in Österreich: AustroControl). Aufgrund dieser Vorschriften haben alle Projektmitglieder im Zuge der Diplomarbeit den Drohnenführerschein (Nachweis zur Absolvierung des Online-Training und der Online-Prüfung) erworben.

Flugautomatisierung

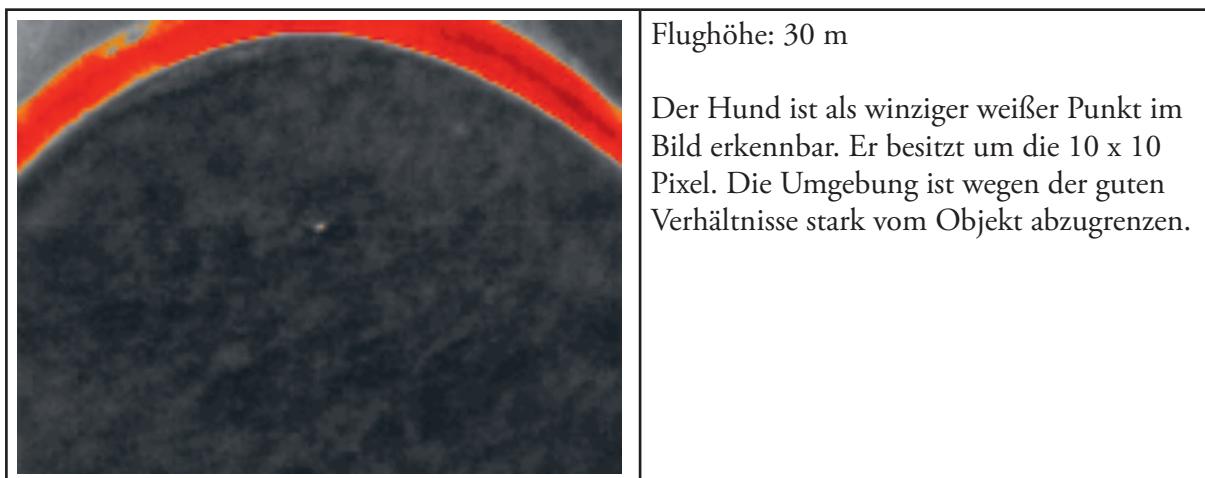
Abgesehen davon werden noch zwei Arten der Flugautomatisierung unterschieden. Im sogenannten „Autonomen Flug“ bewegt sich das Fluggerät von selbst und der Pilot ist nur Beobachter. Der „Automatische Flug“, welcher auf BambiGuard zutrifft, schreibt es vor, dass der Pilot jederzeit eine manuelle oder automatische Landung der Drohne erzwingen kann. Das bedeutet auch, dass eine ständige Aufmerksamkeit und Überwachung während des Flugs notwendig sind.

2.2.3 Tests des Modells DJI Mavic 2 Enterprise Dual

Um die Funktionsfähigkeit der Drohne und die Genauigkeit der Wärmebildkamera zu überprüfen, wurden Testflüge vorgenommen. Bei annähernd ähnlichen Verhältnissen und Wettersituation, wie sie dann auch im realen Einsatz herrschen, ist überprüft worden, wie gut Hunde in von der Drohne erstellten Aufnahmen zu erkennen sind. Die Tests beinhalten Fotoaufnahmen bei stehender Drohne mit verschiedenen Kalibrierungen und Flughöhen, sowie Videoaufnahmen mit unterschiedlichen Geschwindigkeiten. Einer dieser Tests wird exemplarisch im folgenden Abschnitt detailliert beschrieben.

Testergebnisse

Folgende Aufnahmen wurden bei 14°C und bewölktem Wetter mit leichten Schauern erstellt. Ein Hund ist in einem ca. 40 cm hohen Klee-Feld platziert und 10 m von der Straße entfernt. Es wurden mehrere Fotoaufnahmen in verschiedenen Flughöhen bei stehendem Flugkörper gemacht.



	Flughöhe: 14 m Die Konturen des Tieres sind erkennbar und es ist eine Temperatur zuzuordnen. Die Position kann auf ein viertel Meter genau bestimmt werden.
	Flughöhe: 5 m Kopf und Schwanz des Hundes sind zu sehen und differenzieren sich farblich von einander. Die Temperaturen können genau eingeschätzt werden.

Tabelle 2: Testergebnisse DJI Maivc 2 Enterprise Dual

Interpretation

Bei der Erstellung und Analyse der Aufnahmen sind folgende Faktoren als ausschlaggebend herausgestochen. Sie beeinflussen maßgeblich die Erfolgswahrscheinlichkeit, wenn man anhand des Wärmebildes ein Tier erkennen möchte.

Das **Kalibrieren** der Wärmebildkamera ist das Anpassen des Bildes der Kamera an die wirklichen Temperaturen. Die Steuerung der Drohne bietet mehrere Parameter zum Verändern der Rahmentemperaturen und der Ausgabefarbe. Um ein genaues Bild zu erhalten, soll vor dem Flug eine korrekte Kalibrierung vorgenommen werden, mit dem Ziel, die Tiere vom Untergrund unterscheiden zu können.

Die **Temperatur des Bodens** kann die Genauigkeit der Aufnahmen beeinflussen und die Erkennung der Tiere erschweren. Deswegen ist es ratsam, das Feld in den frühen Morgenstunden abzufliegen, da zu diesem Zeitpunkt meist noch ein höherer Temperaturunterschied zwischen Tier und Boden herrscht.

Die **Flughöhe** beeinflusst den Bereich, welcher von der Drohne erfasst werden kann. Mit einer höheren Flughöhe kann ein größerer Bereich abgedeckt werden, jedoch wird die Erkennung der Tiere dadurch ungenauer. Eine Flughöhe von 5-15 Metern ermöglicht eine genaue Erkennung der Tiere bei vernünftiger Abdeckung des Feldes.

Wärmebildsensoren nehmen typischerweise mit niedrigerer Frequenz als optische Sensoren auf. Deswegen muss speziell auf die **Fluggeschwindigkeit** geachtet werden. Bei einer Beispielhöhe von 11 Metern (horizontaler Bildwinkel: 57°) und der gegebenen Bildfrequenz der DJI Mavic 2 Enterprise Dual von 8,7 Hz kann man in der Theorie mit einer Geschwindigkeit von 30 km/h fliegen, wenn jeder Punkt des Feldes 10-mal am Video zu sehen sein soll. Dadurch entsteht genug Toleranz für Fehler seitens der Kamera, Verwackelungen oder menschliche oder software-technische Erkennungsfehler. Ein langsameres Fliegen erhöht die Genauigkeit und damit die Treffsicherheit beim Erkennen von Wildtieren. (Detailliertere Berechnungen und Modell finden sich in der Ergebnisdokumentation)

$$l = \tan(57^\circ) * 11 \text{ m} * \frac{360}{640} = 9,5 \text{ m}$$

$$v = 9,5 \text{ m} * 8,7 \text{ Bilder/s} * \frac{1}{10} * 3,6 = 29,8 \text{ km/h}$$

l Länge des aufgenommenen Bereiches
 v Geschwindigkeit

Direkte **Sonneneinstrahlung** auf den Boden oder das Feld können warme Flecken auf erhabenen Flächen, wie Maulwurfshügeln erzeugen. Im Wärmebild können diese in ähnlicher Intensität wie gesuchte Tiere zum Vorschein kommen, womit sie schwer voneinander zu unterscheiden sind.

2.3 Mobile Applikationen

Die Entwicklung einer App für mobile Endgeräte (englisch „*mobile application*“), vor allem in einem plattformübergreifenden Kontext, bezieht neueste Technologien zur Vollständigkeit der Planung mit ein. Es gibt einige sogenannte **App-Frameworks**, welche den Entwicklungsprozess von Apps für unterschiedliche Endgeräte beschleunigen sollen. Um diese Werkzeuge zu verstehen und zu evaluieren, müssen zuerst grundlegende Begriffe und Konzepte verstanden werden.

Native Apps

Native App Development nennt man die Entwicklung von Applikationen für mobile Endgeräte mithilfe der vom Hersteller bereitgestellten Technologien, also ohne Drittanbieter-Software (Genauer beschrieben in 2.3.4 Native Apps).

App-Frameworks

App-Frameworks sind Sammlungen von Werkzeugen und Komponenten zur Erleichterung der Programmierung von Mobile Applications. Der große Vorteil von Frameworks liegt in der Unabhängigkeit vom Betriebssystem des Endgerätes. Jede der folgend beschriebenen Tools (Flutter, Xamarin und React Native) kann zumindest für die Betriebssysteme Android und iOS kompilieren. Dadurch muss die Software nur einmal mithilfe des Frameworks programmiert werden, aber nicht für jede Plattform einzeln wie bei der nativen Entwicklung.

Die wichtigsten Faktoren für die Entscheidung, welche dieser Entwicklungsmethoden bei einem Projekt zu verwenden ist, werden hier beschrieben.⁵

Die **Entwicklungszeit** steigt enorm, wenn eine App für mehrere Systeme entwickelt werden soll. Verschiedene Anforderungen der unterschiedlichen Ziel-Plattformen machen den Prozess komplizierter. Außerdem bedingt die Entscheidung für eine Technologie, sei es die Verwendung eines bestimmten Frameworks oder das Programmieren in einer nativen Umgebung, auch das Ansammeln von Wissen über diese.

Eine höhere **Geschwindigkeit der App** führt zu einer besseren Benutzererfahrung. Entscheidend ist, wer die Software bedienen und wie sie eingesetzt werden soll, um den geforderten Grad an Geschwindigkeit oder flüssiger Bedienbarkeit herauszufinden. Generell ist feststellbar, dass Apps mit einer nativen UI die schnellste Erfahrung bieten.

Das **User Interface** soll so weit wie möglich in das Betriebssystem des Endgeräts integriert sein, um eine reibungslose Anwendung zu ermöglichen. Außerdem ist es in vielen Projekten erwünscht, eine Applikation auf allen Endgeräten gleich aussehen zu lassen. App-Frameworks geben meist ein bestimmtes Styling oder Benutzergefühl vor, welches meist auf allen Betriebssystemen ähnlich aussieht.

Ein einzigartiges Feature von App-Frameworks kann der **Hot Reload** sein. Dadurch kann man beim Programmieren Änderungen ohne Rekompilierung, am Gerät oder Simulator anzeigen lassen. Man spart sich eventuell lange Build-Zeiten und die Entwicklung ist schneller.

Bibliotheken oder Software Development Kits (SDK) von Drittanbietern erleichtern die Erstellung gewisser Programme enorm, weil sie die Wiederverwendbarkeit von bestehendem Source-Code ermöglichen. Oft ist eine SDK sogar zwingend notwendig für die Entwicklung der geplanten App. SDKs sind üblicherweise für die native Programmierung entwickelt und beispielsweise in zwei Versionen für Android und iOS verfügbar (Android SDK und iOS SDK separat). Frameworks bieten unterschiedliche Möglichkeiten eine SDK, also Plattform-spezifischen Code, einzubinden. Muss die App

⁵ Vgl. Bud 18.02.2022

stark an das Betriebssystem angepasst werden, stellt sich außerdem die Frage, ob sich ein Framework auszahlt oder der plattform-spezifische Teil der App überwiegt. Dadurch kann die Codebasis überladen und unnötig komplex werden.

2.3.1 Flutter

Das Framework Flutter ist von Google entwickelt und Open Source. Es kompiliert für mobile Geräte auf iOS und Android, das Web und Desktop-Geräte mit Windows, MacOS oder Linux. Es wird in der Programmiersprache Dart entwickelt, Flutter selbst ist in C++ geschrieben. Flutter unterstützt Hot Reload und Bildraten zwischen 60 und 120 Hz.

Dart

Die 2013 veröffentlichte Programmiersprache ist stark an JavaScript und C# angelehnt und wurde ursprünglich als moderne JavaScript-Alternative gesehen. Der Code wird mithilfe des Compilers **Dart2js** übersetzt, um von Browsern ausgeführt zu werden. Das Ziel von Dart ist eine schnelle und produktive Entwicklung von Apps für diverse Endgeräte.

Die Sprache ist objektorientiert und unterstützt sowohl einen typisierten Modus (Überprüfung von Typen, Exceptions, Assertions etc.), als auch einen Modus für die Produktion (englisch „*Deployment*“).⁶ Vorteile bringt Dart in der Lesbarkeit der Programme und der vergleichsweise flachen Lernkurve.

Widgets

Die Architektur einer Flutter-App besteht aus Widgets, welche Darstellung und Logik einer Komponente vereinen. Ein Widget kann aus anderen Widgets bestehen und so eine komplexe Applikation widerspiegeln. Flutter kompiliert Widgets nicht in native Komponenten, sondern implementiert sie selbst auf hardwarenaher Ebene mithilfe der Grafik-Bibliothek *Skia*⁷. Standardmäßig sind etliche Widgets inkludiert, wodurch die Erstellung von Apps, welche keine speziellen Komponenten fordern, stark erleichtert wird.

Der größte Nachteil des Frameworks liegt darin, dass es noch in der Entwicklung ist und ständig auf den neuesten Stand gebracht wird. Es ist noch nicht lange auf dem Markt, wodurch wenige Ressourcen zur Verfügung stehen.

⁶ Vgl. ECM 18.02.2022

⁷ Vgl. Flu 18.02.2022

2.3.2 Xamarin

Die von Microsoft entwickelte Plattform Xamarin ist genauso wie Flutter Open Source und legt den Schwerpunkt auf cross-platform Applikationen. Sie basiert auf der von demselben Hersteller stammenden Programmiersprache **C#** und dessen Umgebung **.NET**. Die wichtigsten Zielsysteme von Xamarin sind iOS, Android und Windows, wobei aber auch MacOS und das Web unterstützt werden.

Architektur

Xamarin verwendet native UI-Elemente des Betriebssystems, um seine Komponenten darzustellen. Das definierte User Interface wird bei der Kompilierung in das entsprechende Zielsystem nativ integriert. Dadurch kann eine Benutzererfahrung wie bei einer nativen App erzeugt werden. Die Kontinuität zwischen den verschiedenen Betriebssystemen ist jedoch durch die nativen Komponenten beschränkt.

Das Xamarin-Framework definiert sich dadurch, dass es die Geschäftslogik der Applikation für alle Zielplattformen teilt und darüber hinaus die nativen APIs in C# übersetzt und zugänglich macht. Aus diesem Grund muss der Programmcode der App auf die jeweiligen plattform-spezifischen Standards angepasst werden.

Dotnet

Das .NET-Ökosystem ist weit verbreitet in der Entwicklung von Windows-Anwendungen, Web- und Cloud-Systemen. Es bietet eine hohe Flexibilität und Konsistenz. Die Verwendung im Xamarin-Framework erweitert das Portfolio um mobile Applikationen, wodurch eine große Plattform mit unterschiedlichen Bibliotheken, Ressourcen und Hilfeleistungen mitgenutzt werden kann.

2.3.3 React Native

React Native ist genauso wie Flutter und Xamarin ein Open Source Framework mit Hauptaugenmerk auf Wiederverwendbarkeit. Es wurde 2015 von Facebook (heute Meta Platforms) veröffentlicht und basiert auf der Web-Technologie React. Der Programmcode wird in **JavaScript** geschrieben und unterstützt Android und iOS, sowie MacOS und Windows. Das Framework unterstützt Hot Reload und die Sprachen JavaScript und TypeScript.

Architektur

Applikationen werden wie in React in **Komponenten** strukturiert, welche die Erscheinung und Logik definieren. Je nach Zielplattform implementiert man unterschiedliche Darstellungen (Render-Funktionen), welche auf die nativen Komponenten der Betriebssysteme abgestimmt sind. React Native Apps werden somit, wie bei Xamarin, zu nativen Applikationen kompiliert.

Eine Applikation besteht vereinfacht aus einem UI Thread, welcher die Darstellung der App (Rendering) übernimmt, und einem JS Thread, welcher die Geschäftslogik beinhaltet. Die Bridge stellt die serialisierte Datenübertragung zwischen diesen Threads dar.

React-System

Die starke Ähnlichkeit mit dem verwandten Produkt React bringt eine große Community und ausgereifte Dokumentation mit sich. Die Programmierung von komplexen Applikationen fordert jedoch trotzdem Wissen über die einzelnen Zielplattformen und deren Eigenheiten. Es ist häufig notwendig individuelle Module zu erstellen, eine hohe Kenntnis über das Framework ist gefordert.

2.3.4 Native Apps

Die Entwicklung von Apps in der Weise, wie es der Hersteller des Betriebssystems vorschlägt, nennt man **Native Development**. Android Apps werden in Java oder Kotlin mit der Entwicklungsumgebung Android Studio (bzw. IntelliJ IDEA) programmiert, iOS Apps in Objective-C oder Swift mit der IDE XCode.

Kotlin

In der Programmierung von Android-Apps werden standardmäßig seit 2016 (stabiler Release von Kotlin) die Sprachen Java und Kotlin unterstützt. Letztere soll die Nutzung von Java seit 2019 komplett ersetzen und kann vollends in Java-Code übersetzt werden.⁸ Außerdem kann Kotlin-Code von Java aus aufgerufen werden und umgekehrt.

Kotlin bietet einen effizienteren Programmcode und eine damit gesteigerte Produktivität beim Entwickeln. Es soll übliche Fehler bei der Programmierung durch bessere Syntax vermeiden. Mittlerweile sind die Mehrheit der Ressourcen und Dokumentationen entweder in beiden Sprachen oder nur in Kotlin verfügbar.

Swift

Die von Apple für die Entwicklung von iOS, MacOS und anderen Systemen des Herstellers entwickelte Programmiersprache Swift ist eine Alternative zur herkömmlichen Programmierung mit Objective-C. Die Sprache fordert weniger Speicherkapazität und erlaubt eine effizientere Programmierung.⁹ Sie wird laufend von Apple weiterentwickelt und unterstützt.

⁸ Vgl. And 18.02.2022

⁹ Vgl. Swi 18.02.2022

2.3.5 Conclusio

Für das Projekt BambiGuard ist die Verwendung eines SDK zur Steuerung der Drohne nötig. Der im Abschnitt 2.2 beschriebene Hersteller DJI bietet zwei native Bibliotheken für iOS und Android an und stellt sowohl Dokumentationen als auch Beispiele zur Verfügung. Aus diesem Grund wurde die native Entwicklung der BambiGuard-App gewählt. Obwohl es einen deutlich höheren Aufwand bedeutet, ist die Methode angesichts der vom Projektteam gesammelten Erfahrung mit Android-Apps im Schulunterricht effizienter und weniger risikoreich. Die Wahl eines Frameworks wäre eine neue Domäne und im Spektrum des Projekts, welches sich durch den pionierartigen Einsatz, der im Abschnitt 1.4 erläuterten Bilderkennung auszeichnet, nicht notwendig. Außerdem fordert die Komplexität der App hardwarenahe Programmierung, welche besser durch plattform-spezifische Programmierung erreicht werden kann. Würde man dies mit einem Framework umsetzen, so würden große Teile des Sourcecodes für eine bestimmte Plattform gerichtet und nicht generalisiert für alle sein.

Für die Programmierung der Android-App wird die Sprache Java verwendet, da viele ältere Dokumentationen, wie die der DJI SDK, darin verfasst sind. Außerdem hat das Projekteam mehr Erfahrung in Java, weil diese Entwicklungsmethode in der Schule gelehrt wird.

Die BambiGuard-Anwendung für iOS soll in Swift geschrieben werden. Das bietet alle Vorteile der aktuellen Programmiersprache und den Einstieg in die „State of the Art“ Entwicklung von iOS-Apps.

2.4 Bilderkennung

Die Fähigkeit einer Software ein Objekt aus einem Bild zu erkennen und zu benennen, nennt man Bilderkennung (englisch „*Image Analysis*“ oder „*Image Recognition*“).¹⁰ Eine Anwendung versucht durch die Verarbeitung eines Bildes und die Erkennung von Schemen oder Mustern ein Objekt darin ausfindig zu machen. Der Begriff „Bild“ wird in diesem Kontext einer digitalen Rastergrafik (Pixelgrafik) gleichgesetzt.

Die Aufgabe aus einem Bild Informationen zu erhalten, dessen Teilgebiet die Bilderkennung ist, wird Computer-Vision (deutsch „*Computer-Sehen*“) genannt. Dahinter verbergen sich verschiedene Ansätze und Möglichkeiten Bilder zu verarbeiten, wie beispielsweise die Klassifizierung, Mustererkennung oder die umgekehrte Erstellung von Bildern aus gegebenen Schlüsselworten.

Eine wichtige Unterscheidung bei der Einteilung der Methoden zur Bilderkennung ist zwischen klassischer algorithmus-basierter Erkennung und dem Einsatz von künstlicher Intelligenz (KI). Endlösungen beinhalten oft eine Kombination der Verfahren. Das Bild wird zuerst verarbeitet und danach

10 Vgl. Dud 18.02.2022

mithilfe von neuronalen Netzwerken, einer Art von künstlicher Intelligenz gestützt durch maschinelles Lernen, klassifiziert, getaggt oder mit anderweitigen Informationen ausgestattet.

2.4.1 Bildverarbeitung- und erkennung mit OpenCV

Repräsentativ für „klassische“ Anwendungsfälle kann die Bibliothek OpenCV (Open Source Computer Vision) gesehen werden. Sie enthält Methoden und Algorithmen zur Verarbeitung eines Bildes, z.B. Farbraumkonvertierung, Schwellenwertbildung, Filterung oder geometrische Transformationen. Daneben beinhaltet OpenCV einige Funktionen zur Erkennung von definierten Objekten, wie Personen, Gesichter oder Autos.

Implementierungen

Die ursprünglich in C/C++ umgesetzte Bibliothek erschien im Jahr 2000 und wurde in den Sprachen Python und Java implementiert. Die gängige Entwicklung sieht die Erstellung von Prototypen in der höheren Sprache Python vor und eine Implementierung für die Endsysteme in C/C++, da dadurch eine höhere Performance gewährleistet werden kann. OpenCV kann in mobilen Applikationen in Android entweder als Java-Bibliothek oder mit dem Java Native Interface (JNI, beschrieben in der Ergebnisdokumentation) in der Version für C/C++ eingebunden werden. Apps in iOS bieten die Möglichkeit der Verwendung von Programmcode in C mithilfe einer Bridge (beschrieben in der Ergebnisdokumentation). Alle folgend vorkommenden Codezeilen und Definitionen sind mit der Version 4.5.5 der offiziellen OpenCV Dokumentation abgeglichen¹¹.

Herangehensweise

Angenommen man bekommt als Eingabe ein Bild, in dem man erkennen soll, ob sich ein bestimmtes Objekt darin befindet. Diese Angabe kann als Abstraktion der Anforderungen für BambiGuard gesehen werden.

Man kann das Vorgehen mit OpenCV für jene Problemstellung grob in drei Schritte unterteilen – das **Einlesen**, die **Verarbeitung** und die **Erkennung**¹². In den folgenden Absätzen werden wichtige Operationen erklärt und dargestellt. Die Bibliothek inkludiert eine deutlich größere Anzahl an Algorithmen und Funktionen als die hier beschriebenen, es wurden nur einige übliche und für die Problemstellung wichtige Verfahren ausgewählt.

11 Vgl. Ope 18.02.2022

12 Vgl. Mal 18.02.2022

Zuerst liest man das Material ein und bereitet es auf. Das Bild kann als Datei mit der Funktion `imread` oder aus einem Buffer mit `imdecode` eingelesen werden. Die Bibliothek unterstützt unter anderem die nachfolgend aufgelisteten Formate.

- Bitmap, *.bmp
- JPEG, *.jpeg, *.jpg
- PNG, *.png
- TIFF, *.tiff, *.tif
- WebP, *.webp

Danach verarbeitet man das Bild in der gewünschten Weise. Das kann beispielsweise eine der folgenden Methoden beinhalten.

Farbraum-Konvertierung

Eine Farbraum-Konvertierung wird eingesetzt, um das Material in das gewünschte Format zu bringen. Danach können spezifischere Operationen basierend auf dem gewünschten Farbmodell durchgeführt werden. Eine herkömmliche Konvertierung ist von RGB (Red, Green, Blue) bzw. RGBA (Red, Green, Blue, Alpha) zu Graustufen. Ein Bild in Graustufen verringert die Datenmenge für einen möglicherweise kritischen Verarbeitungsaufwand in späteren Schritten. In einem anderen Fall kann die Konvertierung in das Format HSV (Hue, Saturation, Value; auch „HSB“: Hue, Saturation, Brightness) zu einer leichteren Identifizierung von Farbe führen, da die Helligkeit- und Farbinformationen im Gegensatz zu RGB getrennt gespeichert werden.

Die Funktion `cvtColor` konvertiert eine Eingabematrix aus einem bestimmten Farbraum in ein anderes Format.

```
cvtColor(image, hsv_image, CV_BGR2HSV);13
```

Programmausdruck 1: OpenCV Funktion `cvtColor`

Mat image	Eingabe als Matrix in RGB/BGR
Mat hsv_image	Ausgabe als Matrix in HSV
int CV_BGR2HSV	Conversion Code; Konstante, welche die Konvertierung definiert

¹³ https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab 07.03.2022

Schwellenwert

Das Bilden und Anwenden eines Schwellenwerts (auch „Grenzwert“, englisch „*Threshold*“) reduziert ein zuvor in Graustufen konvertiertes Bild auf nur mehr zwei Pixelzustände, Schwarz und Weiß (man spricht von einem Binärbild). Es wird ein Wert der Graustufen-Skala gewählt (z.B. bei 7 Bit zwischen 0 und 255). Alle Pixel mit einem Wert unter diesem Schwellenwert werden auf 0 gesetzt, alle darüber auf den höchsten möglichen Wert (z.B. 255). Alternativ können auch Algorithmen zur Berechnung von variablen Schwellenwerten eingesetzt werden („*Adaptive Thresholding*“). Dadurch werden abweichende Helligkeiten in den unterschiedlichen Bildbereichen berücksichtigt. Die Implementierung in folgendem Programmcode verwendet beispielsweise den Mittelwert der Nachbarn eines Pixels als seinen Schwellenwert.¹⁴

```
thresholded_image = cv.adaptiveThreshold(image, 255, cv.ADAPTIVE_
THRESH_MEAN_C, cv.THRESH_BINARY, 11, 2);
```

Programmausdruck 2: OpenCV Funktion adaptiveThreshold

Mat image	Eingabe als Matrix in Graustufen
Mat thresholded_image	Ausgabe als Matrix im selben Format
int maxValue = 255	Maximaler Wert eines Pixels
int adaptiveMethod = cv.ADAPTIVE_THRESH_MEAN_C	Methode zum Ermitteln des Schwellenwerts
int threshold_type = cv.THRESH_BINARY	Art des Thresholding
int blockSize = 11	Größe der Nachbarschaft eines Pixels
double C = 2	Konstante, welche vom Mittelwert der Nachbarschaft abgezogen wird



Abbildung 1: Schwellenwert 128 bei einer 7-Bit Skala

14 https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ga72b913f352e4a1b1b397736707afcde3
07.03.2022

Morphologische Transformationen

Eine weitere gängige Methode zur Verarbeitung eines Bildes basiert auf der mathematischen Morphologie. Es handelt sich um ein von der theoretischen Mengenlehre und Topologie abgeleitetes Verfahren. Morphologische Transformationen verändern selektiv Bereiche des Bildes und lassen andere wiederum unverändert. Im Gegensatz zu ganzheitlich angewendeten Operationen nennt man dies **nicht-linear**¹⁵.

In den folgenden Absätzen werden die Operationen der morphologischen Transformation nur oberflächlich beschrieben. Eine mathematisch detaillierte Beschreibung kann in den Referenzen des Literaturverzeichnisses gefunden werden.

Der Grundansatz morphologischer Transformationen ist die Beschreibung des Bildes in Form von Mengen. Vereinfacht ist das bei Binärbildern vorstellbar: schwarze Pixel (Wert 0) bilden die Komplementärmenge der weißen Pixel (Wert 1).

Um die Form dieser Mengen zu analysieren, werden sogenannte Strukturelemente (englisch „*structuring elements*“) verwendet. Sie definieren ausschlaggebend die Anwendung der später erfolgenden Transformationen. Es werden alle Punkte einer Menge iteriert, das Strukturelement daraufgelegt und die Operation durchgeführt. Es besitzt einen Ankerpunkt, welcher dem aktuellen Punkt gleichzusetzen ist (in Abbildung 2 als Kreis gekennzeichnet). Verschiedenförmige Strukturelemente erzeugen unterschiedliche Ergebnisse der Transformationen. In Abbildung 2 sind zwei einfache Formen in unterschiedlichen Größen dargestellt.

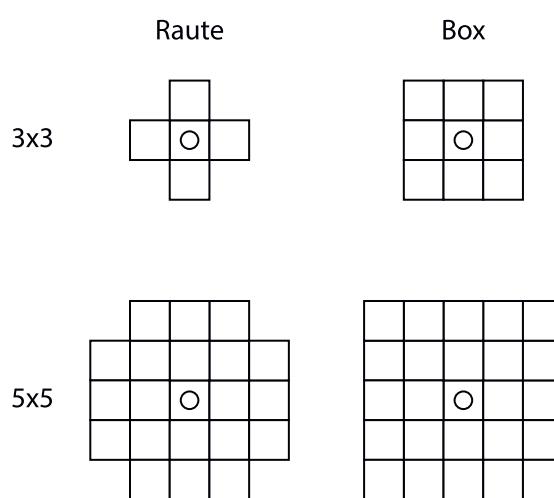


Abbildung 2: Strukturelemente

Die erste Grundoperation der Morphologie ist die **Erosion**. Sie ergibt die Menge der Punkte, auf welche das Strukturelement vollständig in die vom Bild definierte Menge passt. Die Bildmenge wird verkleinert. Abbildung 3 zeigt im rechten Raster die Ausgangsmengen und im linken Raster das Ergebnis der Erosion mit einem 3x3 Box-Strukturelement. Die dunklen Punkte stellen die resultierende Menge dar.

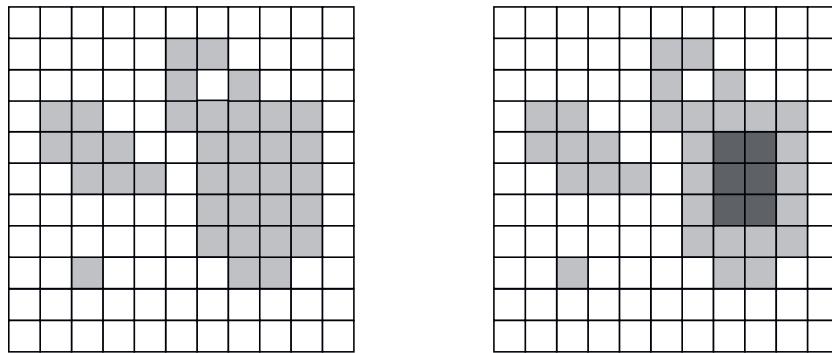


Abbildung 3: Erosion (Strukturelement 3x3 Box)

Die dazu umgekehrte Operation ist die **Dilation**. Sie beschreibt die Menge der Ankerpunkte, für die das Strukturelement die Punkte Bildmenge schneidet. Die Bildmenge wird vergrößert.

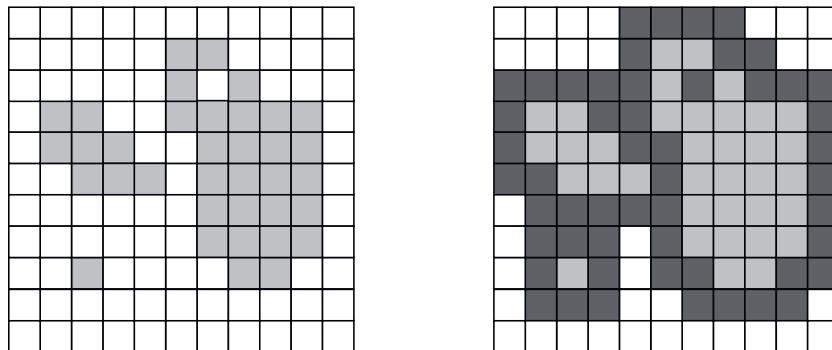


Abbildung 4: Dilation (Strukturelement 3x3 Box)

Die Kombinationen dieser zwei Operationen ergeben das morphologische **Öffnen** (zuerst Erosion und danach Dilation) und das morphologische **Schließen** (zuerst Dilation und danach Erosion)¹⁶. Die Abbildungen 5, 6 und 7 sind nachgebaut und vereinfacht, um die Vorgänge besser zu verdeutlichen. Sie sind als beschreibende Beispiele zu interpretieren.

16 Vgl. Ber 18.02.2022, S. 18

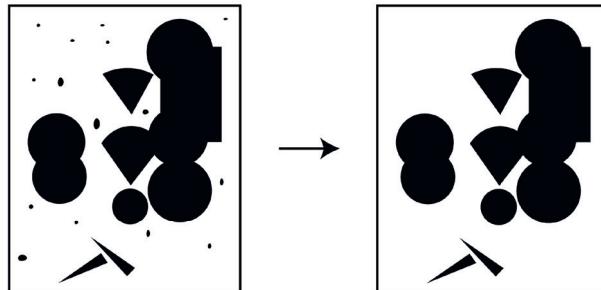


Abbildung 5: Morphologisches Öffnen

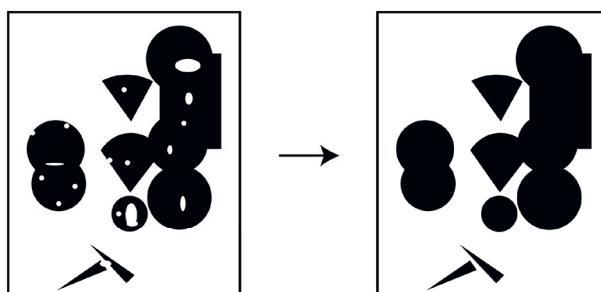


Abbildung 6: Morphologisches Schließen

In der Praxis werden Öffnen und Schließen (englisch „*opening*“ und „*closing*“) für folgende Ergebnisse angewendet.

- Entfernen von kleinen Objekten (Öffnen; wie in Abbildung 5)
 - Füllen von Löchern (Schließen; wie in Abbildung 6)
 - Isolieren von Objekten (Kombination)

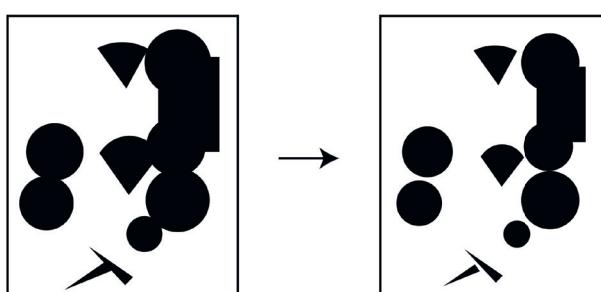


Abbildung 7: Isolation von Objekten durch eine Kombination der morphologischen Transformationen

Konturen eines Bildes lokalisieren

Ein in den vorherigen Schritten bearbeitetes Bild kann nun verwendet werden, um Objekte darin zu erkennen. Eine mögliche Methode dafür ist die Erkennung von Konturen und deren Position im Bild (englisch „*edge detection*“).

OpenCV implementiert einen Algorithmus, welcher die Konturen eines Binärbildes erkennt und diese als Liste von Punkten zurückgibt. Die Punkte der Kontur definieren somit die Position des Objektes, welches im Bild zu sehen ist.

```
findContours (image, contours, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE);17
```

Programmausdruck 3: OpenCV Funktion findContours

Mat image	Eingabe als Matrix in RGB/BGR
vector<vector <Point>> contours	Liste der Konturen, welche aus Listen von Punkten bestehen
vector<Vec4i> hierarchy	Conversion Code; Konstante, welche die Konvertierung definiert
int mode = RETR_TREE	Art der Kontur-Rückgabe und der hierarchischen Beziehung
int method = CHAIN_APPROX_SIMPLE	Methode der Kontur-Erkennung

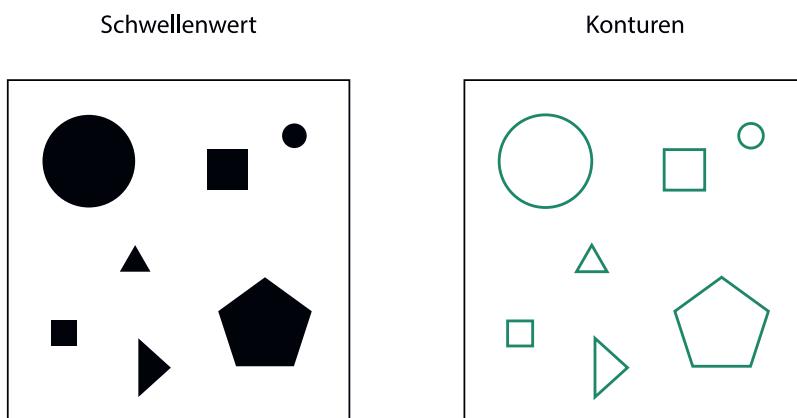


Abbildung 8: Konturen eines Binärbildes

2.4.2 Bilderkennung durch maschinelles Lernen mit TensorFlow

Maschinelles Lernen (englisch „*machine learning*“) ist ein Teilgebiet des Bereiches der künstlichen Intelligenzen (KI). Dabei versucht eine Software durch Beispiele oder schon gelöste Problemstellungen einen Erfahrungsschatz aufzubauen und zu abstrahieren, um daraus neue Datensätze zu lösen. Als „Lösung“ versteht man das Klassifizieren eines Datensatzes, die korrekte Identifizierung von Mustern oder ähnliche Einsätze der KI.

17 https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a
07.03.2022

Neuronale Netzwerke

Neuronale Netzwerke (NN; englisch „*neural networks*“) sind eine Lernvariante des maschinellen Lernens. Am Beispiel von biologischen Nervensystemen bilden künstliche neuronale Netzwerke Neuronen ab, welche in einer bestimmten Weise miteinander verbunden sind. Die Neuronen werden in Schichten angeordnet und sind mit den Neuronen der Nebenschichten aber nicht mit den Nachbarn derselben Schicht verbunden. In Abbildung 9 ist ein einfaches NN mit drei Eingabe-Neuronen (auch „*nodes*“) und einer Zwischenschicht zu sehen.

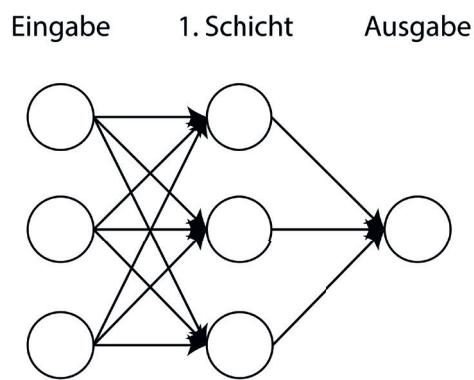


Abbildung 9: Einschichtiges neuronales Netzwerk

Convolutional Neural Networks

Die Modellierung, also die Anordnung der künstlichen Neuronen wird von der spezifischen Aufgabenstellung beeinflusst. Es gibt diverse vorgefertigte Modelle neuronaler Netzwerke, welche auf bestimmte Probleme abgestimmt sind. Für Anwendungen, die Bilder beinhalten wird üblicherweise die Klasse der Convolutional Neural Networks (CNN) verwendet. Diese zeichnen sich einerseits dadurch aus, dass die Eingabe als mehrdimensionale Matrix (die Dimension ist abhängig von den Farbkanälen der Pixelgrafik) erfolgt. Andererseits besteht ein CNN aus mehreren **Convolutional Layers**, gleichzusetzen mit den davor beschriebenen Schichten.

Ein Convolutional Layer besteht aus einem sogenannten Kernel, welcher vergleichbar mit Strukturlementen einer morphologischen Transformation ist. Er iteriert alle Bildpunkte und errechnet aus der jeweiligen Nachbarschaft den Wert des Neurons. In Abbildung 10 ist die Arbeit des Kernels veranschaulicht. Nach einem Convolutional Layer folgen noch **Pooling Layer** (komprimieren der Neuronen durch das Bilden von lokalen Maxima oder Mittelwerten; „*max pooling*“ oder „*mean pooling*“) und **Fully-connected Layer** (alle Neuronen der Eingangsschicht sind mit der Ausgangsschicht verbunden).

Derartige Netzwerke schaffen eine niedrige Fehlerquote beim Einsatz mit Bildern. Sie werden vor allem bei der Klassifizierung eingesetzt.

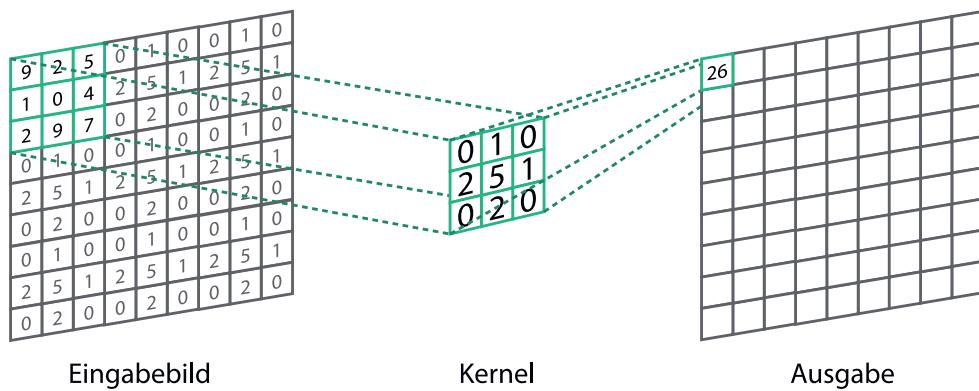


Abbildung 10: Convolution eines CNN (Die Werte der Eingabe, sowie des Kernels in diesem Beispiel sind beliebig gewählt)

Training eines neuronalen Netzwerks

Ein neuronales Netzwerk „lernt“, indem es die Werte (Gewichtungen) einzelner Neuronen anpasst. Vereinfacht gesagt, gibt man dem Netzwerk schon gelöste Beispieldaten als Eingabe und überprüft die Ausgabe die das NN ausgibt. Dem Ergebnis entsprechend verändert man nun die Werte der Neuronen in einer bestimmten Weise und nähert sich so immer weiter den tatsächlichen Lösungen der Probleme an. Diesen Ansatz nennt man **überwachtes Lernen**.

TensorFlow

Für die Anwendung von maschinellem Lernen gibt es verschiedene Frameworks für unterschiedliche Programmiersprachen. TensorFlow ist eines der populärsten und umfassendsten Systeme, welches 2015 von Google veröffentlicht wurde. Das Framework unterstützt die Sprachen Python, C++, Java, JavaScript, Swift und einige mehr. Die Version **TensorFlow Lite** ist speziell für mobile Endgeräte gemacht, um das Ausführen von Modellen auf leistungsschwächerer Hardware zu ermöglichen. Sie ist geeignet für die Anwendung von maschinellem Lernen auf Smartphones mit den Betriebssystemen Android oder iOS. TensorFlow nutzt die freie Bibliothek **Keras** für das Erstellen von Modellen der neuronalen Netzwerke. Der Workflow für den Einsatz von TensorFlow Lite auf mobilen Endgeräten besteht aus zwei Schritten: Zuerst erstellt man ein TensorFlow Lite-Modell – es wird entweder ein trainiertes Modell verwendet oder ein neues Modell angefertigt. Danach führt man es am Endgerät mit den entsprechenden Eingabedaten mithilfe des TensorFlow-Interpreters aus.

2.4.3 Conclusio

Im Vergleich ist der notwendige Entwicklungsaufwand bei der Erstellung eines Algorithmus mit klassischer Bilderkennung höher als der Einsatz eines Modells des maschinellen Lernens – vorausgesetzt man hat Zugang zu einem trainierten Modell, welches auf die Aufgabenstellung des Problems passt. Ist das nicht der Fall, müssen einerseits Trainingsdaten gesammelt werden und andererseits ein Modell mit diesen Daten trainiert werden, wofür ausreichend Rechenleistung bereitgestellt werden muss. Der Aufwand des Sammelns der Trainingsdaten ist schwer einzuschätzen, da es möglich ist, Online-Ressourcen für diesen Zweck zu verwenden, wohlgleich erst eine ausgiebige Planung und Recherche preisgibt, welche Eigenschaften die geforderten Daten benötigen. Bleibt nur mehr die Möglichkeit Trainingsdaten selbst aufzunehmen, so ist mit deutlich höherem Aufwand zu rechnen.

Abgesehen von der anfänglichen Unvorhersehbarkeit, kann die Anwendung von künstlicher Intelligenz eine niedrige Fehlerquote garantieren. Die klassische Bilderkennung hängt in diesem Punkt vollkommen von ihrer Konzeption und den angewandten Operationen ab.

In diesem Projekt entscheidend war die vom Projektteam zuvor gesammelte Erfahrung mit der Bibliothek OpenCV und dessen Handhabung. Der Einsatz in BambiGuard stellt zwar eine starke Vertiefung in das Thema dar, es existierte jedoch schon davor eine Grundbasis an Verständnis. Deswegen entschied man sich für die Bilderkennung mit OpenCV.

2.5 User Interface Design

2.5.1 Unterschiede zwischen User Interface und User Experience

User Interface

Das User Interface (UI, deutsch „*Benutzeroberfläche*“) ist die Schnittstelle zwischen dem Benutzer und der Steuerung der Anwendung. Anzeigeflächen wie Buttons, Eingabefelder und Listen etc. sind Teil des User Interface. Ein User Interface kann vielfältig gestaltet werden. Es wird zwischen grafischen Oberflächen, Eingabefenstern für Computerbefehle, Touchscreens oder Sprachausgaben unterschieden. Die Gestaltung der Oberfläche ändert sich mit den Anforderungen, beispielsweise benötigen Touchscreens größere Steuerungselemente wie Desktop-Anwendungen, bei denen mit dem Mauszeiger präzise ausgewählt werden kann.

User Experience

Die User Experience (UX) beschreibt das Nutzererlebnis, sowie die Nutzererfahrung bei der Bedienung des Systems. Der Ausdruck „User Experience“ kann als Sammelbegriff für alle Nutzerinteraktionen zwischen dem Nutzer und der Oberfläche verwendet werden. User Experience umfasst alle Reaktionen, sowie Emotionen und Gedanken, die während oder nach der Verwendung der Anwendung erfasst werden. Der UX-Experte Peter Morville beschreibt die User Experience anhand einer Honigwabe, bei der Eigenschaften gelistet werden, die ein gutes User Experience-Konzept haben sollte.

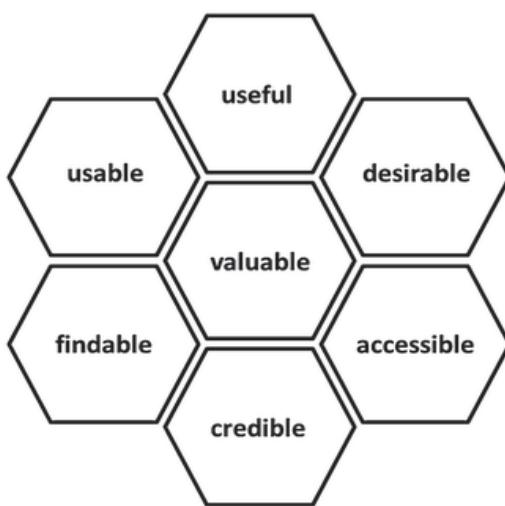


Abbildung 11: Abbildung 11: UI/UX Honigwabe nach Morville

- **Nützlich** (useful): Oberflächen werden nur genutzt, wenn sie hilfreiche Funktionen oder gut aufbereitete Inhalte bieten.
- **Nutzbar** (usable): Die Benutzerfreundlichkeit und Bedienbarkeit eines Interfaces sollte oberste Priorität haben.
- **Auffindbar** (findable): Die besten Inhalte sind zwecklos, können sie nicht einfach vom User gefunden werden.
- **Glaubwürdig** (credible): Die Inhalte werden passend zum Design angezeigt und wecken Vertrauen beim Benutzer.
- **Zugänglich** (accessible): Rücksichtsvolle Aufbereitung und bedürfnisstillende Anpassungen an den Benutzer.
- **Begehrswert** (desirable): Ansprechendes Design, sowie branchentypisches Layout weckt Emotionen beim User.
- **Wertvoll** (valuable): Fügen sich alle Eigenschaften passend zu einem Gesamtbild, erreicht die Oberfläche für den Benutzer einen Wert und wird weitere Zugriffe sowie positives Feedback zur Entwicklung erzielen.

Usability

Eng mit den beiden Ausdrücken User Interface und User Experience verknüpft ist der Begriff Usability. Hierbei werden die Benutzerfreundlichkeit, die Bedienbarkeit und die Gebrauchstauglichkeit zusammengefasst. Der Begriff setzt sich aus den englischen Wörtern „*to use*“ (benutzen) und „*ability*“ (Fähigkeit oder Geschick) zusammen und wird unter der DIN-Norm EN ISO 9241 definiert. Usability richtet sich in ihrer Anwendung an ein inhomogenes Anwenderfeld, so muss bei der Konzeptionierung sowohl auf Kinder als auch auf ältere Menschen geachtet werden. Ebenfalls nicht zu vernachlässigen ist, dass die Nutzung nie ohne Umwelt- sowie Situationseinflüsse passiert. Anders gesagt, ein Mensch reagiert in einer Gefahrensituation anders als in einer Gewohnheitssituation.

Nach einer erfolgreichen Konzeptionierung der Benutzeroberfläche folgt das Testen durch den User. Häufig klafft hierbei eine Spalte zwischen Benutzer und Betreiber bzw. Entwickler auf. Je früher hierbei Probleme gefunden werden können, umso schneller und besser entsteht das Endprodukt. Nachfolgend werden einige Verfahren zum Testen der Benutzeroberfläche aufgelistet:

- **Usability-Untersuchungen im Labor:** Diese Methode ist eine der genauesten und teuersten Methoden. Bei dieser Test-Variante werden beispielsweise die Augenbewegungen verfolgt, die Gedanken verbal mitgeteilt oder die emotionalen Einstellungen erkundet und festgehalten.
- **Remote-Usability-Studie:** Im Gegensatz zu den Laboruntersuchungen ist diese Art kostengünstiger, jedoch weniger angepasst an die eigene Anwendung. Es nehmen freiwillig Teilnehmer daran teil, die sich über eine Webanwendung anmelden können und so als Dienstleister auftreten.
- **Umfragen und Interviews:** Während andere Verfahren auf dem Nutzerverhalten beruhen, erhält man bei Umfragen und Interviews die Meinung der Nutzer.
- **Erfahrungsaustausch mit Experten:** Kernfundament dieses Usability-Reviews bilden UX-Expertenmeinungen. Elementar wird die Benutzeroberfläche nach vorgefertigten Checkliste überprüft, die Checkliste bildet sich aus häufigen Problemen der Vergangenheit von unterschiedlichen Erfahrungsprojekten.
- **A/B-Tests:** Eine direkte Vergleichsmethode in der Parallelentwicklung bieten A/B-Tests. Es werden anhand von zwei Entwicklungen Teile oder das Gesamtkonzept zum finalen Ergebnis gekürzt.

2.5.2 Grundprinzipien und Gesetzmäßigkeiten

In der Literatur findet man einige Grundprinzipien und Grundansätze zum UI/UX-Design. Nachfolgend werden relevante Gesetzmäßigkeiten näher beschrieben und mit je einem Beispiel deren Anwendung veranschaulicht.

Jakobs Gesetz

Das Jakobs Gesetz besagt, dass der Mensch nach grundlegenden ihm bekannten Abfolgen agiert. Nutzer erwarten von einem neuen Produkt, dass Grundsteuerungen oder Funktionen von vertrauten Produkten angewendet werden können.

Fitt's Gesetz

Eines der elementarsten Grundprinzipien bildet Fitt's Gesetz. Die Gesetzmäßigkeit stellt fest, dass Bedienfelder in optimaler Größe dargestellt, sowie in leichter Erreichbarkeit zur Bedieneingabe situierter werden sollen. Besonders wichtig sei dies bei Touch-Targets (Auswählbare Felder für den Bediendialog), um eine fälschliche Eingabe oder eine ungewollte Auswahl zu Vermeiden. Hersteller geben für Ihre Geräte sogenannte Mindestbemessungen an (Apple: 44 x 44 Punkte; Google: 48 x 48 Didot Punkte).

Beispiel aus der Praxis

Apple-Geräte gelten als einer der am besten abgestimmten Software- und Hardware-Konzepte. Beispielhaft ist das sehr weite Raster der App-Icons, wodurch jedes Element viel Platz erhält, selbst eigens hinzugefügte Widgets. Eine weitere Vorzeigefunktion ist der Einhandmodus, bei dem Bedienelemente der oberen Bildschirmfläche in komfortable Fingerreichweite gebracht werden.



Abbildung 12: UI/UX Prinzipbeispiel iPhone Einhandmodus

Hicks' Gesetz

Weniger Auswahl, schnellere Reaktionen und höhere Übersichtlichkeit, das ist die Devise von Hicks' Gesetz. Durch die Verringerung der Auswahlmöglichkeiten kann eine höhere Reaktionsgeschwindigkeit erreicht werden. Darüber hinaus kann durch die Zerteilung von komplexen Themen eine einfache und logische Struktur geschaffen werden.

Beispiel aus der Praxis

Früher waren unzählige Knöpfe auf der **Fernbedienung** (linkes Bild), heute finden sich nur einige wenige Knöpfe (rechtes Bild). Vieles ist in grafische Menüpunkte auf den Bildschirm gewandert.



Abbildung 13: UI/UX Prinzipbeispiel Gegenüberstellung alte/neue TV Bedienungen

Höchststand-Ende-Regel

Unter der Höchststand-Ende-Regel versteht man den Fortschritt oder Fertigstellungsgrad einer Aktion für den Benutzer sichtbar zu machen. Durch einen Leitfaden in Form von Stationen oder einer Fortschrittsanzeige erzeugt man ein Prozessdenken und eine Planbarkeit des Zeitaufwands. Man spricht auch von Journey Maps, die den Weg eines Prozesses ermitteln und anzeigen.

Beispiel aus der Praxis

Als best-practice Beispiel gilt der Newsletter-Service **MailChimp**. Durch die Anleitung mit einer Fortschrittsanzeige erhält der Benutzer ein Feedback über seine durchgeführte Aktion sowie den Stand seiner Aktivität. Im Screenshot kann man erkennen, dass erst einer von drei Schritten erfüllt ist, zusätzlich wird in der Mitte der nächste Schritt beschrieben.

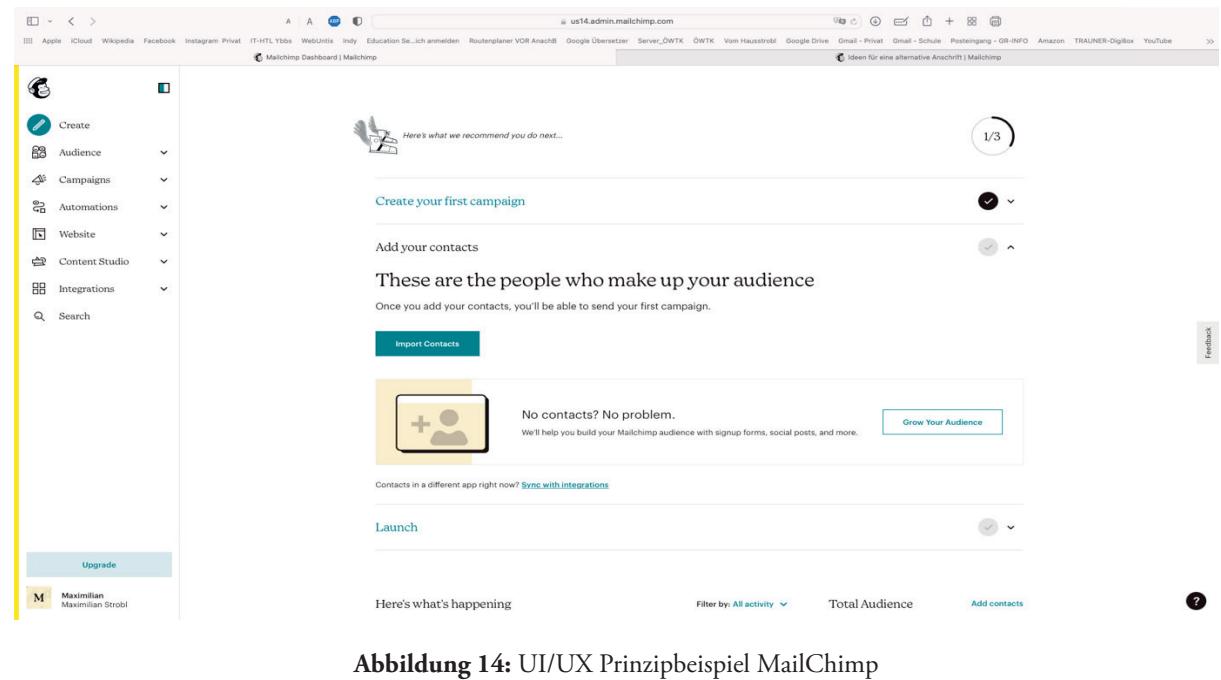


Abbildung 14: UI/UX Prinzipbeispiel MailChimp

Aesthetic-Usability-Effekt

Ohne benutzerfreundliches und ansprechendes Design ist die beste UI/UX-Entwicklung zum Scheitern verurteilt, das besagt der Aesthetic-Usability-Effekt. Kleine Usability-Probleme werden bei einem Design eher toleriert, wenn dem Nutzer das Design zusagt. Durch die korrekte Gestaltung können in der Entwicklungsphase einige Usability-Probleme kaschiert werden, jedoch liegt es in der Psyche der Nutzer, sich schlechte Momente eher zu merken als positive Aspekte.

Beispiel aus der Praxis

Eine Untersuchung zu Usability-Tests, durchgeführt von Andreas Sonderegger und Jürgen Sauer hat ergeben, dass bei gleichen Funktionen und Tastenanordnungen Benutzer die Anwendung des ersten Prototyps besser empfanden (Abbildung 16 rechts), als die des zweiten (Abbildung 16 links). Das liegt daran, dass den Nutzern die Informationsverarbeitung des ersten Prototyps aufgrund der Harmonie der Farben besser gefiel. Sie assozierten den positiven Eindruck des Designs auch auf dessen Funktionalität.



Fig. 1. Two prototypes employed in experiment: (a) aesthetically appealing design; (b) aesthetically unappealing design.

Abbildung 15: UI/UX Prinzipbeispiel Prototyp

Informationarchitektur

Bei der Informationsarchitektur beschäftigt man sich mit der Frage, wie man Inhaltskomponenten strukturieren kann. Die Strukturierung der Inhalte sollte im Einklang mit dem Kontext, dem Inhalt selbst und dem Nutzer sein. Etabliert haben sich in diesem Bereich folgende fünf Ansätze.

Hierarchische Strukturen

Hierarchische Strukturen werden auch als Baumstrukturen oder Eltern-Kind Konzept bezeichnet. Die Informationen werden in sich ausschließenden Kategorien geordnet. Die hierarchische Struktur ist die am häufigsten verwendete Form der Informationsarchitektur.

Sequentielle oder lineare Strukturen

Informationen werden in einer schrittweisen Abfolge strukturiert, dem Nutzer werden Entscheidungen der Navigierung abgenommen. Diese Form der Strukturierung erfolgt bei fest definierten Navigationsabfolgen, beispielsweise bei Tutorials, Bestellvorgängen oder Ausbildungsprogrammen.

Netzstrukturen

Die flexible Netzstruktur ist durch keine kontrollierte Abfolge gekennzeichnet. Die Auswahl der Verknüpfung von Informationen basiert vielmehr auf inhaltlich sinnvollen Kriterien. Bei ungenauem Aufbau kann diese Struktur zu Verwirrung oder Unüberschaubarkeit führen.

Metadaten

Unter der Metadaten-Struktur versteht man die Filterung der Informationen durch einheitliche Informationsmerkmale, wie etwa durch Schlagwortsuche, Größen oder Farben. Durch ihre Matrix-Anordnung bietet sie als einzige Struktur Querverweise.

Soziale Struktur

Eine der modernsten Informationsarchitekturen sind soziale Strukturen, bei welchen Informationen aufgrund von Interaction (Kommentieren des Haupteintrags), Reaktion (Bewerten des Haupteintrags, wie zum Beispiel „*Liken*“) oder Shares (Haupteintrag auf die eigene Seite hinzufügen) sortiert werden und einem anderen User zuerst gezeigt werden. Diese Form der Informationsaufbereitung lehnt sich sehr stark an soziale Medien an und kann vielfältig als Gesamtkonzept oder als Teilkonzept verwendet werden.¹⁸

2.6 Web-Frameworks

2.6.1 Vergleich verschiedener Web-Frameworks

Was ist ein Web-Framework?

Ein Web-Framework (englisch „*Rahmen*“) ist ein Programmiergerüst oder Werkzeug zur Vereinfachung und Beschleunigung der Entwicklung von Webanwendungen. In der Programmierung und Softwareentwicklung gibt es für mehrmals gebrauchte Standardfunktionen vorgegebene Lösungswege in Form von vorgefertigten Programmzeilen. Diese bilden eine einheitliche Vorgehensweise für bekannte Probleme und vereinfachen somit auch die Verständnisfrage beim Programmablauf. Hierbei werden unter anderem Werkzeuge für Datenbankzugriff, das Session-Management oder das Arbeiten mit Vorlagen (englisch „*Templating*“) bereitgestellt. Diese Sammlungen an Grundfunktionen sind in unterschiedlichen Programmen wiederverwendbar, setzen sich zu einheitlichen Standards zusammen und unterstützen den Web-Entwickler bei der sauberen und sicheren Entwicklung. Die einzelnen Frameworks bieten Demoprojekte und Dokumentationen an, wodurch es dem Entwickler erleichtert wird bei Problemen Hilfestellungen zu finden. Ein Web-Framework bietet außerdem eine einheitliche Darstellung in verschiedenen Browsern, sowie die Erleichterungen eines responsiven Designs (Anzeigeformat ist variabel).

Grundsätzlich kann zwischen JavaScript (JS)- und Cascading Stylesheet (CSS)- Frameworks unterschieden werden, je nach Framework spannen sich unterschiedliche Einsatzzwecke auf. Bei JavaScript-Frameworks können Komponenten beziehungsweise Inhalte dynamisch nach Kommunikation mit dem Server aufgebaut werden, da Komponenten eigenständig in das dynamische Document Object Model (DOM) eingehängt werden. Ebenso bieten JS-Frameworks beliebige Möglichkeiten zur Datenerhaltung sowie Datenmanipulation. JavaScript ist die Sprache des Webs, dadurch bietet das JS-Framework meist Funktionen, welche über das Styling hinaus reichen. CSS-Frameworks sind vom Umfang der Dateien schlanker und besitzen einen schnelleren Aufbau im Browser. Sie sind einfacher zu bedienen und die Einarbeitungszeit ist relativ zum JS-Framework verkürzt. Am besten sind die CSS-Frameworks daher für klassische Multi-Page Anwendungen geeignet.

Nachfolgend werden die drei größten JS-Frameworks gegenübergestellt.

AngularJS

AngularJS ist eines der beliebtesten JS-Frameworks. Es wird von Google entwickelt und ist eine Cross-Platform (Mobil- und Desktopanwendungen) zur Entwicklung von Single Page Applications (Einselseiten-Webanwendungen, die aus einem einzigen HTML-Dokument bestehen und Inhalte dynamisch nachladen). Als Design-Pattern wird das MVC-Modell verwendet, Welches eine Trennung von

Darstellung und Logik in der Programmierung vorsieht. AngularJS ist im Gegensatz zu den anderen Frameworks schwieriger zu erlernen und wird client-seitig von JavaScript gerendert, daher ist es nicht SEO-optimiert, wie andere Frameworks. Meist müssen zusätzliche Plug-Ins hinzugefügt werden, um eine SEO-Einschränkung zu überwinden. Unter der Abkürzung SEO („Search Engine Optimization“; deutsch „Suchmaschineneoptimierung“) versteht man die eigenen Inhalte an bestmöglichster Position in den Suchanfragen erscheinen zu lassen.

ReactJS

ReactJS wird von Meta Platforms (früher Facebook) entwickelt und wird häufig zum Erstellen von responsiven Apps verwendet. Anders als AngularJS ist React.js sehr SEO-freundlich durch die hohe Seitenladeleistung und das virtuelle DOM. Das Framework basiert auf einem virtuellen DOM und ist komponentenunabhängig. Dies bedeutet, dass aufgrund von geänderten Properties das DOM unabhängig der programmierten Komponente neu aufgebaut wird. Die Komponenten werden in eigenen Klassen geschrieben und können im ganzen Projekt wiederverwendet werden. Daten werden erst bei Änderungen neu gerendert und sofort angezeigt. Durch die Erweiterung der JavaScript-Syntax **JSX** wird es ermöglicht, HTML-ähnliche Strukturen zu verwenden. Anders als andere Frameworks ist ReactJS eine JavaScript Bibliothek statt eines eigenen Frameworks mit Rahmenstruktur, daher ist es eines der am leichtesten zu lernenden Frameworks.

VueJS

VueJS wurde zum Erstellen von Single-Page-Webanwendungen entwickelt und baut auf dem Design-Pattern **Model View Viewmodel** (MVVM) auf. Ebenso wie React.js nutzt das Framework ein virtuelles DOM. Eine Besonderheit ist eine mitgelieferte CSS-Komponente, die es erleichtert Übergänge und Animationen umzusetzen. Vue.js ist ebenso ein progressives Framework, wodurch die Adaption in vorhandene Projekte erleichtert wird. Ebenso von Vorteil ist die bidirektionale Datenbindung, wodurch Daten nicht über das HTML-DOM direkt ermittelt werden, sondern über einen Event-Handler. Das Framework ist nicht für umfangreiche Skalierungen geeignet und hat aufgrund der kurzen Existenz nicht so viele Plug-Ins und Tools wie ReactJS oder AngularJS.

Vor- und Nachteile von Frameworks

Frameworks erringen immer mehr Reichweite und Beliebtheit in der Webentwicklung. Einer der Vorteile, die bereits angesprochen wurden, ist die Wiederverwendbarkeit von Code-Elementen und Komponenten, wodurch eine schnellere Entwicklung und eine deutliche Reduktion des Zeitaufwandes gewährleistet wird. Ein Framework bietet nicht nur eine schnellere, sondern auch eine sauberere und sicherere Entwicklung. Die meisten Frameworks verfolgen zumindest das Modell-View-Controller-Design Pattern. Ein weiterer nicht unwesentlicher Vorteil bildet das Netzwerk oder die Community

im Hintergrund des Frameworks. Anhand der guten Dokumentation der Frameworks und der Anzahl der Entwickler, die ein solches verwenden, hat man eine höhere und raschere Chance auf Support sowie Updates und Hilfe bei Problemstellungen. Ebenfalls nicht außer Acht lassen sollte man die finanzielle Komponente. JS-Frameworks sind oft die kostengünstigere Wahl beim Entwickeln von komplexen Frontend-Benutzeroberflächen.

Einer der größten Nachteile ist, dass mit dem eingesetzten Framework auch längere Ladezeiten in Kauf genommen werden müssen. Der Ballast an Code von nicht genutzten, jedoch mitgelieferten Methoden, sowie die großen Anwendungen und Daten sind zwei Gründe, die zu längeren Ladezeiten führen. Des Weiteren ist man bei einem Einsatz von Frameworks an dessen Grenzen gebunden. Damit einher geht die Abhängigkeit von Entwicklern, welche sich auf das bestimmte Framework spezialisiert haben, um komplexe Problemstellungen zu lösen.

Ein Unternehmen sollte außerdem Sicherheitsaspekte der Software genau beleuchten. Da der Programmcode der meisten Frameworks im Internet frei zugänglich ist, wird es Hackern einfacher gemacht, sich Zutritt zu einem System zu verschaffen, als wenn man den Programmcode nicht kennt.

2.6.2 Conclusio

Für die Auswahl des Frameworks waren mehrere Aspekte ausschlaggebend. React.js bietet von allen in Betracht gezogenen Frameworks die verständlichste und übersichtlichste Dokumentation¹⁹ an. Weiteres bildet das Konzept von ReactJS den verständlichsten Ansatz für unseren Einsatz und ist durch die Einbindung als Bibliothek gut handzuhaben aufgrund der relativ einfachen Einbindung in Visual Studio Code. Durch die Verwendung die Syntaxerweiterung JSX können Komponenten effizienter und in einer übersichtlichen Weise programmiert werden.

Ein weiterer Auswahlgrund für React.js waren die eigenständigen Module bzw. Komponenten. Statt komplizierter Seitendesign kann die Website in einzelne logische Komponenten aufgeteilt werden, die unabhängig von anderen Komponenten, sowie dem Seitengefüge existieren. Mit diesem Funktionsprinzip ist ein schnelleres Debugging möglich, womit auch die Website und die Komponenten schneller fehlerfrei und lauffähig entwickelt werden. Eng mit diesem Muster zusammenhängend ist das virtuelle DOM, wodurch nur die veränderten Teile der Website neu gerendert werden müssen. Dies spart Laufzeit und Rechenleistung.

19 Vgl. Rea 19.02.2022

2.7 Gestaltgesetze

Bei der Erstellung von Grafiken und Mustern ist es in der Medientechnik besonders wichtig auf gewisse Regeln zu achten, welche die Aufmerksamkeit der Betrachter erhöhen und intensivieren sollen. Basis dieser Gesetze ist die Lehre von Formen und deren Zusammengehörigkeit und Wirkung.

Gesetz der Nähe

Nahe aneinander gelegene Objekte werden meist als zusammengehörig erfasst. Mittels der Setzung von unterschiedlichen Abständen zwischen mehreren Objekten können somit Gruppierungen dargestellt und veranschaulicht werden.

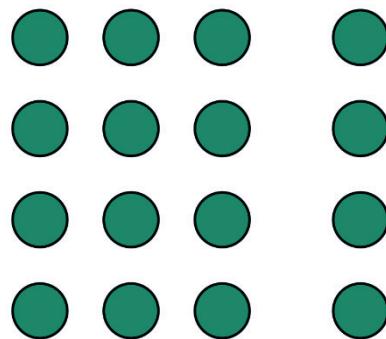


Abbildung 16: Gesetz der Nähe

Gesetz der einfachen Gestalt

Das Gesetz der einfachen Gestalt besagt, dass der Mensch bei komplexen Formen diese vorerst in einfachere Grundformen aufteilt und daraufhin später erst diese als komplex wahrnimmt. Diese Formen werden meist in Dreiecke, Kreise, Rechtecke oder Quadrate aufgeteilt.

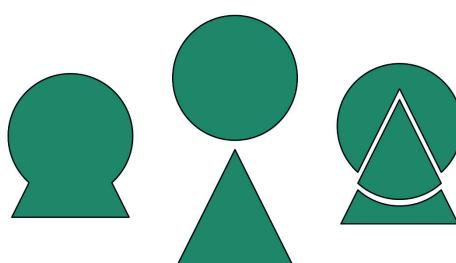


Abbildung 17: Gesetz der einfachen Gestalt

Gesetz der Ähnlichkeit

Objekte, welche dieselbe Farbe, Form oder Größe haben, werden als Gruppe wahrgenommen. Dieses Gesetz basiert darauf, dass Menschen ähnliche Merkmale suchen und daraus eine Gruppierung bilden. So können etwa Objekte mit derselben Farbe und Form dennoch aufgrund ihrer Größe als Teil einer Gruppe erfasst und abgehoben werden.

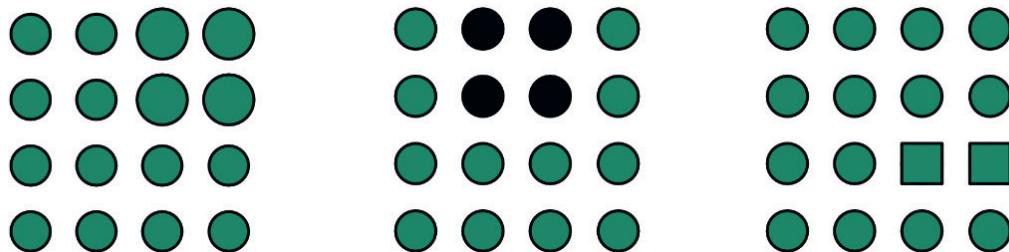


Abbildung 18: Gesetz der Ähnlichkeit

Gesetz der Geschlossenheit

Das Gesetz der Geschlossenheit besagt, dass Objekte, welche sich innerhalb einer von der Gesamtmenge abhebenden Form befinden, als Gruppe wahrgenommen werden. Dies kann durch das Einrahmen oder farbliche Hinterlegen dieser Formen erzielt werden und besitzt ein höheres Gewicht in der Wahrnehmung als das Gesetz der Nähe oder der Gleichheit.

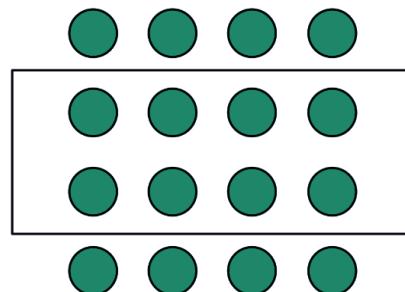


Abbildung 19: Gesetz der Geschlossenheit

Gesetz der Figur-Grund-Trennung

Menschen können Objekte nur differenzieren, wenn sich diese ausreichend voneinander unterscheiden. Es muss somit eine Trennung dieser Objekte vorherrschen, um diese auch als getrennt wahrnehmen zu können. Bewegung, Farben, Konturen, Kontraste und Texturen können dazu beitragen, um Objekte besser voneinander zu differenzieren.

Gesetz der Erfahrung

Aufgrund der Erfahrung von Menschen können bekannte Gestalten oder Formen, trotz starker Abänderung, trotzdem wiedererkannt werden. Ein Baum, welcher stark verformt ist oder eine andere Farbe hat, wird dennoch als Baum erkannt. Weiters spielt dieses Gesetz bei der Wahrnehmung von Gesichtern eine große Rolle. Sie können aus allen beliebigen Betrachtungswinkeln erkannt werden.

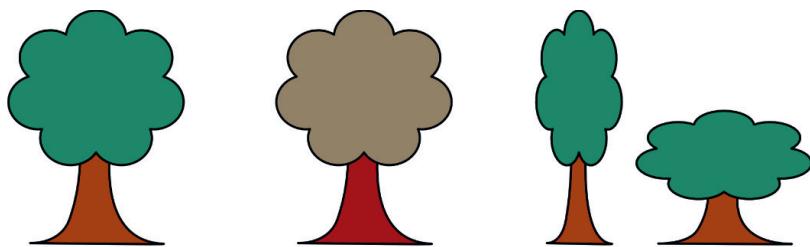


Abbildung 20: Gesetz der Erfahrung

Gesetz der Konstanz

Objekte werden stets anhand ihres Kontextes wahrgenommen und können somit trotz derselben Eigenschaften voneinander differenziert werden. Wichtig für diese Differenzierung ist etwa ein nahegelegenes Referenzobjekt oder eine Hintergrundfarbe, welche die Wahrnehmung der Hauptfarbe beeinflusst.



Abbildung 21: Gesetz der Konstanz

Gesetz der Fortsetzung

Das Gesetz der Fortsetzung besagt, dass Objekte, welche sich überschneiden vom Menschen trotzdem separat wahrgenommen werden. In Abbildung 22 kreuzen sich zwei Linien an einer Stelle. Das Gehirn geht nicht davon aus, dass Linie 1 an der Stelle A fortgesetzt wird, da es sonst ein Richtungswechsel wäre.

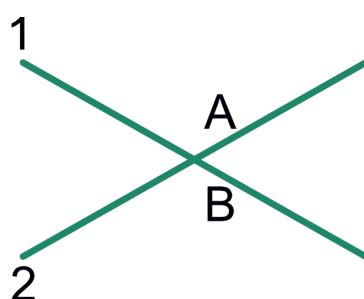


Abbildung 22: Gesetz der Fortsetzung

2.8 Werkzeuge und Wissen

Die Durchführung einer praktischen Arbeit in diesem Umfang bedingt reiches Vorwissen und Fachkenntnis. Hier sind einige der Arbeitstechniken aufgelistet, welche für die Diplomarbeit relevant sind, jedoch schon davor entweder im Schulunterricht oder in anderweitigen Arbeiten erlangt wurden. Weiters werden die wichtigsten Softwareprodukte kurz beschrieben, welche im Rahmen der Umsetzung des Projektes zum Einsatz kommen.

Neben dem unmittelbaren Erarbeiten des Projektes ist das Projektmanagement ein wichtiger Erfolgsfaktor. Es wurde zu einigen Methoden und Management-Prozessen recherchiert, dessen Ergebnisse im Abschnitt 2.8.2 Projektmanagement festgehalten sind.

2.8.1 Software

Es folgt eine Auflistung der Software, welche im Zwecke dieses Projektes zum Einsatz kommt.

Adobe InDesign

Adobe InDesign ist ein Layout- und Satzprogramm, welches von Werbeagenturen und Grafikern sowie in Druckereien und Verlagen für die Erstellung von Magazinen oder anderen Druckprodukten eingesetzt wird. Im Rahmen der Diplomarbeit wird InDesign für die Erstellung der Infomaterialien, wie etwa einem Infoflyer oder der Visitenkarten verwendet.

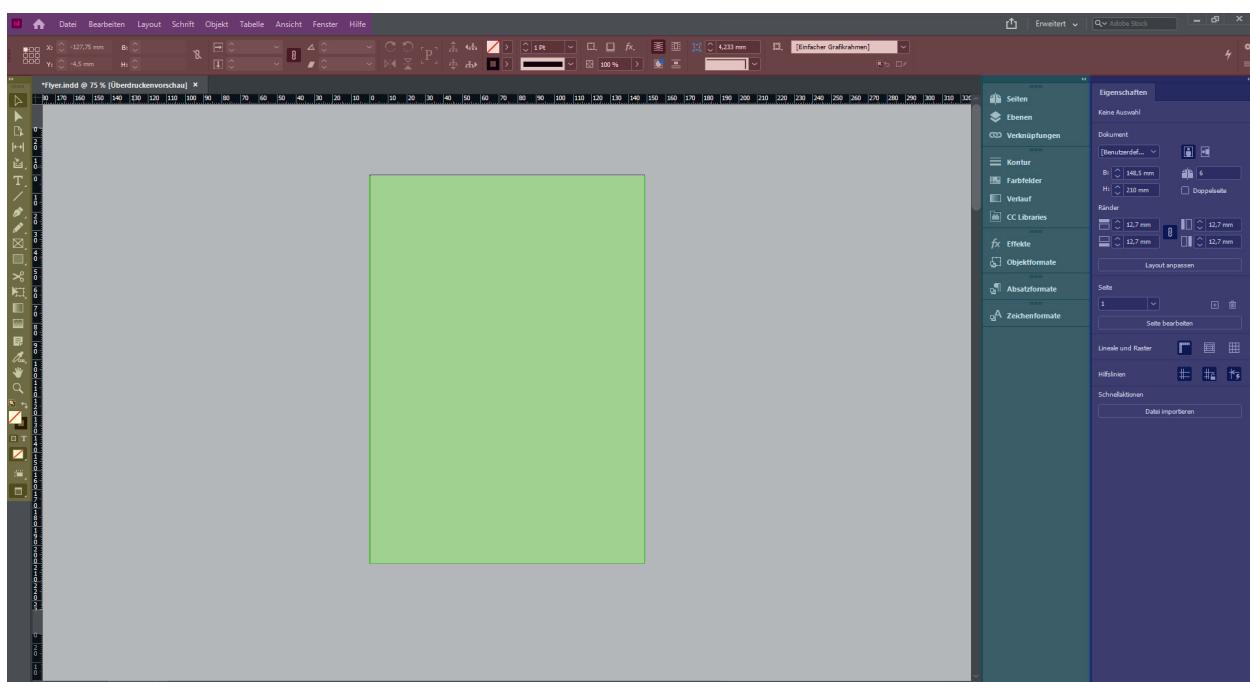


Abbildung 23: Adobe InDesign Benutzeroberfläche

Die in Magenta markierte Fläche zeigt die Menüleiste InDesigns. Es können Dateien exportiert, Ansichten angepasst, Arbeitsbereiche eingestellt, Einstellungsfenster geöffnet und Objekte manipuliert werden. Die in Rot markierte Fläche zeigt ein kontextabhängiges Menü, welches für das ausgewählte Objekt gilt und Möglichkeiten wie etwa das Transformieren und Anordnen bietet.

Die in Gelb markierte Fläche beinhaltet die einzelnen Werkzeuge, mit welchem Objekte oder Texte ausgewählt und manipuliert werden können. Wichtige Werkzeuge, welche häufig verwendet werden sind folgend aufgezählt.

- Mit dem Auswahl und Direktaswahl-Werkzeug können Objekte und einzelne Punkte verschiedener Formen verschoben und transformiert werden.
- Der Zeichenstift ermöglicht es, komplexe Formen zu erstellen. Es werden einzelne Punkte erstellt und miteinander verbunden, diese können rund oder eckig sein und beliebig angepasst werden.
- Der Rechteckrahmen ermöglicht es, Platzhalter zu schaffen und ein grundlegendes Layout zu erstellen.
- Verlaufswerkzeug
- Text-Werkzeug
- Pipette
- Rechteck/Ellipsen-Werkzeug

Die in Grün markierte Fläche ist die Leinwand des Benutzers, in welcher sich die erstellten Elemente befinden. Sie wird bei der Erstellung eines neuen Projekts definiert.

Die violette Fläche zeigt das Fenster Eigenschaften, in welchem ebenfalls Objekte angeordnet und angepasst werden können.

Die blaue Fläche zeigt ein Menü, welches es ermöglicht Seiten anzuordnen, Ebenen zu bearbeiten, Verknüpfungen zu verwalten, Effekte anzuwenden und Farben zu verwalten.

Alle Menüpunkte, welche nicht erwünscht sind, können entfernt und benötigte Punkte hinzugefügt werden. Dies ermöglicht das Erstellen von individuellen Benutzeroberflächen und erhöht dadurch die Effizienz des Benutzers.

Adobe Illustrator

Adobe Illustrator ist ein vektorbasiertes Grafik- und Zeichenprogramm. Es können Logos, Visitenkarten, Flyer oder andere Druck Produkte, welche verlustfrei gespeichert werden, erstellt oder bearbeitet werden. Für die Diplomarbeit wichtige Funktionen im Bereich der Erstellung des Logos und anderer Grafiken werden von Adobe Illustrator bereitgestellt und darin umgesetzt.



Abbildung 24: Adobe Illustrator Benutzeroberfläche

Die in Magenta markierte Fläche zeigt die Menüleiste Illustrators. Hier können ebenso wie in InDesign Dateien exportiert, Ansichten angepasst, Arbeitsbereiche eingestellt, Einstellungsfenster geöffnet und Objekte manipuliert werden. Die in Gelb markierte Fläche beinhaltet die einzelnen Werkzeuge, mit welchen Objekte oder Texte ausgewählt und manipuliert werden können. Wichtige Werkzeuge, welche häufig verwendet werden sind etwa:

- Auswahl und Direktaswahl Werkzeug
- Zeichenstift
- Text-Werkzeug
- Formerstellungswerkzeug: Dieses Werkzeug ermöglicht es mit freier Hand komplexe Formen zu erstellen, das Programm versucht anhand der gezeichneten Form ein Objekt zu erstellen, welches geglättete Kurven und Kanten vorweist.
- Pinsel Werkzeug
- Das Pinsel Werkzeug kann zur stilistischen Erweiterung der Datei, jedoch auch zur groben Konzeptionierung eines Layouts oder einer Form verwendet werden.

Die in Grün markierte Fläche ist wiederrum ähnlich zu InDesign die Leinwand des Benutzers in welcher sich die erstellten Elemente befinden, und wird bei der Erstellung eines neuen Projekts definiert.

Die violette Fläche zeigt die verschiedenen Einstellungsfenster, welche wiederum individuell angepasst werden können und Möglichkeiten zur Farbauswahl, Transformation und Verwaltung von Ebenen bieten. Alle Menüpunkte, welche nicht erwünscht sind, können entfernt und benötigte hinzugefügt werden.

Adobe Photoshop

Adobe Photoshop ist ein Bildbearbeitungsprogramm für Rastergrafiken. Es bietet eine Vielzahl von Funktionen und wird für diverse Anwendungsbereiche in der Bildbearbeitung verwendet. So können etwa Weißabgleich, Tonwertkorrektur, Manipulation von Gesichtsmerkmalen, Anpassung von Belichtung und Farbtemperatur vorgenommen werden. Zusätzlich dazu werden auch Funktionen wie Filter, verschiedene Pinsel und Korrekturwerkzeuge zur Verfügung gestellt.

Im Rahmen der Diplomarbeit wird Photoshop für das Zuschneiden von Bildern, Korrigieren und der Erstellung von Effekten benutzt.

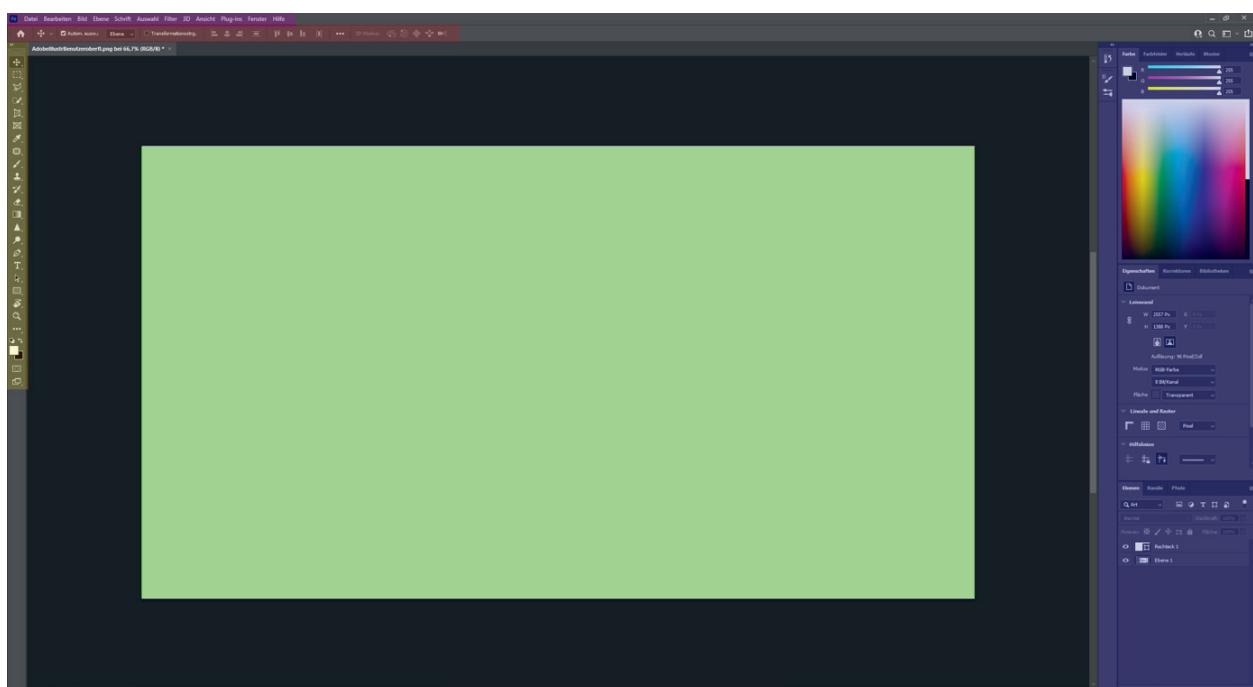


Abbildung 25: Adobe Photoshop Benutzeroberfläche

Die in Magenta markierte Fläche zeigt die Menüleiste Photoshop's, welche die selben Möglichkeiten wie InDesign und Illustrator bietet. Zusätzlich können Filter angewendet und eine Vielzahl von Farb-, Helligkeits- und Tonwertanpassungen vorgenommen werden.

Die in Rot markierte Fläche zeigt wiederrum ein kontextabhängiges Menü, so wie es auch bereits bei InDesign und Illustrator vorherrscht, zusätzlich können noch zum Beispiel für Pinsel Einstellungen vorgenommen werden wie etwa die Deckkraft, oder Härte.

Die gelbe Fläche beinhaltet die einzelnen Werkzeuge mit welchen Ebenen oder Text ausgewählt und manipuliert werden können. Wichtige Werkzeuge, welche häufig verwendet werden sind folgend aufgelistet.

- Auswahlwerkzeuge: Photoshop bietet mehrere Möglichkeiten zur Auswahl von Objekten (z.B. mit freier Hand, automatisch oder per Polygonlasso)
- Pipette
- Ausbessern/Reparatur Pinsel
- Diese Werkzeuge ermöglichen es Bildfehler auszubessern.
- Text-Werkzeug

Zusätzlich zu diesen Werkzeugen wurden meist einige Farb- und Helligkeitsanpassungen vorgenommen. Diese können in der Menüleiste unter dem Punkt Bild->Korrekturen gefunden werden.

Die in Grün markierte Fläche ist wiederrum die Leinwand des Benutzers. Die violette Fläche zeigt die verschiedenen Einstellungsfenster, und bietet die selben Möglichkeiten wie sie auch in Illustrator vorherrschen. Alle Menüpunkte, welche nicht erwünscht sind, können wiederum entfernt und benötigte hinzugefügt werden.

Affinity Designer

Affinity Designer ist ebenso wie Adobe Illustrator ein vektorbasiertes Grafik- und Zeichenprogramm. Es ist mit Geräten wie Tablets kompatibel und bietet sehr ähnliche Funktionen zu Adobe Illustrator. Im Rahmen der Diplomarbeit wird Affinity Designer für das Erstellen von Prototypen der Logos und Grafiken verwendet, da dies im Vergleich zur Desktop Version von Adobe Illustrator einen schnelleren Workflow bietet.

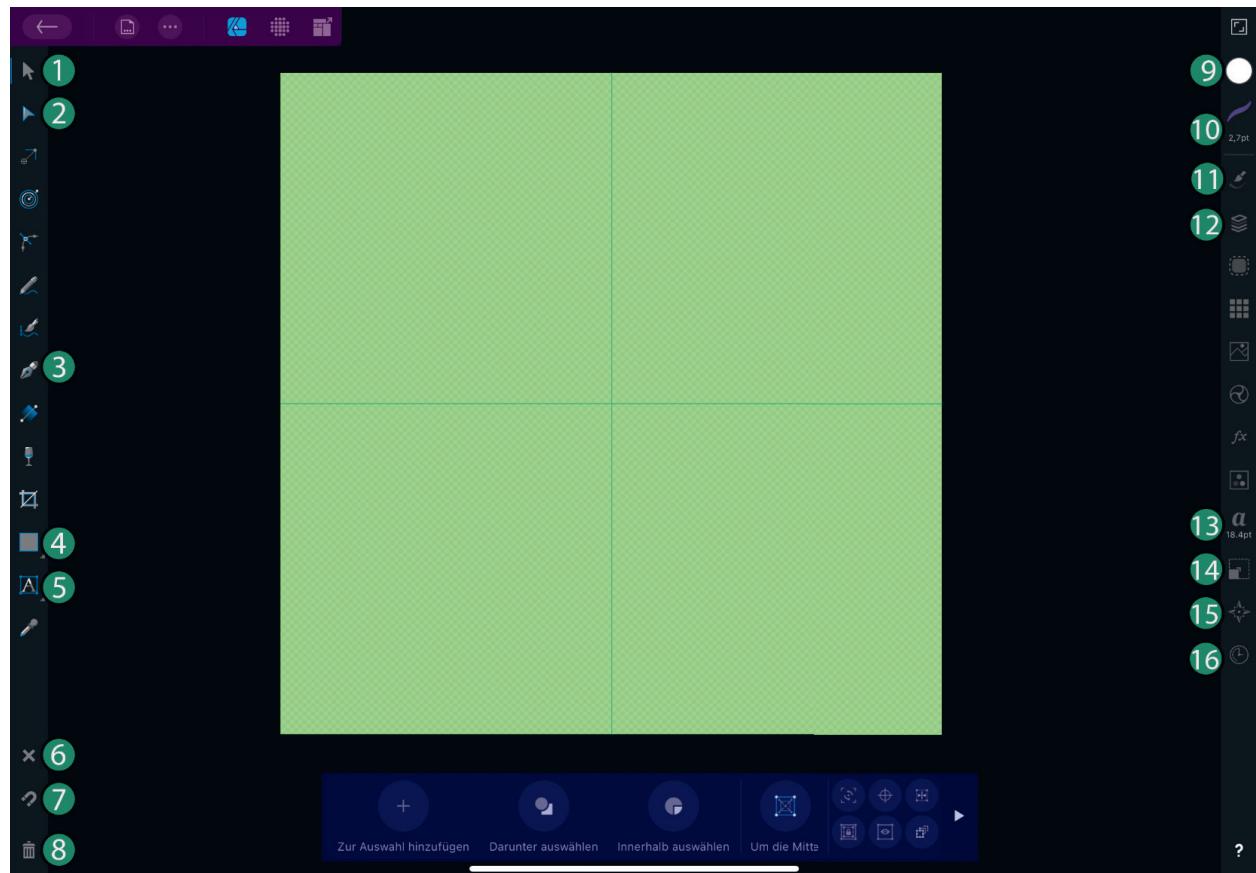


Abbildung 26: Affinity Designer Benutzeroberfläche

Die in Magenta markierte Fläche zeigt die Menüleiste des Affinity Designers, hier können Exporte und Importe vorgenommen werden. Einstellungen wie Hilfslinien oder Raster können ebenfalls getroffen werden. Zudem kann in dieser Leiste zwischen den verschiedenen Benutzeroberflächen gewechselt werden (Vektor, Pixel, Export). Die in Grün markierte Fläche ist die Leinwand des Benutzers und wird bei der Erstellung eines neuen Projekts definiert. Die violette Fläche zeigt dem ausgewählten Werkzeug entsprechende Kontextoptionen. Beispielsweise besitzt das Knoten-Werkzeug in Abbildung 26 Optionen für die Art des Eckpunktes – es können Punkte verbunden, geöffnet oder gelöscht werden.

Folgend sind die Werkzeuge von Affinity Designer erklärt und mit der Nummerierung aus Abbildung 26 gleichgesetzt.

1. Das Transformations-Werkzeug ermöglicht das Verschieben und Transformieren von Objekten und Ebenen.
2. Das Knoten-Werkzeug ermöglicht es, einzelne Punkte von Vektoren zu verschieben oder zu transformieren (Anpassen der Rundungen von einzelnen Vektorpunkten).
3. Das Zeichenstift-Werkzeug ermöglicht es, schrittweise Punkte eines Vektors zu erstellen.
4. Das Formen-Werkzeug ermöglicht es, verschiedene vorgefertigte Formen zu erstellen (z.B. Kreise,

Rechtecke, Dreiecke oder Polygone).

5. Das Text-Werkzeug ermöglicht das Einfügen von Text. Dieser kann auch mit Objekten verbunden und an deren Konturen geführt werden.
6. Deselektieren von ausgewählten Elementen.
7. Die Magnetische-Ausrichtung hilft dem Nutzer bei der Orientierung von Objekten. Es wird die Proportionalität zwischen den Objekten für die Ausrichtung verwendet.
8. Löschen von ausgewählten Elementen.
9. Der Farbwähler ermöglicht es, Farben für Füllung und Kontur zu wählen.
10. Die Kontur-Optionen ermöglichen das Einstellen der Kontur.
11. Die Pinsel-Optionen bieten die Möglichkeit, verschiedene Pinselarten zu verwenden.
12. Die Ebenen-Optionen ermöglichen das Verwalten und Sortieren von Ebenen. Sie können neu erstellt, gruppiert, umbenannt oder gelöscht werden. Zusätzlich ist es möglich die Deckkraft der Ebenen zu ändern oder diese zu sperren, um unbeabsichtigte Änderungen zu vermeiden.
13. Die Schrift-Optionen ermöglichen es, Schriftart, Größe, Ausrichtung, Zeichenabstände und Absätze anzupassen.
14. Die Transformations-Optionen ermöglichen es, Objekte zu spiegeln und zu transformieren.
15. Die Navigator-Optionen ermöglichen es, die Rotation der Zeichenfläche zu sperren und mehrere Ansichtsmodi gleichzeitig zu verwenden (Vektor & Pixel Ansichtsmodus).
16. Das Protokoll beinhaltet den Verlauf der zuletzt vorgenommenen Änderungen. Es kann zu einer älteren Version des Projekts gewechselt werden.

Adobe Premiere Pro

Adobe Premiere Pro ist eine Videoschnittsoftware, welche einen umfangreichen Bereich an Werkzeugen zum Schnitt und Bearbeiten von Videos und Clips bereitstellt. Im Zuge der Diplomarbeit wird Premiere Pro für das Schneiden und Bearbeiten von Videos, welche anschließend auf den Social-Media-Kanälen und der Website veröffentlicht werden, verwendet.

Adobe Lightroom

Adobe Lightroom ist ähnlich zu Photoshop ein Bildbearbeitungsprogramm, welches eine Vielzahl an stilistischen Korrekturen von Bildern zulässt. Es bietet Funktionen wie die schnelle und einfache Anpassung von Bildeigenschaften (Helligkeit, Kontrast, Tiefen, Mitten und Lichter etc.). Lightroom erfüllt den Zweck, Bilder für soziale Medien schnell bearbeiten zu können und diese stilistisch anzupassen.

Adobe XD

Adobe XD ist eine vektorbasierte Grafiksoftware zum Entwurf von grafischen Benutzeroberflächen für Web- und Mobile-Apps. Adobe XD unterstützt sowohl Wireframe, Mockup, Webdesign, als auch interaktives click-through-Prototyping.

Visual Studio Code

Der Code-Editor von Microsoft unterstützt eine Vielzahl von Markup- und Programmiersprachen. Sein modularer Aufbau erlaubt es, Erweiterungen für diverse Anwendungsbereiche dem Editor hinzuzufügen. Visual Studio Code wird für die Editierung diverser Programmcodes verwendet.

Visual Studio

Genauso von Microsoft entwickelt ist Visual Studio eine vollwertige IDE für die Desktop-, Web- und Mobile-Entwicklung. Sie implementiert Code-Highlighting und IntelliSense für C, C++, C#, Visual Basic und weitere von Microsoft unterstützte Produkte. Visual Studio sticht heraus bei Aufgaben wie dem Konfigurieren von Compiler-Parametern und dem Automatisieren von Entwicklungs-Prozessen.

Android Studio

Die auf der IntelliJ IDEA basierte Entwicklungsumgebung hat ihren Einsatz in der Entwicklung von Android-Apps mit Java oder Kotlin. Sie implementiert übliche Funktionen wie die Android Debug Bridge oder einen Layout-Designer.

XCode

XCode ist eine Entwicklungsumgebung von Apple und wird zur Programmierung für macOS/iOS verwendet. XCode kann mit den Programmiersprachen Swift und Objective-C verwendet werden mit dem Cocoa-Framework, ebenso wie die Sprachen C und C++. Ebenso werden IDE-übliche Funktionen wie der Interface Builder, Instruments und der iPhone-Simulator mitgeliefert.

2.8.2 Projektmanagement

Die Vorbereitung auf diese Diplomarbeit enthält auch die Befassung mit verschiedenen Projektmanagement-Methoden und -Programmen.

Quire.io

In der Webanwendung Quire.io können Arbeitspakete als Liste in einem GANTT-Diagramm (Zeitleiste) oder auf einem KANBAN-Board visualisiert werden. Die Kollaboration mit maximal 10 Teammitgliedern ist kostenlos; bei größeren Projekten sind monatliche Gebühren zu zahlen.

Quire.io wird für die Organisation, Zuteilung und Kategorisierung von Tasks verwendet. Die Anwendung ist alleinstehend ohne Verwendung von weiteren Zeitmanagement-Tools, welche die Verknüpfung zu den Arbeitspaketen bräuchten, gut einsetzbar. In Quire.io fehlt jedoch die Option Arbeitspaketen einen Arbeitsaufwand zuzuweisen.

Monday.com

Ähnlich wie Quire.io ist Monday.com eine reine Webanwendung für Projektmanagement und Team-Kollaboration. Sie ist weitaus umfangreicher und bietet verschiedene Vorlagen für andere Branchen und Aufgabenbereiche, wie beispielsweise Marketing, Vertrieb oder Recruiting.

Die Software ist für Teamgrößen von maximal 2 Personen kostenlos nutzbar. Die Bearbeitung der Tasks in GANTT-Diagrammen benötigt den Kauf des Abonnements.

Jira

Von dem Software-Unternehmen Atlassian entwickelt ist Jira ein Vorreiter im agilen Projektmanagement. Neben klassischer Arbeitspaket-Organisation und -Zuweisung erlaubt die Webanwendung auch die Erstellung von Berichten und Diagrammen zur Kontrolle des Sachfortschritts.

Jira ist primär für die Verwendung im Projektmanagement mit der Methode SCRUM entwickelt. Die Grundprinzipien der Software basieren auf einem Backlog mit Tasks, Sprints und SCRUM-Rollen.

Tabellarische Aufzeichnung

Weniger automatisiert jedoch stärker individualisierbar ist ein Management mithilfe von einfachen Tabellen und verknüpften Verzeichnissen. Dafür kann beispielsweise die Software Microsoft Excel eingesetzt werden. Diese Methode ist vor allem für einfache Aufzeichnungen und Organisation in Listen geeignet.

Das Softwareprodukt Microsoft Project ist zur Vollständigkeit der Recherche miteinzubeziehen. Zu Lasten dessen Inkompatibilität mit Betriebssystemen, welche von Windows abweichen, ist es jedoch für die Teammitglieder, welche MacOS oder Linux verwenden, unbrauchbar.

Schlussfolgernd hat sich das Team der Diplomarbeit für die Organisation der Arbeitspakete mit der Webanwendung Quire.io entschieden. Zusätzlich sollen primitive Aufzeichnungen in Tabellen mitgeschrieben werden. Dieses System bietet eine hohe Flexibilität mit genug Automatisierung für die Planung hierarchischer Task-Listen.

3. Ergebnisdokumentation

3.1 Einleitung

3.1.1 Was ist BambiGuard?

BambiGuard greift einen Großteil der Methode des manuellen Abfliegens auf, bei dem eine Drohne händisch gesteuert und das Wärmebild vom Piloten ausgelesen wird. Als Ergänzung dazu automatisiert die BambiGuard App die Flugplanung, die Flugsteuerung und die Erkennung der Rehkitze. Mithilfe einer auf der Drohne montierten Wärmekamera (DJI Mavic 2 Enterprise Dual, im Abschnitt 2.2 Drohne genauer beschrieben) sollen Rehkitze automatisch durch eine Bilderkennungs-Software erkannt werden. Die Berechnungen finden auf der BambiGuard Smartphone-App statt, welche die originale von DJI entwickelte App vollkommen ersetzt. Weiters können sich Helfer über dieselbe App mit dem Piloten der Drohne verbinden, um bei Funden von Rehkitzen benachrichtigt zu werden. Eine Kompass-Funktion leitet die Helfer in die Richtungen der Fundorte, um Kommunikationsschwierigkeiten zu minimieren.

Der Mehrwert von BambiGuard gegenüber anderen Verfahren ist die Zeitersparnis, Genauigkeit, Effektivität und Erleichterung der Suche und der Kommunikation zwischen dem Piloten und den Helfern.

Das konkrete Ergebnis des Projekts ist eine App für Android und iOS, die mit einer mit Kamerاسensoren ausgestatteten Drohne kommuniziert, diese steuert und die Bilddaten auswertet. Die mit der Drohne verbundene Instanz der App führt eine Bilderkennung zum Identifizieren von Rehkitzen aus. Andere Instanzen gelten als Helfer und werden per GPS zu Fundorten geleitet. Darüber hinaus soll auf das Thema beworben und darauf aufmerksam gemacht werden. Dafür kommen mehrere Werbematerialien und Medien zum Einsatz. Es wird außerdem eine Website zur Repräsentation der Arbeit entwickelt.

3.1.2 Themenabgrenzung

BambiGuard ist keine von Grund auf neue Idee, sondern baut auf vorhandene Techniken auf. Die Diplomarbeit bezweckt eine Verbesserung der derzeitigen Systeme und erleichtert es, Felder mit einer Drohne abzusuchen und Rehkitze zu finden. Es existieren bereits Apps zur Drohnensteuerung und es existieren bereits Drohnen, welche für die Rehkitzrettung verwendet werden. Es wird weder eine eigene Drohne noch eine von Grund auf neue Steuerung entwickelt. Die Arbeit bildet eine notwendige Verbesserung dieser genannten Bereiche und erweitert diese beiden um ein intelligentes Leitsystem für

Helper.

3.1.3 Entwicklungsprozesse

Das Projekt BambiGuard wird nach der Methode des Wasserfall-Modells geplant und durchgeführt. Demnach sind die Projektabschnitte und Arbeitspakete vor der Durchführung geplant worden. Die Aufteilung der Arbeit auf die Projektmitglieder findet im vornherein statt, wodurch unabhängige Arbeitspakete nebeneinander stattfinden können.

Der Programmcode, welcher im Rahmen der Bilderkennung und der BambiGuard Android App entsteht, wird laufend mit dem Versionierungswerkzeug Gitlab synchronisiert. Dadurch werden Änderungen im Programm protokolliert und unterschiedliche Versionen des Codes sichtbar.

3.2 Software-Architektur

3.2.1 Anforderungen

Das BambiGuard-System soll die Probleme gängiger Methoden zur Rettung von Rehkitzen mithilfe von technischer Automatisierung lösen. Für die Auflistung und klare Definition der Anforderungen bzw. der Features der Anwendung ist eine Liste mit Punkten, wie User Stories, erstellt worden.

- Die BambiGuard App soll genauso für **iOS**, wie für **Android** dieselben Funktionen haben und vom Umfang ident sein.
- Ein **Pilot** (Vorkenntnisse im Drohnenflug vorhanden) kann innerhalb der App die Flugplanung, Drohnensteuerung und Verwaltung der Helper durchführen. Er soll die Anwendung nicht verlassen müssen.
- Vor dem Flug definiert der Pilot die **Mission**, welche nach dem Start im automatischen Flug ausgeführt wird. Er soll jederzeit manuell eingreifen können.
- Die **Erkennung** der Wildtiere erfolgt ohne Zutun des Piloten.
- **Helper** können sich mit derselben App für einen Flug zur Verfügung stellen. Sie sollen ohne Vorkenntnisse bei der Suche mithelfen können. Die App lotst sie per GPS zu einem gefundenen Rehkitz, welches von ihnen händisch aus dem Feld getragen wird. Ein Pilot kann mehrere Helper zu einer Mission hinzufügen.
- Das Ziel ist die Ausarbeitung eines Systems, welches funktional sein soll, jedoch nicht für die Veröffentlichung oder kommerzielle Nutzung gedacht ist.

3.2.2 Tech-Stack

Aus den im vorherigen Schritt definierten Anforderungen wird das Konzept von BambiGuard entwickelt. Die Auswahl der Technologien richtet sich nach den Vorkenntnissen und Erfahrungen der Teammitglieder, sowie der Recherche im Zuge der Grundlagen und Methoden.

Komponenten von BambiGuard

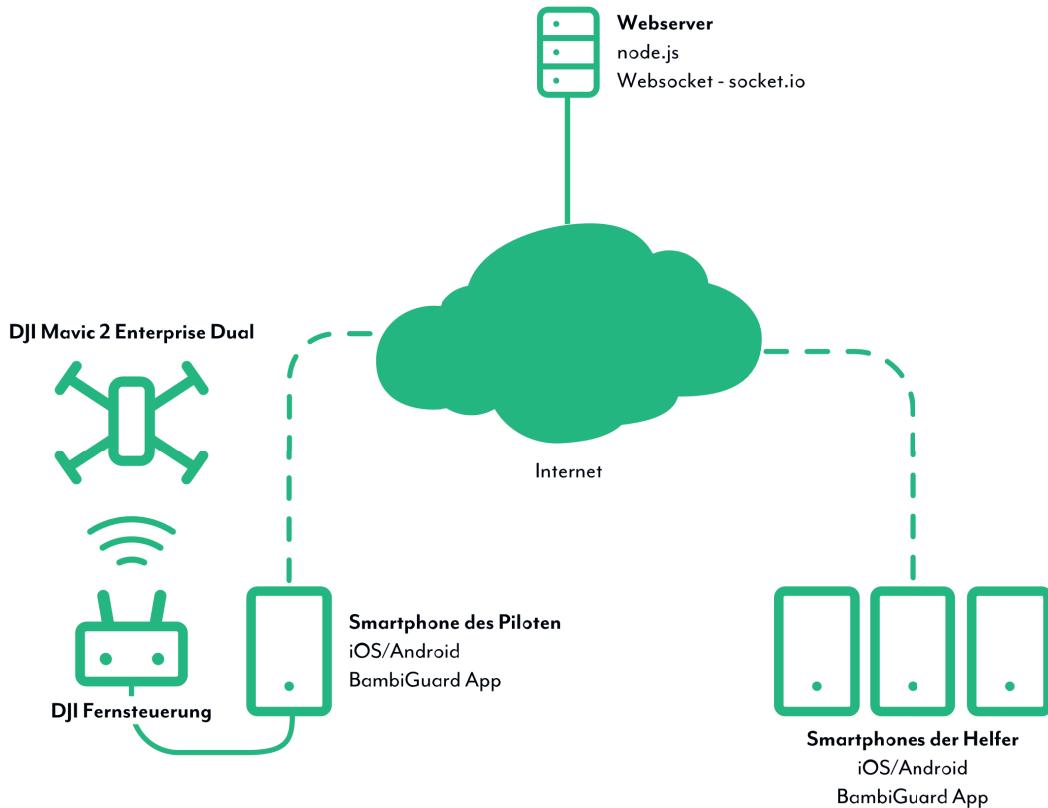


Abbildung 27: Komponentendiagramm

Das Smartphone des Piloten, auf welchem die BambiGuard-App verwendet wird, ist mit der Fernsteuerung per Kabel verbunden. Der Pilot ist nur für das Planen und Kontrollieren des Flugs zuständig. Anfangs kann er einen Flug erstellen, Helfer hinzufügen und das abzufliegende Feld definieren. Während des Fluges bekommt er in Echtzeit die Videodaten der Kamera, Informationen über das Fluggerät (Akkustand, Flughöhe, Distanz etc.) und Nachrichten bzw. Standorte, wenn Rehkitze detektiert werden.

Alle Helfer tragen Smartphones mit der BambiGuard-App mit sich. Sie werden vom Piloten zu seiner Sitzung hinzugefügt und bekommen laufend Benachrichtigungen, wenn Rehkitze gefunden werden. Die Aufgabe der Helfer ist, auf einen Fund zu reagieren, das Rehkitz aus dem Gefahrenbereich der Wiese zu bringen und das Tier danach in der App als gefunden zu markieren, um den Piloten und andere Helfer zu informieren.

Beide Versionen der App (iOS und Android) implementieren die Bibliothek Socket.io (beschrieben im Punkt „Komponenten“) und die entsprechend von DJI zur Verfügung gestellten Software Development Kits (SDK) für die Anbindung zur Drohne (Mobile SDK) und für die Darstellung des User Interfaces (UX SDK).

Die verwendeten Sprachen und Technologien sind folgend aufgelistet:

- **Java** ... Android App
- **Swift** ... iOS App
- **C++** ... Bilderkennung
- **JavaScript** ... Node.js Server
- **XML** ... Android Layout Markup

Kommunikation zwischen den Geräten

Die gesamte Kommunikation zwischen dem Gerät des Piloten und den Geräten der Helfer erfolgt mithilfe von WebSocket-Verbindungen über einen Webserver. Andere Optionen der Kommunikation wie Bluetooth, Bluetooth Low Energy, WiFi und NFC sind für den Anwendungsfall ungeeignet, da sie eine zu geringe Reichweite im Außenbereich haben (Bluetooth, als weitreichendste Peer-to-Peer Technologie, schafft in der Version 4.2 in guten Bedingungen Outdoor eine Reichweite von rund 60 Metern¹). Außerdem bringt die Verwendung dieser Technologien mit dem Ziel iOS- und Android-Geräte miteinander zu verbinden große Hürden mit sich, da die Implementierungen zu einem großen Teil nicht miteinander kompatibel sind. Aus diesen Gründen werden auf eine Internet-Verbindung der Smartphones und eine zentrale Server-Instanz zurückgegriffen.

Der Webserver wird von der Umgebung Node.js betrieben und implementiert die Bibliothek Socket.io, welche auch in der Android und iOS App verwendet wird. Diese Bibliothek macht es einfach, über einen WebSocket mit verschiedenen Systemen zu kommunizieren, da sie auf sehr vielen Plattformen verfügbar ist. Sie bietet eine konstante und schnelle Verbindung von Client zu Server, mit zugleich geringem Entwicklungsaufwand.

Der Zweck des Servers ist zuerst dem Piloten und den Helfern die Möglichkeit zu bieten, sich in eine gemeinsame Sitzung zu begeben. Nachdem sich die Helfer mittels Namen als Identifikation dem Server bekannt gegeben haben, sieht der Pilot sie in einer Liste und kann sie dem Flug hinzufügen. Danach leitet er alle Informationen direkt von Smartphone zu Smartphone, da jegliche Verarbeitung, wie die Bilderkennung, auf den Endgeräten stattfindet.

Vorteile und Nachteile des Designs

Durch die Einbindung der Helfer in einer variablen Art ist die Anwendung für verschiedenen Feldgrößen skalierbar. Die Flugplanung sowie der Flug selbst sind unabhängig von der Größe des Gebietes, einzig muss die Drohne immer in Sichtweite des Piloten bleiben. Die Aufgaben können mit Helfern auf mehrere Personen aufgeteilt werden und es ist keine verbale Kommunikation während des Fluges nötig. Dabei kann jede Person mit einem Smartphone, welches GPS und eine SIM-Karte besitzt als Helfer fungieren.

1 Siehe <https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/#estimator> 04.03.2021

Mit BambiGuard muss nur eine Person eingeschult sein und den Drohnenführerschein besitzen. Der größte Vorteil des Systems ist jedoch die gesteigerte Effizienz im Vergleich zu herkömmlichen Methoden zur Suche von Rehkitzen. Das Feld kann ohne Stopp komplett abgeflogen werden und die Helfer suchen bei Entdeckungen parallel nach den Tieren. Der Pilot kann sich zur Gänze auf das Fliegen konzentrieren und muss die Drohne nicht anhalten, um auf die Rettung eines Rehkitzes zu warten.

Durch die Verbindung der Smartphones über das Internet können zwar beliebige Distanzen zwischen ihnen sein, jedoch muss eine permanente Aufrechterhaltung einer Internetverbindung aller Smartphones gewährleistet sein. Das kann in abgelegenen Gebieten oder in höheren Lagen zu Problemen führen. Auch die Verdeckung der Sicht der Kamera durch Pflanzen kann problematisch bei der Erkennung von Wildtieren sein.

Ein in der Planung unvorhersehbares Problem ist die Genauigkeit der GPS-Sensoren in den Smartphones der Helfer. Nachdem die Position des Rehkitzes genau bestimmt wurde, müssen die Smartphones der Helfer abhängig von ihrer Position die Richtung zum Tier weisen. Die verschiedenen Hersteller und Bauarten von Android- und iOS-Geräten machen es unmöglich, eine Aussage über die Spezifikationen der Sensorik zu treffen.

3.2.3 Schnittstellen

Folgend werden komponenten-übergreifende Schnittstellen definiert, welche bei der Dokumentation der Entwicklung der einzelnen Bereiche nicht genauer beschrieben werden.

Socket.io

Informationen werden eventbasiert vom Helfer zum Piloten und umgekehrt über den Webserver übertragen. In Tabelle 3 sind die wichtigsten Events aufgelistet und beschrieben.

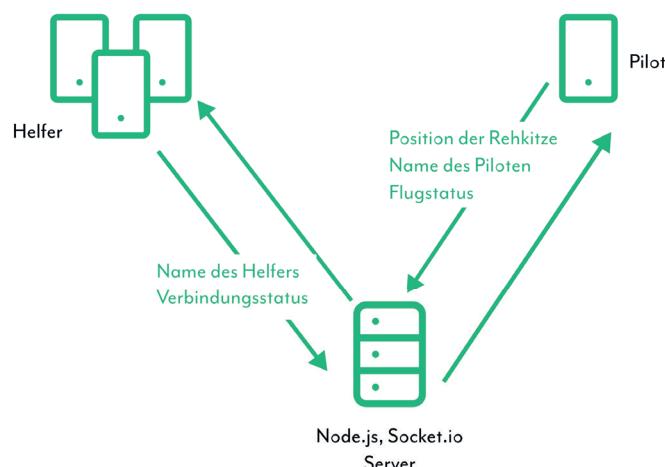


Abbildung 28: Kommunikation zwischen Pilot, Server und Helfer

Event	Richtung	Beschreibung
user_identification	Pilot/Helper – Server	Bekanntgabe, ob man Helper oder Pilot ist
get_helpers	Pilot – Server	Pilot fordert Liste der Helper an
acquire_helper	Pilot – Server	Pilot fügt Helper zu Mission hinzu
join_flight	Server – Helper	Server informiert Helper über Flug
bambi_found	Pilot – Helper	Pilot entdeckt Rehkitz
bambi_rescued	Helper – Pilot	Helper hat Rehkitz gerettet

Tabelle 3: Wichtige Events der Schnittstelle zwischen Pilot, Server und Helper

NDK

Das Android Native Development Kit (Android NDK) ist die Schnittstelle zwischen der Java-Applikation und der in C++ geschriebenen Bilderkennung. Der Datenaustausch erfolgt mittels Pointer. Die Funktion der Bilderkennung erhält von der Android App als Parameter ein Pointer zu einem Array (Bytes oder Integer) mit den Bilddaten und die Größenangaben dieses Arrays. Als Rückgabewert wird eine Liste der Positionen der erkannten Rehkitze zurückgegeben. Die Positionen sind als Abstand in Pixeln zum Mittelpunkt des Bildes definiert.

3.3 App-Design

Zu Beginn der Entwicklung werden Anforderungsprofile der Nutzer von der zukünftigen App BambiGuard erstellt. Diese wurden durch intensive Gespräche mit Landwirten, Jägern und Tierschützern geführt. Die aufgekommenen Argumente konnten weiters auf einige Hauptpunkte zusammengefasst werden. Die App sollte demnach folgende Anforderungen erfüllen:

- Einfache und unkomplizierte Bedienung
- Effiziente Durchführung der Absuche, ohne lange Vorbereitungszeit oder Nacharbeit
- Für den Laien selbsterklärend und intuitiv gestaltet
- Überschaubarkeit auf Flugweg, verbleibende Strecke sowie geschätzte Restzeitdauer

User Stories

Aufgrund dieser fünf Hauptargumente wurden für die Bedienung als Drohnenpilot und Helper zwei verschiedene Arten von User Stories herausgearbeitet.

- Als Drohnenpilot möchte ich, dass ich mich auf den Drohnenflug konzentrieren können, damit dieser reibungslos abläuft.
- Als Drohnenpilot möchte ich, dass ich Rückmeldungen zum Status des Fluges und der Rehkitzsuche bekomme.
- Als Drohnenpilot möchte ich, dass das System zuverlässig und ohne lange Vorbereitung funktioniert, damit ich schnell mit einem effektiven Flug starten kann.

- Als Drohnenpilot möchte ich, dass ich schnell und unkompliziert meine Helfer dem Flug hinzufügen kann, um keine Zeit zu verlieren.
- Als Helfer möchte ich, dass ich zuverlässig und genau zum Rehkitz geleitet werde, damit keine unnötigen Wege oder Wartezeiten durch Kommunikation entstehen.
- Als Helfer und Laie möchte ich, dass die Handhabung der App selbsterklärend und intuitiv ist, damit keine langen Lernzeiten entstehen.
- Als Helfer möchte ich nachsehen können, wie weit der Drohnenflug fortgeschritten ist, um die verbleibende Strecke und Zeitdauer einzuschätzen.

3.3.1 Modell

Auf Basis der Nutzerprofile und der User Stories wurde das erste Modell unserer Benutzeroberfläche erstellt. Es stellt ein erstes Grundgerüst dar und gibt einen Überblick in welche Dimension und Entwicklung das weitere App-Design laufen wird. Das Modell stellt ein grobes Wireframe zur festlegung der Platzaufteilung dar.

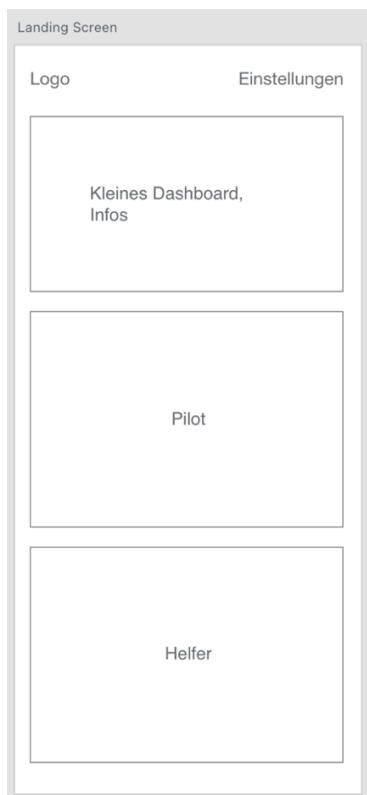


Abbildung 29: Landing Screen
(Modell)

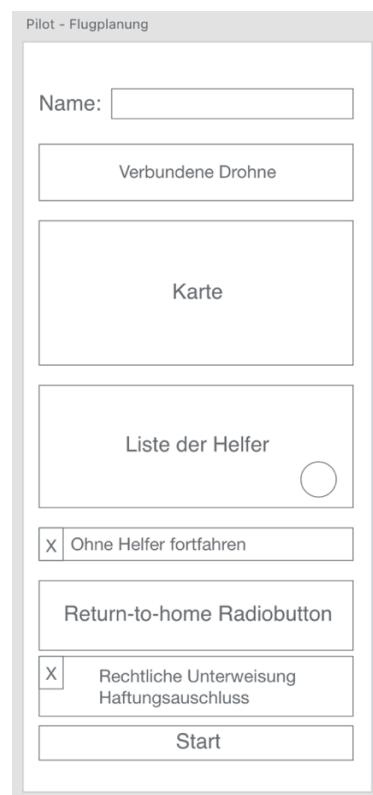


Abbildung 30: Flugplanung
(Modell)

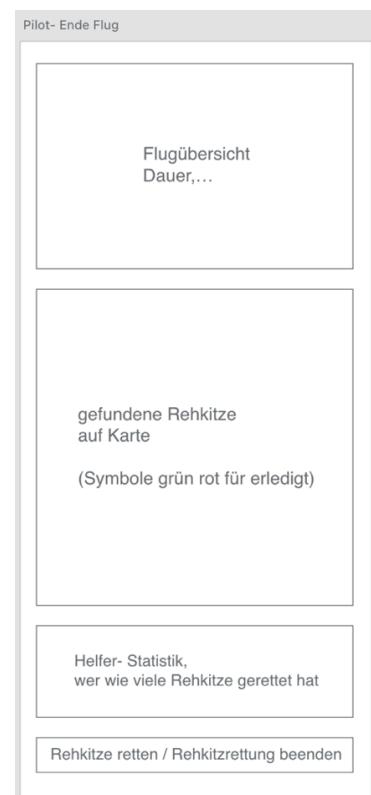
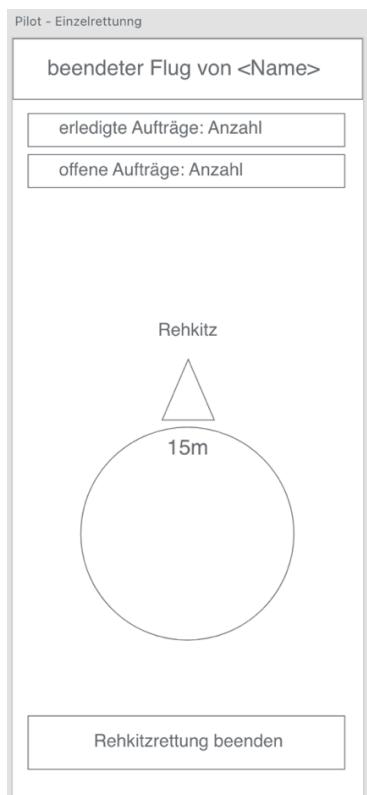
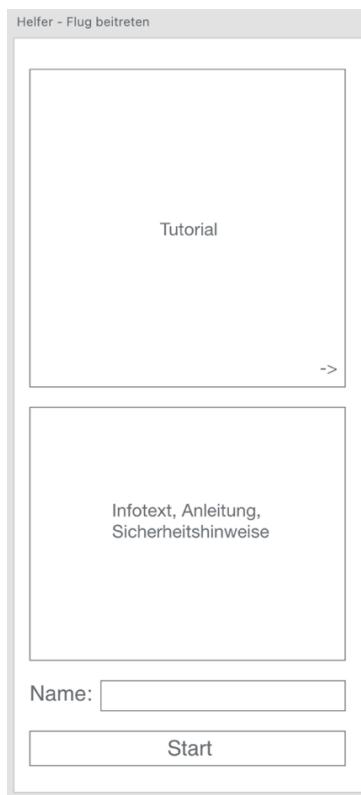
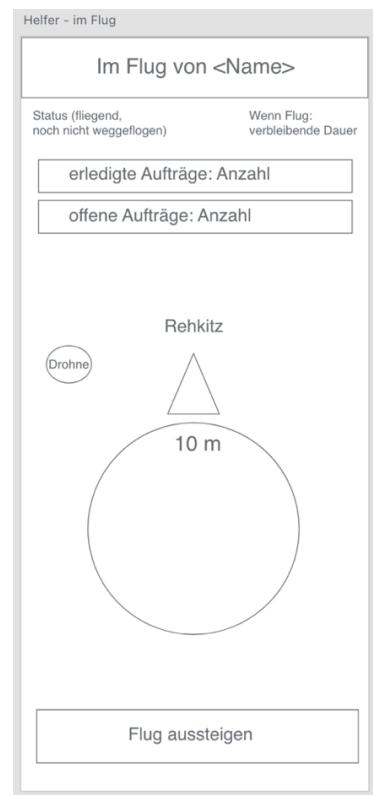


Abbildung 31: Pilot – Flugende
(Modell)

**Abbildung 32:** Pilot – im Flug (Modell)**Abbildung 33:** Pilot – Einzelrettung (Modell)**Abbildung 34:** Helfer – Flug beitreten (Modell)**Abbildung 35:** Helfer – im Flug (Modell)

Landing Screen

Als Ankunfts- und Plattform wurde eine zentralen Screen für Pilot und Helfer gewählt. Der Landing Screen teilt sich in drei Grundbereiche: dem Informationsteil, dem Pilotenauswahlbutton, sowie dem Helferauswahlbutton. Im Informationsteil werden flugrelevante Daten wie Wetter, Datum, Uhrzeit sowie allgemeine Infos zur Bedienung angezeigt. Die beiden Buttons darunter führen zu den jeweiligen spezifischen Screens (Pilot bzw. Helfer). Über dem Informationsteil sollen das Logo sowie ein Einstellungsbanner platziert werden, um Grundeinstellungen zu treffen.

Pilot – Flugplanung

Bei der Flugplanung werden die grundlegenden Einstellungen für den Flug getroffen. Der Pilot muss im ersten Abschnitt seinen Namen eintragen, der für die Helfer später in ihrer Ansicht erscheint. Währenddessen soll das Smartphone bereits mit dem Drohnencontroller verbunden sein. Aufgrund dessen wird im nächsten Bereich bereits das verbundene Drohnenmodell erkannt und dargestellt. Nachfolgend werden in der Kartenansicht die Eckpunkte des Feldes gekennzeichnet sowie das Abflugmuster im Hintergrund berechnet. Ein Pflichtfeld für die weiteren Schritte bildet das Hinzufügen von Helfer, welche sich zuvor in der Helferansicht registrieren müssen, um in der Pilotensicht hinzugefügt werden zu können. Der Kreis im Feld der Helfer bildet den Hinzufügen-Button. Ein darüber hinaus gehendes Ziel stellt die Absuche von nur einer Person, dem Piloten dar, hierfür wird ein Button „Ohne Helfer fortfahren“ platziert. Um im Falle eines Abbruchs oder einer Beendigung dem Fluggerät mitzuteilen, wie der Flug beendet werden soll (selbstständiges Zurückfliegen an die Startposition oder ein manuelles Zurückfliegen) werden Radio-Buttons zur Auswahl dieser Varianten verwendet. Um das rechtliche Belangen sowie die Eigenverantwortung des Drohnenpiloten in den Mittelpunkt zu stellen, wird eine Checkbox zur rechtlichen Absicherung des BambiGuard-Teams eingerichtet. Werden alle oberen Bereiche korrekt ausgefüllt, wird der Start-Button klickbar und die Absuche mithilfe der Drohne kann gestartet werden.

Pilot – im Flug

Die Planung der „im-Flug“-Ansicht ist sehr unsicher und wage. Die Ansicht sowie das Design sind sehr stark vom verwendeten Hersteller DJI abhängig. Die momentane Planung wurde aufgrund von vorhanden Ansichten des Herstellers gewählt. In der oberen Leiste sollen sich die Komponenten in folgender Reihenfolge befinden: Logo - Dauer des Fluges – Anzahl der bereits gekennzeichneten Rehkitze – Signal, Akku, Windwarnung, Verbindung – Einstellungen

In der Mitte der Ansicht wird sich wie in bereits existierenden DJI-Apps die Karte befinden, zusätzlich werden hier noch das gekennzeichnete Feld mit dem Abflugmuster angezeigt. Der rechte Button wird zum Stopp-Button umfunktioniert und dient zum Abbrechen des automatischen Fluges. In den Bereichen unterhalb der Karte befinden sich die Kamera-Ansicht (Ansicht der Fotokamera), die technischen Daten zum Flug (Fertigstellungsgrad, Komplikationen, ...) sowie eine Statistik der Helfer (welcher Helfer wie viele Rehkitze bereits gefunden hat).

Pilot – Ende Flug

Nach Beendigung des Fluges, wird der Pilot auf die „Ende Flug“-Ansicht geleitet. Die Ansicht teilt sich in drei Bereiche: der Flugstatistik (hier werden Flugübersicht, Dauer des Fluges, erfolgreiche Beendigung, etc. angezeigt), der Rehkitzstatistik (gefundene Rehkitze werden auf der Karte gekennzeichnet;

grün bedeutet gefunden und gerettet; rot für gefunden und noch nicht gerettet) und der Helfer-Statistik (eine Liste, welcher Helfer wieviele Rehkitze gefunden hat). Nach diesen drei Bereichen befindet sich der Abschluss-Button. Wenn die Absuche mit Helfern ausgewählt wurde, gelangt man mit diesem Button wieder auf den Landing Screen. Wurde jedoch die Auswahl „Ohne Helfer fortfahren“ gewählt, so gelangt man in die „Pilot – Einzelrettung“.

Pilot – Einzelrettung

Die Einzelrettung des Piloten ist als Bonusziel definiert, daher wird sie nur bei ausreichender Zeit in Betracht gezogen. Die BambiGuard-App wurde grundsätzlich so konzipiert, dass ein Pilot und mehrere Helfer zusammenhelfen, um die Effektivität und Effizienz zu maximieren. Ist dies nicht möglich, wurde die Möglichkeit zur Einzelrettung entworfen. Hierbei wird der Pilot mit einem in der Mitte der Ansicht positionierten Kompass zum gefundenen Rehkitz geleitet. Der Pfeil symbolisiert die Richtung, in der das Rehkitz liegt. Darunter werden die errechneten Meter zur Position angezeigt. Die beiden oberen Anzeigeleisten sollen einen Überblick über den Fortschritt kennzeichnen. Durch die Beendigung mit dem Button „Rehkitzrettung beenden“ wird man wieder auf den Landing Screen geleitet.

Helper – Flug beitreten

Durch den Klick auf den Button „Helper“ am Landing Screen gelangt man in den Helper-Modus und in die Ansicht „Flug beitreten“. Wie auch andere Ansichten, wird auch diese in vier Sektoren gegliedert. Der erste Sektor dient dem Helper, allgemeine Informationen zum Such- und Rettungsvorgang zu erhalten. Das Tutorial wird mit Hilfe von Texten und Bildern veranschaulicht. In der weiteren Sektion findet man einige Sicherheitshinweise sowie rechtliche Komponenten für die Mithilfe der Rehkitzrettung. In der letzten Sektion muss sich der Helper im System registrieren, um auf dem Flugplanungs-Screen des Piloten hinzugefügt werden zu können. Mit „Start“ wird die Aktion freigegeben und dem Piloten angezeigt. Wird der Helper erfolgreich hinzugefügt, so wird dieser erst dann auf die „Im Flug“-Ansicht weitergeleitet.

Helper – im Flug

Die „Im Flug“-Ansicht bildet die Hauptfunktion des Helpers ab. In der Titelleiste wird der Name des Piloten angezeigt. Darunter kann man erkennen, ob der Flug bereits begonnen hat oder nicht, ebenso wie die voraussichtliche Dauer des Fluges (Restflugdauer). Darunter befindet sich eine Statistik der erledigten und offenen Aufträge. Aufträge kommen als Pop-Up auf den Screen des Helpers und müssen angenommen oder abgelehnt werden. In der Mitte der Ansicht befindet sich der Kompass, der in Richtung der gefundenen Rehkitze zeigt und die geschätzte Wegroute dorthin berechnet. Als Sicherheitsaspekt könnte die Drohnenposition angezeigt werden. Mit dem Klick auf den Button „Flug aussteigen“ gelangt der Helper wieder auf den Landing Screen zurück.

3.3.2 Wireframe

Aus dem erstellten Modell bildeten sich zwei Ansätze zur Umsetzung des App-Designs. Der erste Ansatz wurde sehr stark an das Modell angelehnt. Mit diesem wurde zunächst auch die Programmierung gestartet. Nach einiger Zeit wurde der erste Ansatz verfeinert und neu entworfen, einige geplanten Features wurden überflüssig oder hätten keinen Mehrwert für den Nutzer gebracht. Darüber hinaus wurde das vorhandene Konzept auch auf UI/UX-Gesetze geprüft und angepasst. Nach Überarbeitung wurde der zweite Designentwurf zum App-Design erkoren und umgesetzt. Der erste Design-Entwurf wurde mit dem Arbeitstitel „Entwurf_X“ versehen und in der nachfolgenden Dokumentation werden Ansichten dieses Entwurfs mit einem „X_“ gekennzeichnet.

Die entworfenen Wireframes dienen der genauen Anordnung von Elementen und der Benutzerführung unserer App „BambiGuard“. Die Wireframes nehmen keinen Bezug auf die letztendliche Farbgestaltung.

Landing Screen

Der Landing Screen wurde nach dem Redesign weitestgehend geändert. So wird er zwar noch die entscheidende Ankunftsplattform bleiben, jedoch nur mehr als Drehscheibe zwischen den beiden Ansichten fungieren.

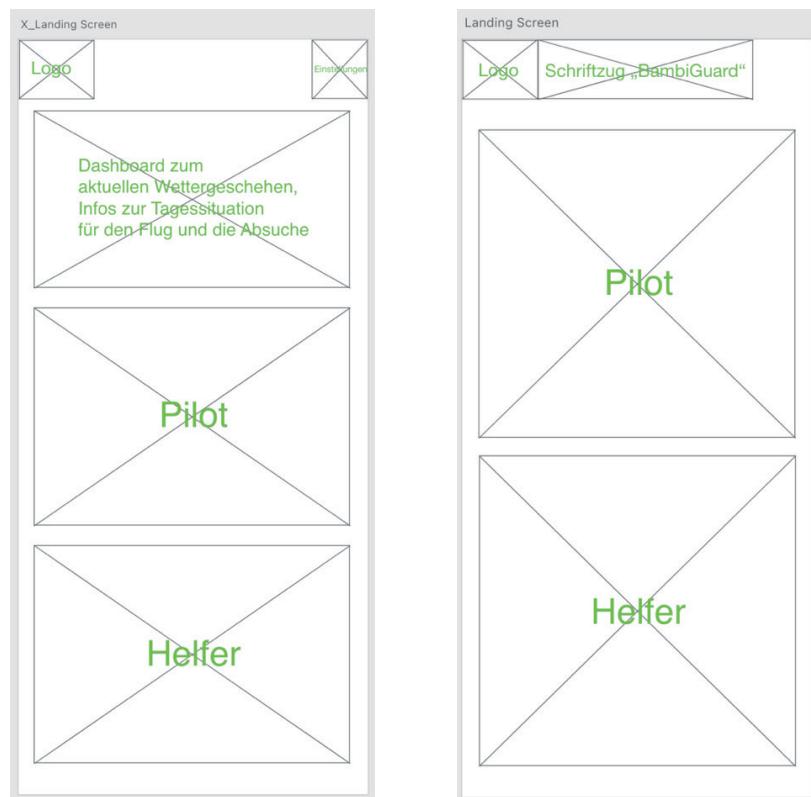


Abbildung 36: Landing Screen (Wireframe)

Im ersten Entwurf wurde auf das Informationsupdate großen Wert gelegt, jedoch werden gedachte Informationen überhaupt nicht gebraucht oder ohnedies schon wahrgenommen. Am besten wird diese Erkenntnis mit den geplanten Parametern „aktueller Wettergeschehen“ sowie „Datum“ aufgezeigt. Befindet sich ein Team bereits auf dem Feld der Absuche, so wird ihnen schnell klar werden, wie die Wetterverhältnisse sind und können selbst feststellen, ob Regen oder Sonnenschein vorherrscht. Durch den Parameter „Datum und Uhrzeit“ könnte ein Informationsüberfluss stattfinden, da diese Information den Nutzern bereits vom Endgerät mitgeteilt und angezeigt wird, dies würde die Überschaubarkeit einschränken. Eine Komponente, welche anfangs geplant wurde, war der Einstellungsreich oben rechts, wobei in der App kein Einstellungsbutton benötigt wird.

Pilot – Flugplanung

Wie nach Erstellung des ersten Wireframe vermutet, wurde der Platzbedarf auf der Flugplanungs-Ansicht zu klein und so wird diese im zweiten Entwurf auf zwei Ansichten aufgeteilt. Durch diese Aufteilung wurde zum einen mehr Platz für die bestehenden Komponenten (größere Bedienelemente) und zum anderen eine Unterteilung der Arbeitsprozesse erreicht. Dadurch wird der Benutzer nicht überfordert.

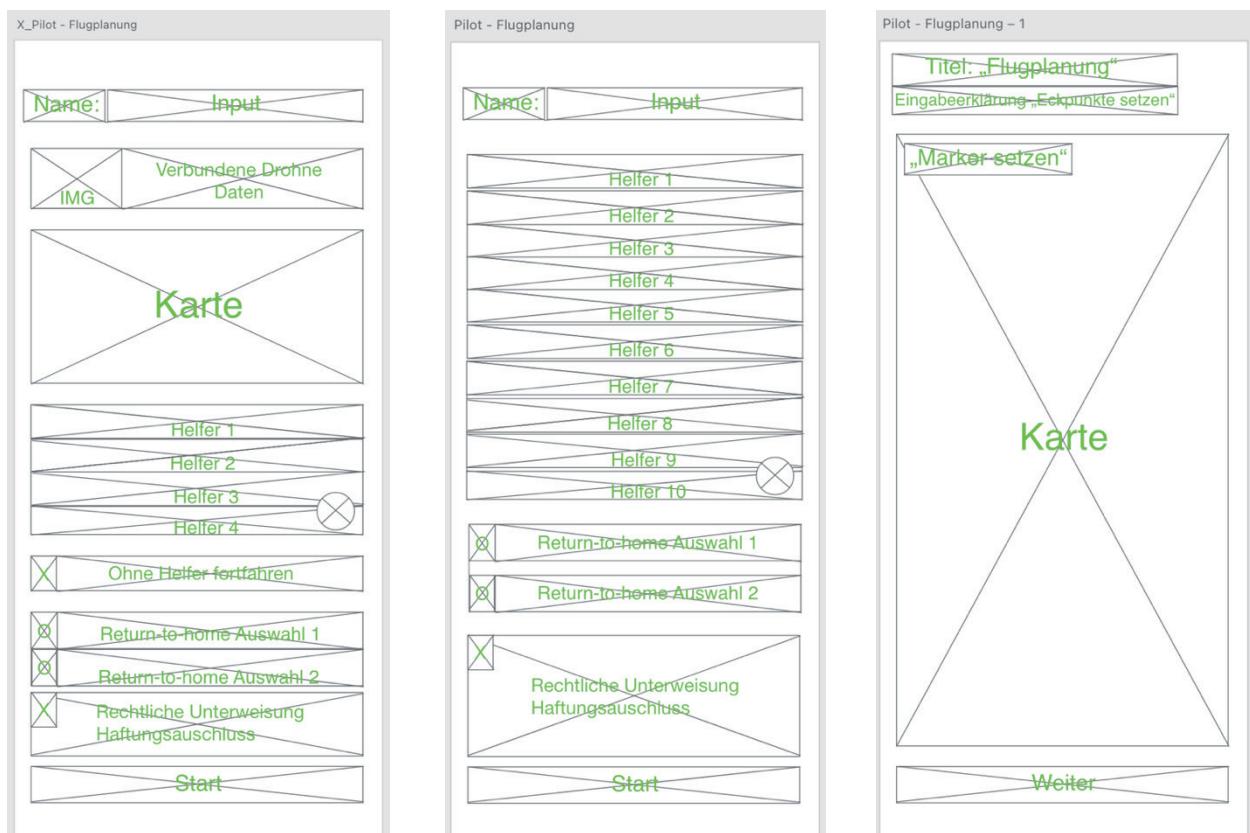


Abbildung 37: Pilot – Flugplanung (Wireframe)

Aus dem ersten Entwurf nicht mehr übertragen wurde die Ansicht des Drohnenmodells. Dies wird in einer abgeänderten Form in der Ansicht „Pilot – im Flug“ integriert. Ebenso nicht mehr Teil des zweiten Entwurfs ist die Checkbox „Ohne Helper fortfahren“. Für diese Form der Rettung wurde unsere App „BambiGuard“ nicht entwickelt und würde keinen Mehrwert für die App bedeuten. Mit der Ansicht „Pilot – Flugplanung – 1“ wird eine Erleichterung der Eckpunktauswahl in der App möglich. Im ersten Entwurf wurde der Platz für diese Funktion zu klein. Die Registrierung des Piloten sowie das Hinzufügen der Helper wird in der zweiten Ansicht „Pilot-Flugplanung“ ermöglicht als auch die Beendigungsmöglichkeiten des Fluges und die rechtliche Aufklärung und Bestätigung des Piloten.

Pilot – im Flug

Ebenso wie der Landing Screen wurde auch die Ansicht „Pilot – im Flug“ umstrukturiert. Basis für die neue Aufstellung bildet die gespaltene Kameraansicht, wodurch sowohl das Bild der Kamera als auch die Karte zu sehen ist. Die oberste Leiste am Screen wurde weitestgehend vom Hersteller DJI übernommen, nur die Einbindung des Logos wurde designtechnisch verändert. Der Grundgedanke der Übersicht bleibt gleich, daher wird auf dem finalen Entwurf links eine Übersicht platziert. Die Flugübersicht und die Helper-Statistik werden aufgrund von Ablenkungsgründen auf die Endübersicht verschoben. Bevor eine Aktion in der Ansicht gestartet wird, muss die Drohne angeschlossen werden. Ein Pop-Up des Aufrufs weist auf die Konnektivität des Gerätes und das Drohnenmodell selbst hin.

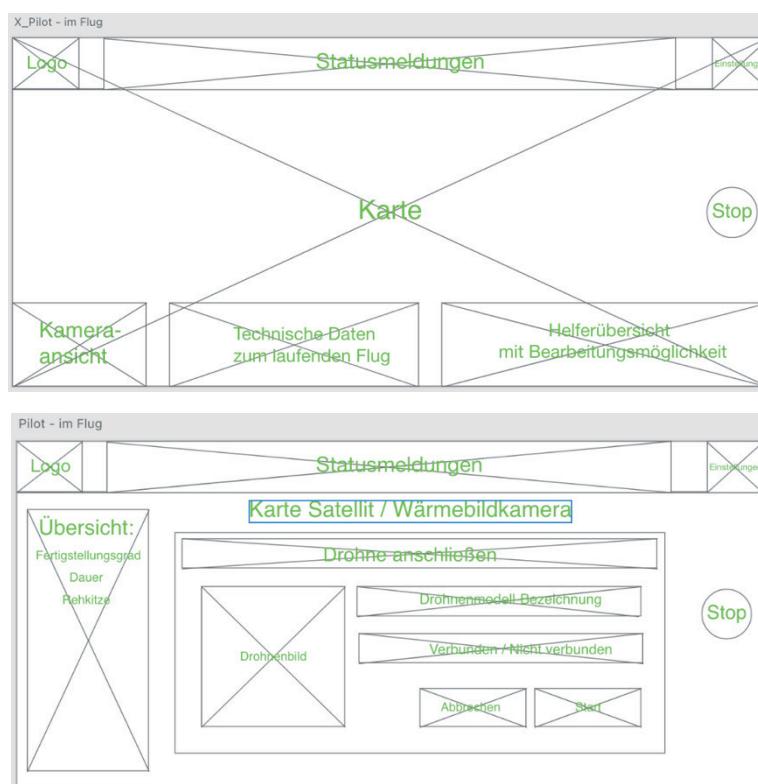


Abbildung 38: Pilot – im Flug (Wireframe)

Pilot – Ende Flug

Unverändert nach dem Redesign des App-Layouts blieb die „Pilot -Ende“- Ansicht. Die Sektionen Flugübersicht, gefundene Rehkitze, sowie die Helfer-Statistik blieben im finalen Entwurf. Grundgedanke hierbei ist, die Ergebnisse des Fluges kompakt zusammenzufassen. Der Pilot soll nach dem Flug genau Bescheid wissen, ob der Flug erfolgreich war oder nicht. Dies ermöglicht ihm im Bedarfsfall auch eine erneute Absuche für sinnvoll oder notwendig zu erachten.

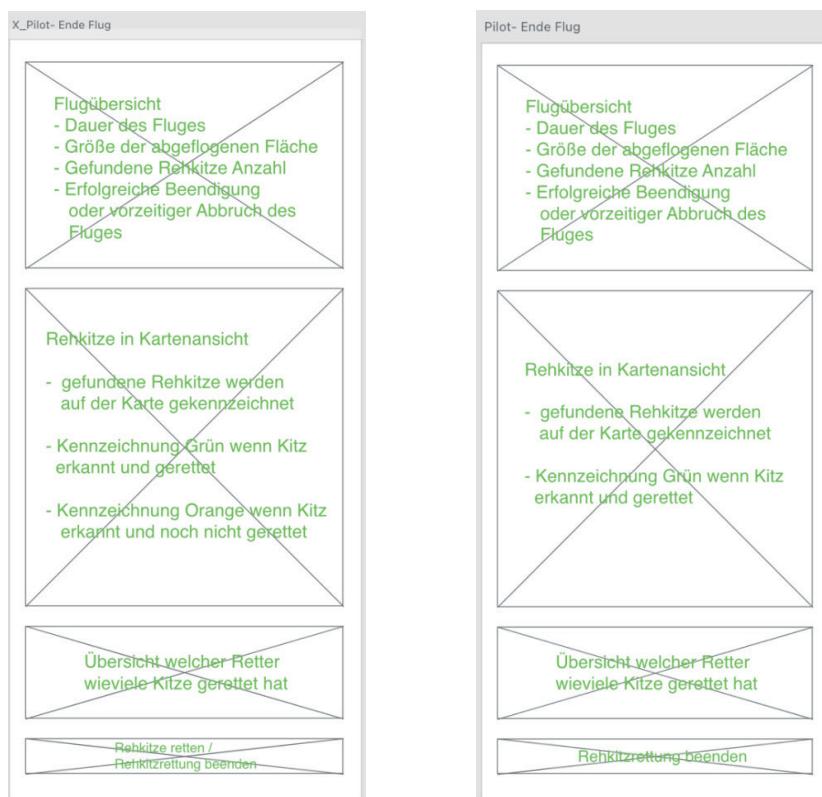


Abbildung 39: Pilot – Ende Flug

Pilot – Einzelrettung

Die Ansicht „Pilot – Einzelrettung“ ist von Beginn an, als Zusatzziel gesteckt worden. Die Grundidee war es, nicht nur eine Rehkitzrettung mit mehreren Helfern zu entwickeln, sondern auch eine Einzelrettung nur als Pilot durchzuführen. Die Einzelrettung bietet die gleichen Funktionen und Oberfläche wie in der „Helfer – Im Flug“ - Ansicht. Die „Pilot -Einzelrettung“ wurde nach Teamberatungen aus dem App-Ablauf genommen, da diese Ansicht nicht mit unseren erzielten Vorteilen zusammenpasst.

Helper – Flug beitreten

Der erste Entwurf der „Helper – Flug beitreten“-Ansicht war mit sehr vielen Informationen überflutet. Auf kleineren Bildschirmen könnten die Elementblöcke zu klein dargestellt werden, daher wurde die Ansicht im zweiten Entwurf auf die wichtigsten Elemente reduziert. Im Gegensatz zum ersten wird beim überarbeiteten Wireframe der Hauptfokus auf die Grundfunktion, das Tutorial, gelegt. Die Bereiche „Infotext“, „Anleitung“ und „Sicherheitshinweise“ wurden in abgeänderter Form in den „Tutorialtext“ integriert. Durch das Zusammenfassen dieser Inhaltsbereiche wurde für das „Tutorialbild“ und den „Tutorialtext“ mehr Platz auf der Ansicht frei. Der Registrierbereich im unteren Bereich der Oberfläche bleibt unverändert Teil der Ansicht.

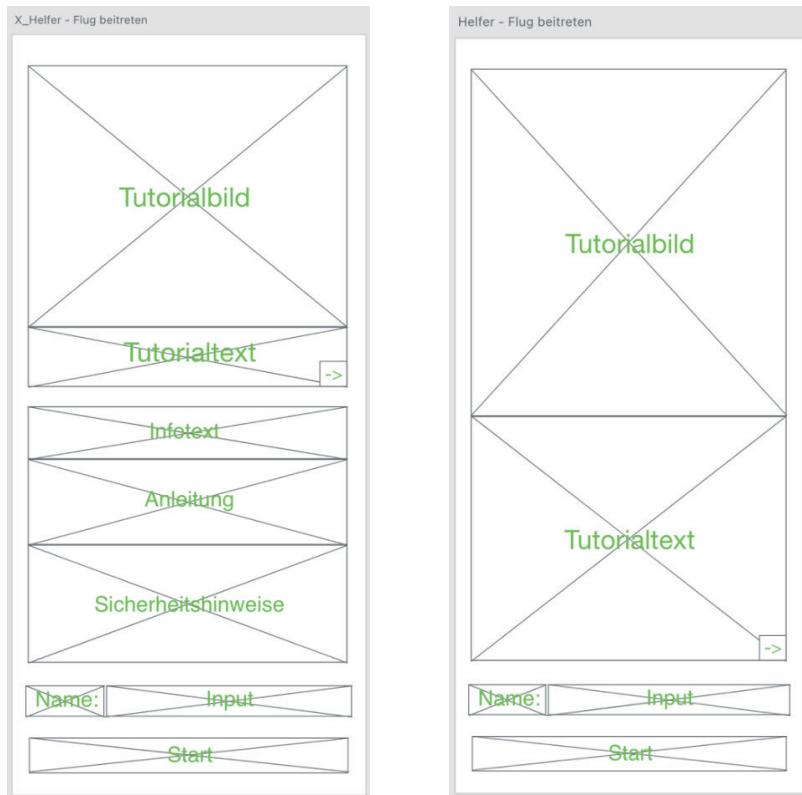


Abbildung 40: Helper – Flug beitreten (Wireframe)

Helper – im Flug

In der Ansicht „Helper – im Flug“ liegt der Hauptfokus auf der Navigierfunktion zu den Rehkitzen. Im ersten Entwurf war die Übersicht der erledigten und offenen Anträge noch Teil des Entwurfs, im Überarbeitungsprozess konzentrierte man sich eher auf die Hauptfunktion und gab diesen Symbolen mehr Platz. Ebenso wurde die Drohnenposition aus dem Entwurf entfernt, da es in der Anwendung nur für Verwirrung sorgt und vom Hauptkompass ablenkt.

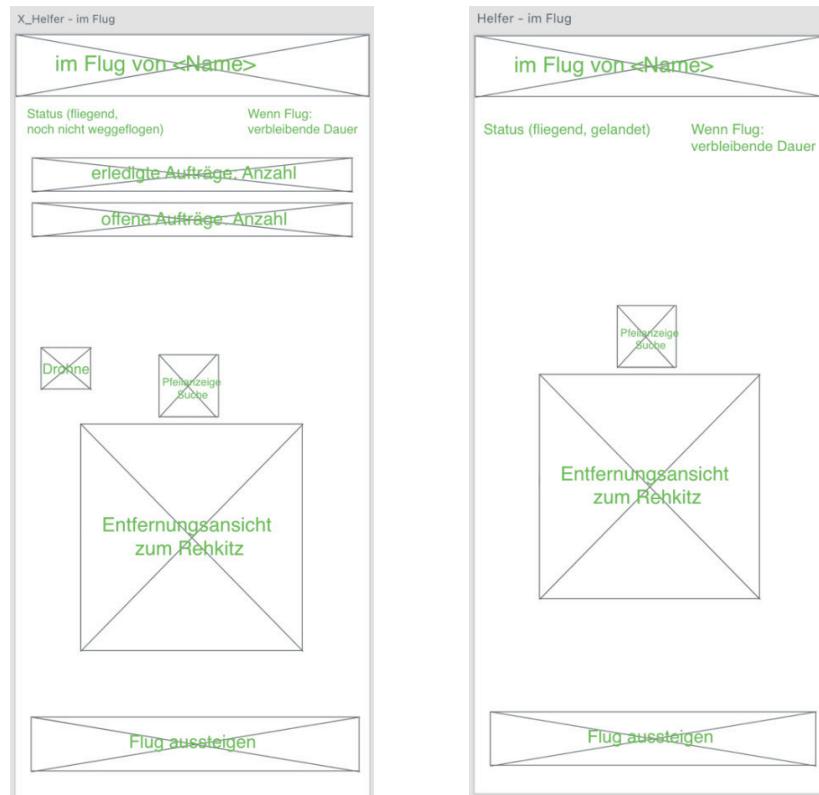


Abbildung 41: Helfer – im Flug (Wireframe)

3.3.3 Mockup und Prototyp

Landing Screen

Im Mockup-Entwurf des Landing Screens kann man auf den ersten Blick erkennen, dass es sich um zwei unterschiedliche Rollen handelt, die zur Auswahl stehen. Das Entfernen der umgebungs und witterungsbedingten Daten aus dem Erstentwurf sorgt für mehr Klarheit bei der Auswahl für Pilot und Helfer. Die entfernten Informationen sind außerdem weder für den Piloten noch für die Helfer relevant, da sie sich beim Starten der App schon im Einsatzgebiet befinden und das Wetter selbst beobachten können. Weitere Aspekte sind die Lesbarkeit und die Klarheit der Touch-Targets. Eine Auswahl ist nun auch bei ungenauerer Auswahl zieltreffend möglich.

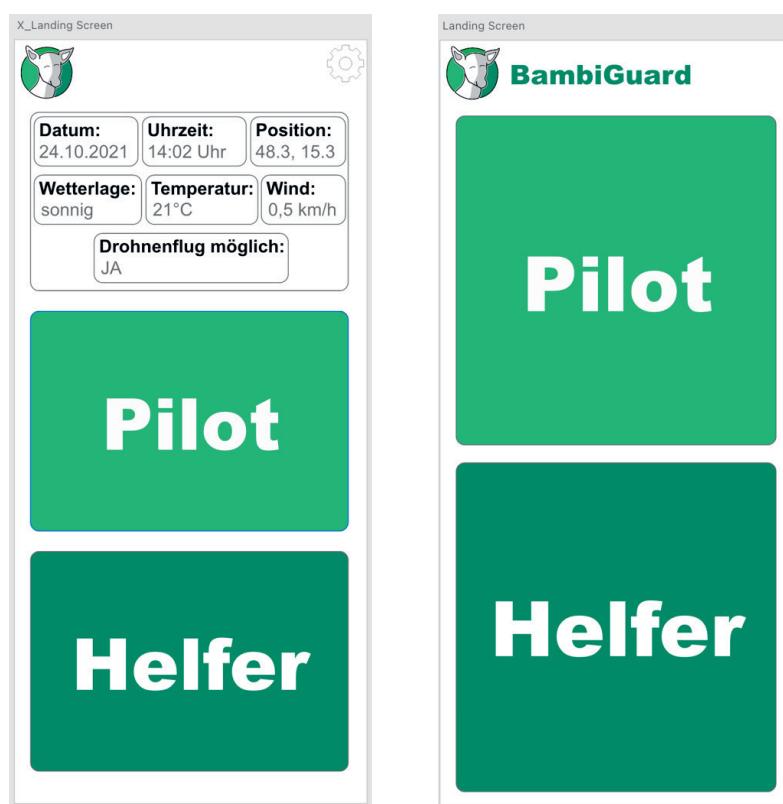


Abbildung 42: Landing Screen (MockUp)

Pilot – Flugplanung

Gerade im grafischen Design sieht man die Unterschiede der beiden Entwürfe sehr deutlich. Im „Entwurf X“ kann man erkennen, dass die Bereiche sehr dicht ohne große Zwischenräume aneinander gereiht sind. Ein weiteres Argument für die Aufteilung der beiden Ansichten ist die zu kleine Darstellung der Karte im ersten Entwurf. Ein großer Vorteil der Umstrukturierung ist die größere Kartenansicht sowie die Überschaubarkeit und die größeren Zwischenräume zwischen den Bereichen.

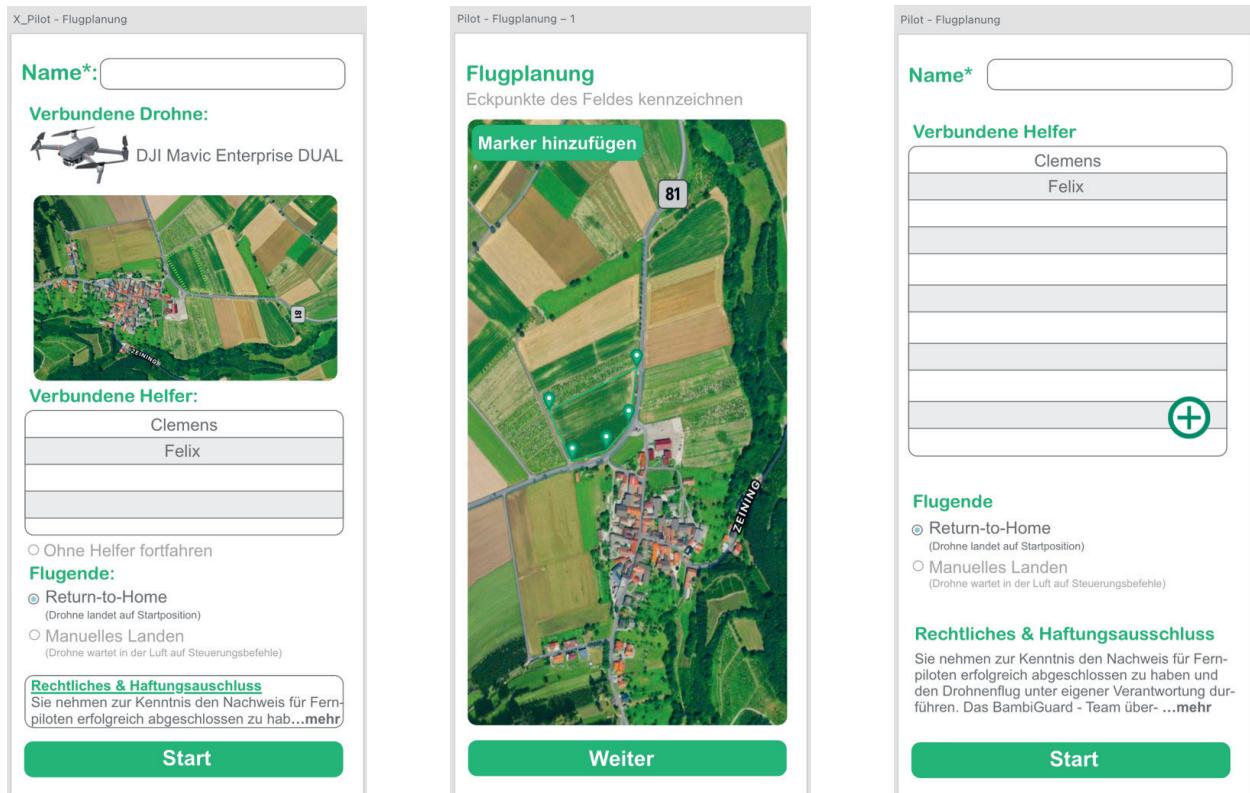


Abbildung 43: Pilot – Flugplanung (MockUp)

Pilot – im Flug

In der Ansicht „Pilot – im Flug“ kann man erkennen, dass sich die beiden weißen Blöcke in den Vordergrund drängen. Im ersten Entwurf hätte man den Flugablauf grafisch nicht so gut verfolgen können. Durch den weißen Seitenbereich im zweiten Entwurf behält der Pilot (durch den Kontrast zu den farbreichen Ansichten) den Ablauf immer im Blick. Ebenso ist im zweiten Entwurf nicht nur die Karte mit der Position der Drohne, sondern auch das Bild der Wärmebildkamera zu sehen.

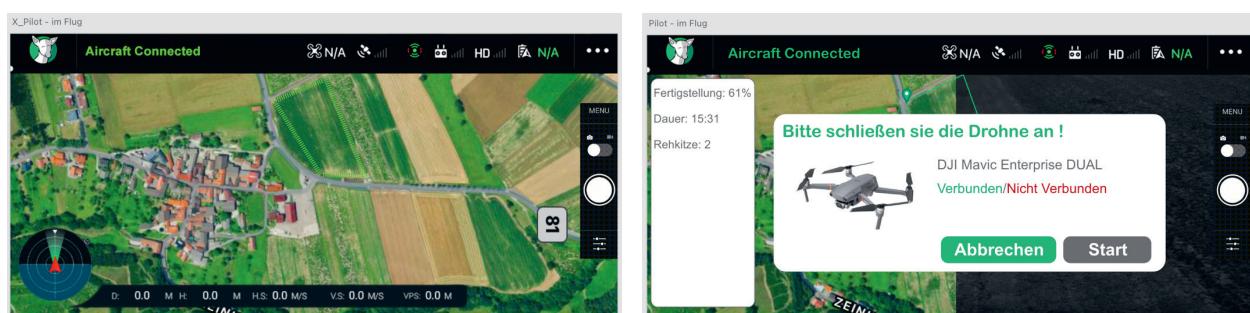


Abbildung 44: Pilot – im Flug (MockUp)

Pilot – Ende Flug

Bei der Ansicht „Pilot – Ende Flug“ hat sich zum Erstentwurf wenig geändert. Von Beginn der Konstruktion an wollte man immer eine Übersicht über den gesamten Ablauf bieten und alle Fäden zusammenfließen lassen. Die Endansicht ist die durchdachteste Ansicht des gesamten Ablaufs. Lediglich der Parameter „zu suchende Rehkitze“ wurde aufgrund eines geänderten Verlaufs entfernt.

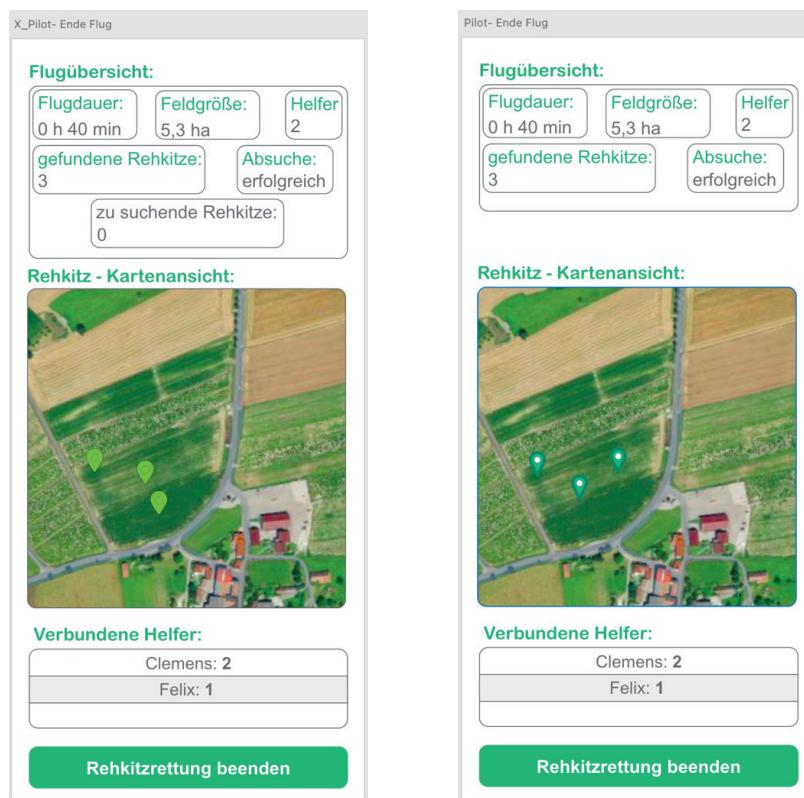


Abbildung 45: Pilot – Ende Flug (MockUp)

Helper – Flug beitreten

Klar ersichtlich an der dunkleren Farbe findet man sich in der Rolle des Helfers. Das Icon unten rechts auf der Benutzeroberfläche gibt dem Nutzer ein deutliches Zeichen und die Aufforderung, zu warten. Beim Entwurf_X war die Ansicht von Informationen überladen. Die Ansicht ist in Abbildung 46 ersichtlicht.

Helper – im Flug

Nicht nur an der dunklen Farbe kann man erkennen, dass man sich als Helper in der App angemeldet hat, sondern auch an der Überschrift „im Flug von Maximilian“ und am Button „aus Flug aussteigen“. Durch die entfernten Aufträge-Parameter liegt auf den beiden verbliebenen Parametern „Status“ und „Verbleibende Flugdauer“ eine erhöhte Aufmerksamkeit. Abbildung 47 zeigt den Screen „Helper – im Flug“.

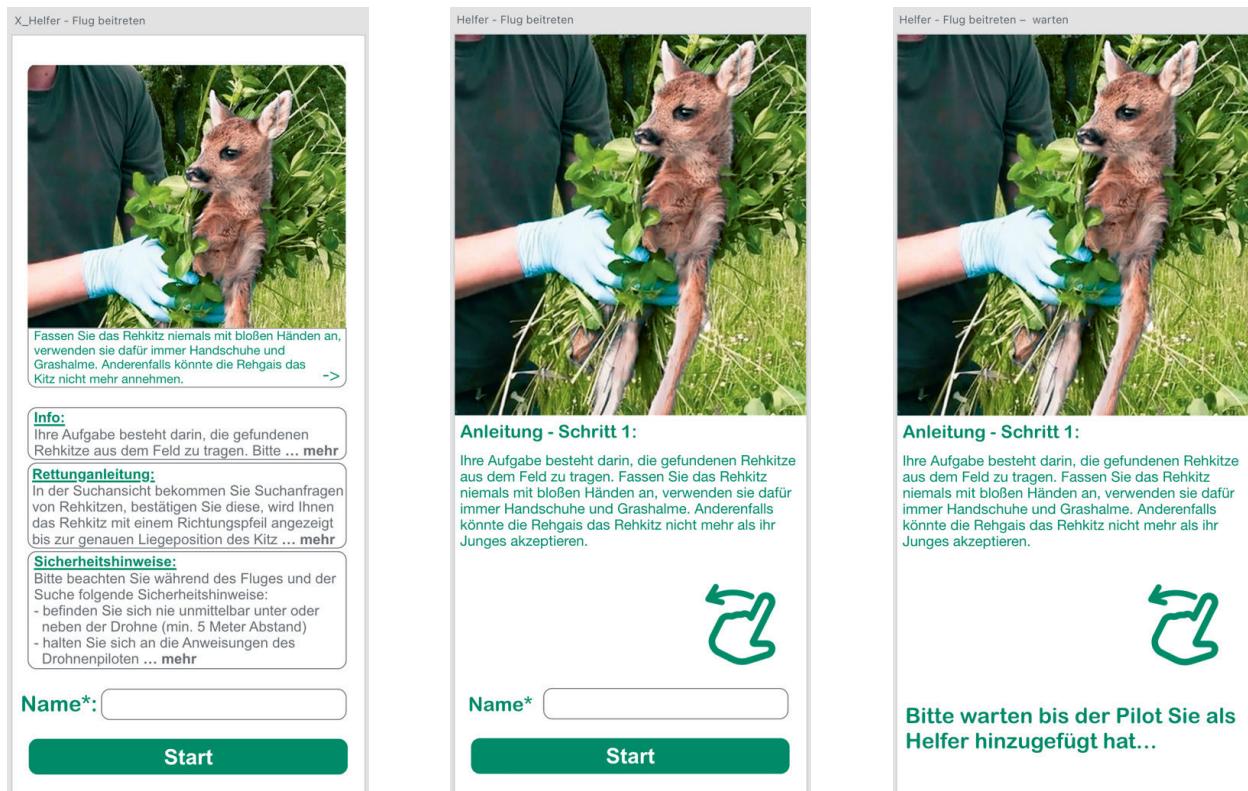


Abbildung 46: Helper – Flug beitreten (MockUp)



Abbildung 47: Helper – im Flug (MockUp)

Gesamter App-Ablauf

Der gesamte Ablauf der App teilt sich auf acht Ansichten. Fünf davon für den Piloten und vier für den Helfer. Der Landing Screen ist für beide die erste Ansicht. Durch die beiden unterschiedlichen Farben der beiden Rollen (Pilot und Helfer), kann man sich orientieren, in welcher man sich momentan befindet.

Der Pilot gelangt mit dem Klick auf den Auswahlbutton „Pilot“ im Landing Screen direkt in die erste Flugplanungsansicht. Hier muss er die Eckpunkte des Feldes kennzeichnen und kann erst nach der Auswahl in die zweite Flugplanungsansicht gehen. In der zweiten Ansicht trägt der Pilot zuerst seinen Namen ein und kann danach sofort seine Helfer seines geplanten Fluges hinzufügen. Zum Schluss der Planung muss noch die Aktion am Flugende festgelegt werden. Wenn die Drohne bereits mit dem Controller verbunden ist, kann sofort mit der Suche begonnen werden. Ist dies noch nicht passiert, muss man spätestens zu diesem Zeitpunkt das Smartphone mit dem Controller verbinden. Ist die Absuche fertig, bekommt der Pilot eine Zusammenfassung in der letzten Ansicht.

Wird der „Helfer“-Auswahlbutton geklickt, gelangt der Nutzer in ein Erklärtutorial zur Absuche. Fügt der Pilot den Helfer erfolgreich hinzu, gelangt dieser in die Navigierfunktionsansicht, wo er durch Mitteilungen Aufträge zur Rehkitzrettung erhält. Bei Beendigung der „Helfer – im Flug“-Ansicht gelangt der Helfer wieder zurück zur Eingabe des Namens und des Tutorials um doch notwendige Infos zu bekommen. Der gesamte Prototyp kann unter folgendem Link im Web oder auf dem eigenen Smartphone durchgeklickt werden: <https://xd.adobe.com/view/2b985c39-94ae-4f51-9b15-a07b5fa8dc58-7ecb/?fullscreen>



Abbildung 48: Gesamter App-Ablauf in Adobe XD

3.4 Bilderkennung

BambiGuard soll gemäß den Anforderungen Rehkitze, welche in einem mit Gras bewachsenen Feld liegen, durch ein Wärmebild aus einigen Metern Höhe automatisiert erkennen. Durch die Recherche im Zuge des Kapitels Grundlagen und Methoden wurden unterschiedliche Arten der Umsetzung dieser Bilderkennung analysiert. Es wurde der Einsatz der Grafikbibliothek OpenCV gewählt.

Die Entwicklung des Algorithmus zur Bilderkennung findet vor der Programmierung der App statt. Vom Standpunkt der Planung aus ist dies ein kritischer Teil des Projektes, weil diese Methoden (Bilderkennung) von den Projektmitgliedern noch nie in einem vergleichbaren Umfang umgesetzt wurden. Durch die Vielseitigkeit von OpenCV kann die gesamte Entwicklung der Bilderkennung separat zur App stattfinden und einzeln getestet werden. Der Algorithmus ist in C++ geschrieben und kann deswegen genauso in eine Desktop-Anwendung, wie in eine iOS oder Android App eingebunden werden.

3.4.1 Schritte des Algorithmus

Um die Vorgehensweise effizient zu gestalten, sind die Schritte, welche zur funktionierenden Erkennung der Wildtiere führen, zuerst in der Theorie durchdacht worden.

Das Programm erhält als Eingabe eine Pixelgrafik in Graustufen. Dunkle Flächen sind als niedrige Temperaturen und helle Flächen als hohe Temperaturen zu interpretieren. Die genauen Temperaturen können mithilfe der Werte der Kalibrierung der Wärmekamera bestimmt werden, diese sind jedoch für die Funktion des Algorithmus irrelevant (der Algorithmus kann an die Kalibrierung der Kamera angepasst werden). Die Ausgabe des Algorithmus ist eine Liste von Punkten (X- und Y-Koordinate in Pixel, abhängig von der Bildmitte als Position der Kamera), welche die Positionen der Rehkitze im Bild repräsentieren. Diese Punkte können dann mithilfe des Bildwinkels, der Flughöhe und der Koordinaten der Drohne auf die Koordinaten der Rehkitze schließen. Das Umwandeln der Koordinaten in Längen- und Breitengrade wird in Java bzw. Swift erfolgen.

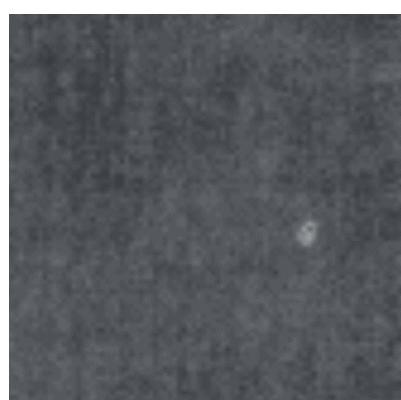


Abbildung 49: Graustufen Wärmebild mit platziertem Hund

Folgende Schritte sind für die Bilderkennung geplant:

1. Weichzeichnung

Eine Gauß'sche Unschärfe vernichtet Störfaktoren, wie kleine reflektierende Flächen, welche sich als einzelne helle Pixel im Bild erkennbar machen.

2. Schwellenwertfindung

Graustufenpixel werden mit Thresholding zwei Zuständen zugewiesen.

3. Morphologische Transformationen

Die Kombination aus morphologischem Öffnen und Schließen zerstört nach dem Schwellenwert die übrig gebliebenen Störpixel. Warme Flächen, welche zu klein sind, um als Rehkitz zu gelten, werden durch diesen Schritt aussortiert.

4. Konturen ermitteln

Im Binärbild werden die Konturen als Menge aus Punkten erkannt.

5. Überprüfung der Konturen

Die Größe und Umgebung der Konturen werden überprüft und nach Erfahrungswerten auf Validität kontrolliert.

6. Mittelpunkte errechnen

Der Durchschnitt der Punkte einer Kontur ergibt die Position des Rehkitzes.

3.4.2 Einrichtung der Entwicklungsumgebung

Visual Studio dient als Entwicklungsumgebung für die Programmierung der reinen Bilderkennung. Damit können Build-Prozesse automatisiert und das Projekt kompakt verwaltet werden. Es wird Visual Studio 2019, Version 16.11 mit OpenCV 4.5.3 verwendet.

Nach der Erstellung eines neuen Projekts in Visual Studio müssen die Include-Pfade um die Bibliothek OpenCV erweitert werden. In den Eigenschaften des Projektes unter C/C++ / Allgemein wird bei „Zusätzliche Includeverzeichnisse“ der Pfad angegeben, in dem OpenCV installiert ist.

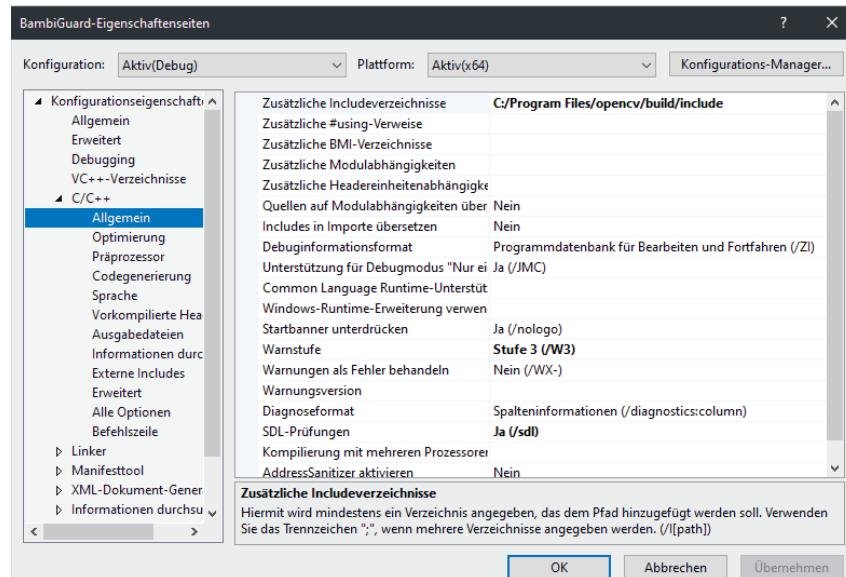


Abbildung 50: Zusätzliche Includeverzeichnisse in Visual Studio

In den Einstellungen unter Linker/Eingabe muss aus dem lib-Verzeichnis von OpenCV die Datei `opencv_world453.lib` als „Zusätzliche Abhängigkeit“ hinzugefügt werden.

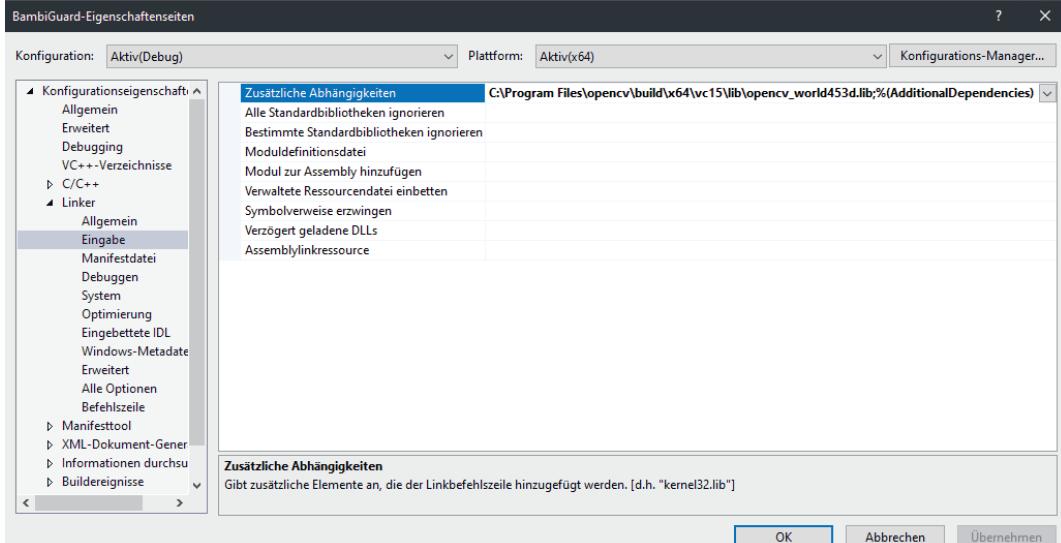


Abbildung 51: Zusätzliche Abhängigkeiten in Visual Studio

Mit dieser Konfiguration kann OpenCV durch die Statements in Programmausdruck 4 in ein Programm eingebunden werden.

```
#include <opencv2/opencv.hpp>
using namespace cv;
```

Programmausdruck 4: Inkludierung der C++ Bibliothek OpenCV

3.4.3 Programmierung

Es wird eine header-Datei erstellt, welche zum Debugging in Visual Studio, sowie in Android und iOS eingebunden werden kann. Sie beinhaltet hauptsächlich die Klasse BambiGuardRecognition, welche mit der Flughöhe (double) des aufgenommenen Bildes initialisiert wird. Die Funktion detectBambisInImage bekommt eine OpenCV Mat als Eingabe und ist der Einstiegs-punkt des Algorithmus. Bei der Verwaltung des Bildes ist zu beachten, dass in C stets Pointer zu den Objekten weitergegeben werden und daher eine Bearbeitung in einer Funktion außerhalb deren Ebene wirksam ist.

```
vector<Point> detectBambisInImage(Mat image)
{
    // Verarbeitung der Grafik
    Mat imageProcessed = processImage(image);

    // Erkennen der Konturen
    vector<vector<Point>> contours = findContoursInImage(
        imageProcessed);
    vector<Point> contourCenters;

    // Überprüfen der Konturen
    for (int i = 0; i < contours.size(); i++)
    {
        if (!isContourAreaInRange(contours[i]) ||
            !isEnclosingCircleAreaInRange(contours[i]) ||
            !isContourSurroundedByGrass(contours[i], image))
            continue;

        // Berechnung der Mittelpunkte
        contourCenters.push_back(getContourCenter(contours[i]));
    }

    eliminateClosePoints(contourCenters);

    // Berechnung der Position
    for (Point point : contourCenters)
        point = getPointRelativeToImageCenterInMeter(point);

    return contourCenters;
}
```

Programmausdruck 5: Funktion detectBambisInImage aus der Datei BambiGuard.h

Bildverarbeitung

Die Verarbeitung erfolgt wie beschrieben mit eingebauten Funktionen von OpenCV. Die Kernelgröße des Gauß'schen Filters ist mit dem Wert 5x5 Pixel gesetzt, der Kernel der morphologischen Transformation beträgt 10x10 Pixel. Der mittlere Schwellenwert 127 erzielte beim Testen mit dem Videomaterial das beste Ergebnis. Dieser Schwellenwert müsste bei einer abweichenden Kalibrierung der Wärmebildkamera geändert werden.

```
GaussianBlur(image, image, Size(5, 5), 0);
threshold(image, image, thresholdNumber, 255, THRESH_BINARY);
morphologyEx(image, image, MORPH_OPEN, kernel);
```

Programmausdruck 6: Bildbearbeitende Funktinoen aus der Datei BambiGuard.h

Konturenfindung

Die Funktion `findContours` liefert eine Liste von Punkten mit X- und Y-Koordinaten zurück.

```
findContours(image, contours, RETR_TREE, CHAIN_APPROX_NONE);
```

Programmausdruck 7: Konturenfindung in der Datei BambiGuard.h

Kontrolle

Zur Validierung der Ergebnisse werden mehrere Kontrollen durchgeführt. Eine erkannte Kontur muss eine Weite im Bereich 0,1 bis 2 Metern haben. Dafür wird der kleinste umschließende Kreis (englisch „*enclosing Circle*“) als Rechengrundlage genommen (`isContourAreaInRange`). Diese definierten Werte werden auch verwendet, um die Fläche des Bereiches zu überprüfen (`isEnclosingCircleAreaInRange`). Die letzte Funktion (`isContourSurroundedByGrass`) überprüft, ob die unmittelbare Umgebung der Kontur hell oder dunkel ist. Sie lässt nur als solche erkannte im Gras liegende Tiere durch den Filter. Programmausdruck 8 zeigt diese drei Funktionen zur Kontrolle der erkannten Konturen.

```

bool isContourAreaInRange(vector<Point> contour)
{
    double area = contourArea(contour);
    return (area > minimalBambiArea || area < maximalBambiArea);
}

bool isEnclosingCircleAreaInRange(vector<Point> contour)
{
    float radius = 0.0;
    minEnclosingCircle(contour, center, radius);
    return (radius * 2 > MINIMAL_BAMBI_WIDTH) ||
           (radius * 2 < MINIMAL_BAMBI_WIDTH *
            FACTOR_TO_MAX_BAMBI_WIDTH);
}

bool isContourSurroundedByGrass(vector<Point> contour, Mat image)
{
    Point2f center;
    float radius = 0.0;
    minEnclosingCircle(contour, center, radius);

    float surroundingRadius = radius + RADIUS_ARROUND_BAMBIS *
        pixelsPerMeter;
    float averageValue = getAverageValueOfCircleSegment(
        center, radius, surroundingRadius, image);

    return averageValue < MAXIMAL_SURROUNDING_AVERAGE_VALUE;
}

```

Programmausdruck 8: Kontrolle der Erkennung in der Datei BambiGuard.h

3.4.4 Photogrammetrie

Die Berechnung der Vergleichswerte für diese Überprüfungen und der Position der Rehkitze im Bild ist Teil der Photogrammetrie (auch Bildmessung). Basierend auf Angaben über die Spezifikationen der Kamera und deren Lage kann auf die Position eines bestimmten Bildpunktes in der echten Welt geschlossen werden.

Die Kamera besitzt einen horizontalen Bildwinkel von 57° und nimmt mit einer Bildgröße von 640x360 Pixeln auf. Abbildung 52 und Abbildung 53 zeigen Skizzen der Zusammenhänge zwischen den Spezifikationen der Kamera und des Bildausschnitts.

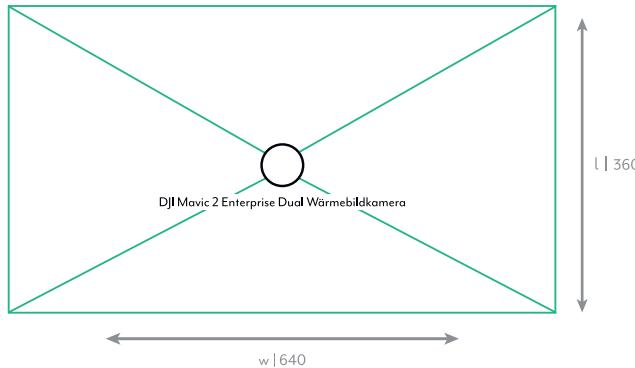


Abbildung 52: Bildbereich der Wärmebildkamera
(nicht maßstabsgetreu)

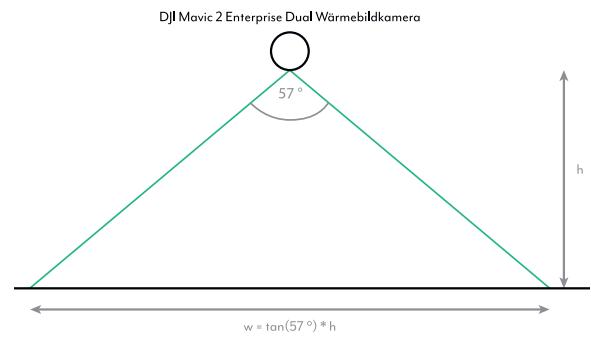


Abbildung 53: Bildwinkel der Wärmebildkamera
(nicht maßstabsgetreu)

Zur korrekten Darstellung wurden mit der Software GeoGebra Modelle der Kamera erstellt.

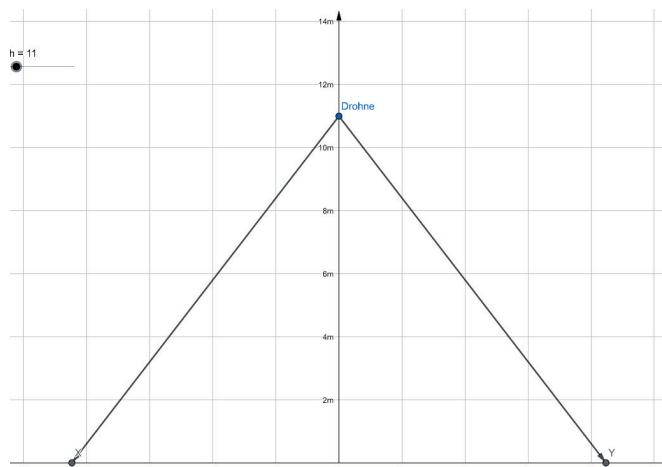


Abbildung 54: Bildbreite bei einer Flughöhe von 11m

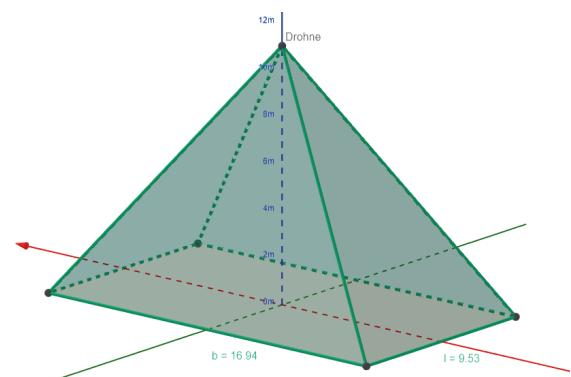


Abbildung 55: Bildausschnitt bei einer Flughöhe von 11m

Ausgehend von diesen Modellen kann aus einer Länge in Metern im Feld dessen äquivalente Distanz im Bild in Pixeln errechnet werden.

```
imageWidth = tan(HORIZONTAL_FIELD_OF_VIEW) * flightHeight;
imageHeight = imageWidth * (VERTICAL_PIXELS / HORIZONTAL_PIXELS);
pixelsPerMeter = HORIZONTAL_PIXELS / imageWidth;
```

Programmausdruck 9: Berechnung des Pixel-zu-Meter Quotienten in BambiGuard.h

3.4.5 Debugging und Testen

Für das Entwickeln des Algorithmus werden mit der Drohne aufgenommene Einzelbilder nach Bedarf präpariert, um bestimmte Situationen nachzustellen. Zum Testen werden auch MP4-Testvideos verwendet.

```
Mat image = imread(photoPath);
cvtColor(image, image, COLOR_BGR2GRAY);
```

Programmausdruck 10: Einlesen und Konvertieren in BambiGuard.cpp



Abbildung 56: Präpariertes Wärmebild mit zwei eindeutigen Tieren

Ausgabe

Das Bild wird nach Teilschritten des Algorithmus immer wieder ausgegeben, um die Funktionalität zu testen. Dazu werden folgende Funktionen von OpenCV verwendet.

```
circle(image, center, radius, (255, 255, 255), 1);
drawContours(image, contours, -1, Scalar(255, 255, 255), 3);
imshow("BambiGuard", image);
```

Programmausdruck 11: Verschiedene Funktionen zur Ausgabe eines Bildes; BambiGuard.h



Abbildung 57: Einzeichnung eines detektierten Rehkitzes

3.5 App für Android

3.5.1 Entwicklungsumgebung Android Studio

Android Studio

Für die Entwicklung der Android App wird die IDE Android Studio verwendet, welche wichtige Features wie einen Layout-Editor und ADB-Support mitbringt. Sie ist frei zum Download verfügbar. Eine Applikation kann entweder im Emulator oder mithilfe der *Android Debug Bridge* (ADB) auf einem Smartphone ausgeführt werden. Bei der Entwicklung von BambiGuard wird hauptsächlich letztere Variante verwendet, wobei das Gerät entweder via USB oder Wifi verbunden wird. Dafür müssen auf dem Smartphone die Entwickleroptionen aktiviert sein, sowie USB- oder WIFI-Debugging zugelassen werden.



Abbildung 58: USB-Debugging am Smartphone zulassen

Bei Android-Geräten mit der Android 10 oder älteren Versionen muss Wifi-Debugging mithilfe der adb-tools aktiviert werden. Dieses zusätzliche Softwarepacket kann auf der Website von Android heruntergeladen werden. Nach Anschließen des Smartphones per USB setzt man folgende Befehle in der Kommandozeile ab, um das Smartphone per TCP mit der Android Debug Bridge zu verbinden.

```
adb tcpip 5555  
adb connect smartphone_ip_adresse:5555
```

Versionskontrolle

Der Android-Programmcode wird auf dem Gitlab-Server der Schule versioniert und gespeichert. In dem Repository befinden sich neben der BambiGuard-App auch wichtige Demoprojekte und Beispiel-Apps. Um nur die notwendigen Dateien zu synchronisieren, wird eine `gitignore`-Datei eingerichtet. Diese schließt unter anderem alle kompilierten Dateien und lokalen Konfigurationen aus, sowie `gradle`-Dateien und das gesamte `build`-Verzeichnis.

```
...  
# Built application files  
*.apk  
*.aar  
*.ap_  
*.aab  
...
```

Programmausdruck 12: Auszug aus der Datei `.gitignore` für das Android-Projekt

3.5.2 Informationshierarchie

Nachdem BambiGuard die Funktionen des Piloten und die des Helfers in derselben App beinhaltet, müssen unter den Activities geteilte Informationen strukturiert werden. Es wird eine Struktur mit Model und ViewModel verwendet, obwohl es keine direkte Implementierung von MVVM ist. Die Klassen `PilotViewModel` und `HelperViewModel` speichern jeweils Instanzen der Sockets (`Socket.io`) und Flug-relevante Daten (Helper: Name des Piloten; Pilot: Liste der Helfer, Flugstatus, Wegpunkte etc.). Sie sind als Singleton umgesetzt und sind von jeder Activity aus erreichbar. Dadurch können Daten aus einer Activity einfach in die nächste mitgenommen werden, ohne Bundles oder Serialisierung einsetzen zu müssen.

Die Model-Klassen beinhalten zwei Implementierungen der Sockets, den `BambiGuardCoveragePlanner` (Abschnitt Flugplanung mit Mapbox), den `BambiGuardDetector` (Abschnitt Bilderkennung und NDK) und weitere Dateien. Activities und Fragments sowie ausgelagerte Komponenten und die Klassen zur Verwaltung der DJI SDK sind logisch in eigene Packages geordnet. Dadurch ist eine Übersicht über das gesamte Projekt gegeben.

In Abbildung 59 sind alle Klassen der Android App in der Ordnerstruktur ersichtlich.

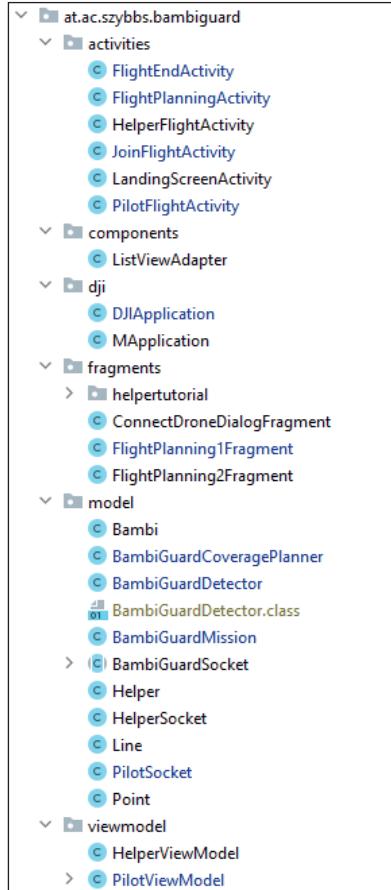


Abbildung 59: Gesamter Klassenbaum der BambiGuard Android App

3.5.3 Benutzeroberfläche

Beim Implementieren der Benutzeroberfläche hielt man sich an das Layout und die Struktur des Prototyps bzw. der Wireframes. Sie ist in XML deklariert, wodurch eine Trennung von Darstellung und Logik, wie in der Architektur von Android vorgesehen, möglich ist.

Ressourcen

Android-Apps verwenden eigene XML-Dateien für die Definition von konstanten Werten, wie Farben, Styles oder fixen Zeichenketten. In BambiGuard werden alle Farben (`colors.xml`) dem CICD Manual (siehe Abschnitt 3.8 Repräsentation) entsprechend entnommen. Dafür sind die hexadezimalen RGB-Werte in `<color>`-Tags mit entsprechenden `name`-Attributen gespeichert. Die Farben können somit mit `@color/ [name]` in anderen XML-Ressourcen identifiziert und eingesetzt werden. Programmausdruck 13 zeigt einige Definitionen von Farben in der BambiGuard-App.

```
<resources>
    ...
    <color name="colorBlack">#000000</color>
    <color name="colorPrimary">#29b17b</color>
    <color name="colorPrimaryDark">#308167</color>
    ...
</resources>
```

Programmausdruck 13: Farbdefinitionen in colors.xml

Wiederkehrende Elemente wie Buttons oder Überschriften können als style-Elemente in der Datei styles.xml angelegt werden. Jedes Element erhält, wie bei der Definition der Farben, ein identifizierendes name-Attribut und kann beliebig viele <item> Kinderelemente haben. Ein Style wird im Layout auf ein Element mit der Eigenschaft style="@style/[name]" angewendet. Auf diese Weise können konsistente Gestaltungen mit minimalem Änderungsaufwand umgesetzt werden. In BambiGuard werden beispielsweise die Buttons, Überschriften und Info-Nachrichten mit dieser Methode umgesetzt.

```
<style name="bigButton" parent="Widget.AppCompat.Button">
    <item name="android:textStyle">bold</item>
    <item name="android:textColor">@color/colorWhite</item>
    <item name="android:textSize">48sp</item>
</style>
```

Programmausdruck 14: Eine Style-Definition in styles.xml

Neben den Farben und Styles werden auch die Icons der App selbst entworfen und eingebunden. Sie sind in Adobe Illustrator als Vektorgrafiken angelegt, im Format PNG exportiert und im Ordner / drawable abgelegt. Im Vergleich zu Vektorgrafiken sparen Pixelgrafiken beim Redern zur Laufzeit Rechenleistung.

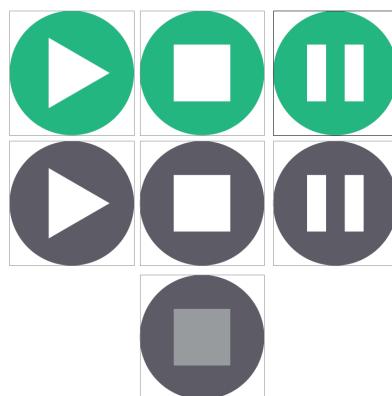


Abbildung 60: Icons zur Steuerung während des Flugs (activity_pilot_flight.xml)

Activities und Fragments

Jede Ansicht aus dem Prototyp wird in der ersten Version von BambiGuard als eigene Activity in Android umgesetzt. Die Layouts werden zum größten Teil als Code geschrieben und im Designer, dem grafischen Layout-Editor von Android Studio überprüft und evtl. in die richtige Position gebracht. Android stellt diverse Möglichkeiten zur Positionierung zur Verfügung. In BambiGuard wurde häufig das LinearLayout (horizontale oder vertikale Anreihung von Elementen) und das ConstraintLayout (Verknüpfen von Elementen mittels Constraints; deutsch „Bedingungen“) verwendet.

Um in jeder Ansicht das Logo der App einzublenden und den Titel anzuzeigen, wird das Widget Toolbar in die Activities eingebunden. Dieses ersetzt die standardmäßige Toolbar, welche mithilfe eines style-Attributs im Android Manifest deaktiviert werden kann.

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    ...
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"
    android:theme="?attr/actionBarTheme"
    app:navigationIcon="@mipmap/bambi_guard_logo"
    app:titleTextColor="@color/colorWhite" />
```

Programmausdruck 15: Toolbar in den Layout-Dateien der BambiGuard App

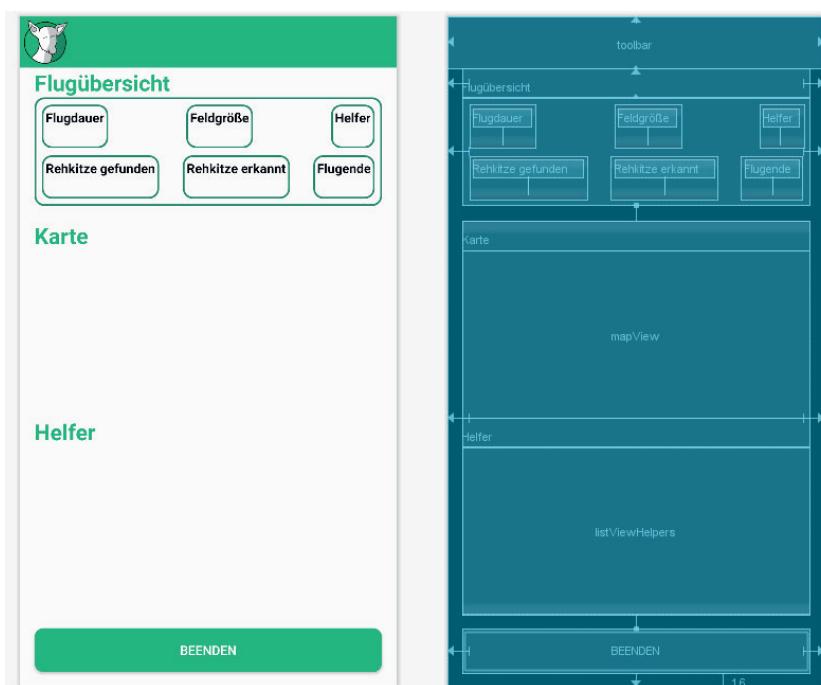


Abbildung 61: activity_flight_end.xml im Layout-Editor

Bei der Erstellung der Activities zur Flugplanung entstanden erst einige Probleme, da in vorhergegangenen Versionen des Wireframes die Größe der Elemente unterschätzt wurde und zu wenig Platz auf einem Smartphone-Bildschirm vorhanden war. Aus diesem Grund wurde die Flugplanung auf zwei Ansichten aufgeteilt, welche in einer Activity umgesetzt, aber mit Fragments separiert sind. Der ursprüngliche Entwurf, bei dem die Karte und die Eingabefelder in derselben Ansicht zu sehen sind, wurde verworfen. Diese Fehlkonzeptionen kosten Zeit und Arbeitsstunden, führen aber zu einem runderem Ergebnis, da die Probleme besser und ganzheitlicher gelöst werden. Eine Alternative zum Aufteilen auf zwei Fragments ist das Einfügen in eine ScrollView. Es wurde dagegen entschieden, da der Benutzungsfluss und die Übersichtlichkeit darunter leiden würde.

Die zwei Fragments der Flugplanung werden nun in der Activity mittels des FragmentManager organisiert und beim Drücken des „Weiter“-Buttons ersetzt.

```
private void replaceFragment(Fragment fragment) {
    fragmentManager.beginTransaction()
        .setReorderingAllowed(true)
        .replace(R.id.fragmentContainerView, fragment, null)
        .commit();
}
```

Programmausdruck 16: Ausschnitt aus FlightPlanningActivity.java

Neben der Flugplanung wird auch die JoinFlightActivity (Einstieg des Helfers in den Flug) mit Fragments gelöst. Mithilfe des ViewPager2 können Fragments angezeigt werden und mit einem Swipe („Wisch“-Bewegung) verknüpft werden. Um diesen zu implementieren wird eine Unterklasse des FragmentStateAdapter erstellt (at.ac.szybbs.bambiguard.fragments.helperTutorial.SliderAdapter). Dieser enthält einzig eine Liste mit allen Fragments, die im ViewPager angezeigt werden sollen.

```
private void initFragmentSlider() {
    ArrayList<Fragment> fragments = new ArrayList<>();
    fragments.add(new HelperTutorial1Fragment());
    fragments.add(new HelperTutorial2Fragment());
    fragments.add(new HelperTutorial3Fragment());

    ViewPager2 viewPager = findViewById(R.id.viewPager);
    FragmentStateAdapter adapter = new SliderAdapter(
        fragments, this);
    viewPager.setAdapter(adapter);
}
```

Programmausdruck 17: Ausschnitt aus JoinFlightActivity.java



Abbildung 62: Erstes Fragment des ViewPagers (fragment_helperTutorial1.xml)

Flugsteuerung

Die Steuerung der Drohne (`PilotFlightActivity`) findet im Querformat (*Landscape-Orientierung*) statt. Damit die App beim Wechseln auf diese Activity die Orientierung ändert, wird im Android Manifest bei der Deklaration der Activity das Attribut `screenOrientation` auf `sensorLandscape` gesetzt. Mit dieser Konfiguration ist Querformat in beide Richtungen erlaubt. Je nach Rotation des Smartphones wird die Ansicht gedreht.

```
<activity
    android:name=".activities.PilotFlightActivity"
    android:screenOrientation="sensorLandscape"
    android:hardwareAccelerated="true"/>
```

Programmausdruck 18: PilotFlightActivity im Android Manifest (AndroidManifest.xml)

Für die Flugsteuerung werden überwiegend die Widgets der DJI UX SDK verwendet. Dabei ist ein Bug in Android Studio aufgetreten, welcher das Anzeigen des Layouts im Designer verhindert. Es ist nicht möglich das Aussehen zu kontrollieren, wenn Widgets eingebunden sind. Dieser Fehler ist jedoch nur im Projekt BambiGuard aufgetaucht und nicht in den Demoprojekten von DJI. Aus diesem Grund wurde die XML-Datei parallel in ein anderes Projekt kopiert, um das Layout zu gestalten und zu kontrollieren.

Kompass

Ein Helfer wird mithilfe eines Pfeiles, welcher wie ein Kompass funktioniert, in die Richtung eines Rehkitzes geleitet. Die Anzeige ist als Grafik eingebunden, welche um ihre Mitte rotiert und so die Richtung verändert.

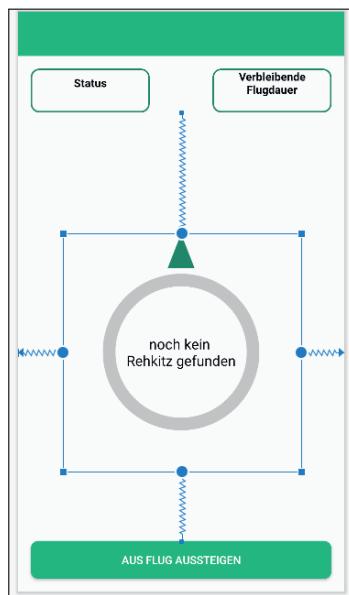


Abbildung 63: Kompass im Layout-Editor

Dialoge

In der Gestaltung der User Experience sind Dialoge ein geeignetes Mittel, um den Benutzer Informationen mitzuteilen oder eine Aktion zu erzwingen. In BambiGuard werden sie verwendet, wenn keine Verbindung zum Server aufgebaut werden kann (siehe Abschnitt 3.7 Nodejs Server). In diesem Fall öffnet sich eine Box, in welcher der Benutzer aufgefordert wird, Wifi oder Mobile Daten zu aktivieren. Diese Art von Dialog ist als AlertDialog umgesetzt, welcher nur einen positiven Button hat und verworfen werden kann.

```
AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);
dialogBuilder
    .setTitle(R.string.error_when_connecting_with_server)
    .setMessage(R.string.please_turn_on_wifi_to_connect)
    .setCancelable(true)
    .setPositiveButton(R.string.ok, (dialog, id) -> {
        viewModel.getSocket().reconnect();
    });
AlertDialog alertDialog = dialogBuilder.create();
alertDialog.show();
```

Programmausdruck 19: Alert Dialog in der Datei FlightPlanningActivity.java

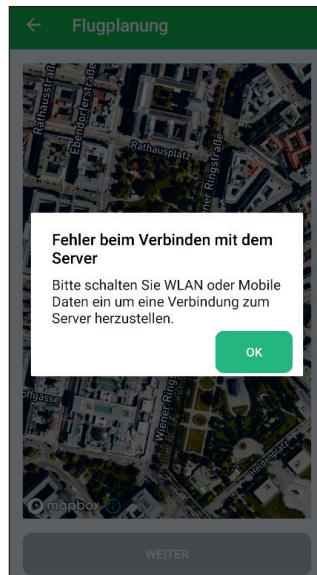


Abbildung 64: AlertDialog, wenn keine Verbindung zum Server aufgebaut werden kann

Eine zweite Art von Dialog wird beim Anschließen der Drohne angezeigt. Dieser soll den Benutzer zwingen, eine Entscheidung zu treffen. Entweder kann er die Drohne anschließen oder die Aktion abbrechen. Dieser Dialog ist als spezialisiertes DialogFragment umgesetzt. Das Fragment erstellt zwar ein normales AlertDialog, kann die Funktionalität der Buttons und der Verbindungs-Informationen jedoch in eine eigene Klasse auslagern. Dieser Dialog ist nicht abbrechbar, der positive Button wird erst aktiv, sobald die Drohne verbunden und erkannt wurde.

```
public class ConnectDroneDialogFragment extends DialogFragment {
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.
            Builder(getContext());
        ...
        return dialog;
    }
    ...
}
```

Programmausdruck 20: Ausschnitt aus ConnectDroneDialogFragment.java

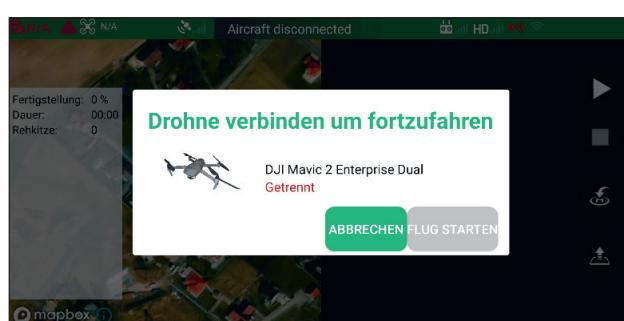


Abbildung 65: Dialog beim Verbinden mit der Drohne

Listenansichten

Ein weiteres spezielleres Element der BambiGuard-App sind die Listen, in denen die Helfer gezeigt werden. Indes sie zuerst als Liste mit einem Button zum Hinzufügen neuer Helfer konzeptioniert waren, ist nach Überlegungen zur Umsetzung eine andere Variante gewählt worden.

Während der Flugplanung werden verfügbare Helfer, welche in einer `ListView` angezeigt werden, ausgewählt und damit in eine zweite `ListView` verschoben. Dadurch sieht man immer diejenigen Helfer, die noch keinem Flug zugewiesen sind und separat die schon inkludierten Helfer. Diese Umsetzung ist einfacher und schlanker zu realisieren, da sie mit einer Klasse (`ListViewAdapter extends ArrayAdapter<String>`) und simplen `OnClickListener` auskommt.

```
ListViewAdapter listViewAvailableHelpersAdapter = new
    ListViewAdapter(getContext(), R.layout.list_view_row);
ListView listViewAvailableHelpers = view
    .findViewById(R.id.listViewAvailableHelpers);
listViewAvailableHelpers.setAdapter(listViewAvailableHelpersAdapter);
listViewAvailableHelpers.setOnItemClickListener(
    (parent, view, position, id) -> moveHelper(position));
```

Programmausdruck 21: ListView Initialisierung in der Datei `FlightPlanning2Fragment.java`

3.5.4 Flugplanung mit Mapbox

Für die Definition des Flugbereichs im Rahmen der Flugplanung ist eine Karte vorgesehen. Der Benutzer soll mithilfe von Markern den Bereich individuell abstecken können. Für die Umsetzung dieser Anforderungen stehen auf Android mehrere Karten APIs zu Verfügung, wie Google Maps API, Mapbox oder Bing Maps SDK. Obwohl die von Google entwickelte Maps API von Grund auf eine gute Einbindung in Android Apps und eine gute Dokumentation und Community bietet, ist der Drittanbieter Mapbox gewählt worden. Der Hauptgrund ist die Preisgestaltung der verschiedenen APIs – Mapbox kann im Gegensatz zur Google Maps API in größerem Rahmen gratis und ohne Angaben von Zahlungsinformationen genutzt werden. Die technischen Spezifikationen von Mapbox sind durchwegs vergleichbar mit der Google Maps API, einzig in der Protokollierung und Bereitstellung von Demoprojekten ist sie nicht auf demselben Niveau, wie die Google Maps API.

API-Einrichtung

Für die Verwendung der Mapbox API ist ein Schlüssel nötig, welchen man durch die Registrierung auf der Website erhält (mapbox.com). Es können mehrere Projekte und somit mehrere Tokens (deutsch „Wertmarken“) erstellt werden.

Name	Token	Last modified	URLs
Default public token	pk.eyJ1IjoiY2x1bwVuc2xvc2JpY2hsZXIiLCJhIjoiY2t3N mlJwc3RhMHazYjJxcWx4amZ2enp5MCJ9.V9mmk3- OdGmgr1b6xwDSXg	3 months ago	N/A
BambiGuard	SECRET TOKEN	3 months ago	0

Abbildung 66: API-Schlüssel in der Web-Oberfläche von Mapbox

Der zuvor generierte Token muss nun in die Applikation eingebunden werden. Dies kann, wie in der Dokumentation empfohlen, in der Datei strings.xml geschehen, um eine zentrale Stelle für konstante Definitionen von Zeichenketten zu haben.

```
<string name="mapbox_access_token">sk.eyJ1IjoiY2x1bwVuc2xvc2JpY2hsZXIiLCJhIjoiY2t3N</string>
```

Programmausdruck 22: API-Schlüssel in der Datei strings.xml

Das Mapbox SDK für Android wird mit dem Distributions-Werkzeug Maven ausgeliefert. Die Konfiguration muss auf Projekt-Ebene hinzugefügt werden, um die Inhalte herunterzuladen und verwenden zu können.

```
allprojects {
    repositories {
        ...
        maven {
            url 'https://api.mapbox.com/downloads/v2/releases/maven'
            authentication {
                basic(BasicAuthentication)
            }
            credentials {
                username = 'mapbox'
                password = ***
            }
        }
    }
}
```

Programmausdruck 23: Maven Definition in der Datei build.gradle (Projekt-Ebene)

Das SDK kann nun in die Abhängigkeiten der gradle-Datei auf Modul-Ebene hinzugefügt werden.

```
implementation 'com.mapbox.mapboxsdk:mapbox-android-sdk:6.2.1'
```

Programmausdruck 24: Mapbox Implementierung in der Datei build.gradle (Modul-Ebene)

Folgende Implementierung in der Activity fügt die Karte, mit Zoom auf die Wiener Innenstadt hinzu. Mithilfe des Attributs mapbox_styleUrl kann das Aussehen der Karte geändert werden. Es stehen Straßen, Satelliten, Verkehr und weitere Styles zur Verfügung. Hier wird die Satellitenkarte mit eingezeichneten Straßen verwendet.

```
<com.mapbox.mapboxsdk.maps.MapView  
    android:id="@+id/mapView"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:mapbox_cameraTargetLat="48.21"  
    app:mapbox_cameraTargetLng="16.36"  
    app:mapbox_cameraZoom="15"  
    app:mapbox_styleUrl="mapbox://styles/mapbox/satellite-streets-  
    v11" />
```

Programmausdruck 25: Mapbox-Tag in der Datei activity_flight_planning1.xml

Einzeichnungen, wie Orte, Straßen oder Gewässer werden in einer bestimmten Sprache beschriftet. Jede Ebene (englisch „Layer“) der Karte bekommt ein Attribut, welches die Sprache seiner Beschriftungen bestimmt. Um nun die Sprache der gesamten Karte zu ändern, werden die Eigenschaften jeder Ebene überarbeitet.

```
for (Layer layer : map.getLayers ())  
    layer.setProperties(PropertyFactory.textField("{name_de}"));
```

Programmausdruck 26: Karte übersetzen in der Datei FlightPlanning1Fragment.java

Erfassung der Position

Beim Starten der Activity zur Flugplanung soll die Position des Piloten ermittelt, in der Karte eingezeichnet und auf diese fokussiert werden. Dazu ist es nötig, die Berechtigung zur Entnahme der genauen Position in das Android Manifest einzutragen (redundant mit Berechtigungen der DJI SDK; beschrieben in Abschnitt Arbeiten mit der DJI SDK).

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Programmausdruck 27: Definition der Berechtigung für die genauer Position in der Datei `AndroidManifest.xml`

Die Berechtigungen müssen während der Laufzeit der App nochmals überprüft werden, um eine korrekte Funktionalität zu garantieren. Standardmäßig wird das mit der Funktion `checkSelfPermission` gelöst. Mapbox enthält jedoch die Helferklasse `PermissionsManager`, welche die Verwaltung erleichtert. Damit reicht folgender Programmcode, um die Berechtigungen abzuarbeiten. Die Funktion `enableLocation` wird aufgerufen, sobald die Karte von der Mapbox API geladen wurde.

```
private void enableLocation() {
    if (PermissionsManager.areLocationPermissionsGranted(
        getContext())) {
        initLocationEngine();
        initLocationLayer();
    } else {
        permissionsManager = new PermissionsManager(this);
        permissionsManager.requestLocationPermissions(
            getActivity());
    }
}
```

Programmausdruck 28: Position Erfassen in der Datei `FlightPlanning1Fragment.java`

Um die Position des Smartphones zu erhalten, wird die Klasse `LocationEngine` verwendet, welche automatisch den besten Provider (GPS oder Netzwerk) wählt. Danach gibt dessen Methode `getLastLocation` Koordinaten zurück, auf welche die Karte fokussieren kann. Durch die Implementierung des Interface `LocationEngineListener`, wird die Position laufend aktualisiert.

```
private void initLocationEngine() {
    locationEngine = new LocationEngineProvider(getContext())
        .obtainBestLocationEngineAvailable();
    locationEngine.setPriority(LocationEnginePriority
        .HIGH_ACCURACY);
    locationEngine.activate();

    Location lastLocation = locationEngine.getLastLocation();
    if (lastLocation != null) {
        setCameraPosition(lastLocation);
        ...
    } else {
        locationEngine.addLocationEngineListener(this);
    }
}
```

Programmausdruck 29: Funktion `initLocationEngine` in der Datei `FlightPlanning1Fragment.java`

Das LocationLayerPlugin ist dazu da, die Position als Knopf auf der Karte anzuzeigen und dem Benutzer ein Feedback zu geben.

```
private void initLocationLayer() {
    locationLayerPlugin = new LocationLayerPlugin(
        mapView, map, locationEngine);
    locationLayerPlugin.setLocationLayerEnabled(true);
    locationLayerPlugin.setCameraMode(CameraMode.NONE);
    locationLayerPlugin.setRenderMode(RenderMode.NORMAL);
}
```

Programmausdruck 30: Die Funktion initLocationLayer in der Datei FlightPlanning1Fragment.java

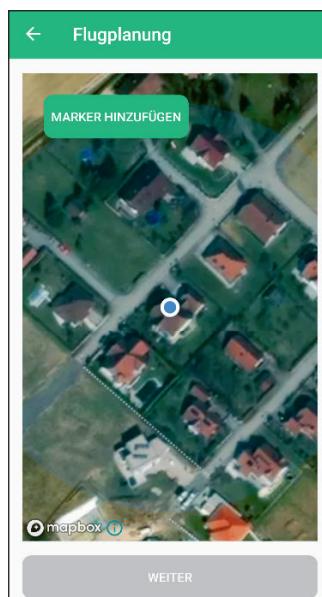


Abbildung 67: Mapbox Karte mit Position

Da die Dokumentation von Mapbox nur die Implementierung in Kotlin beschreibt, muss man für Beispiele in Java auf andere Ressourcen für die Entwicklung zurückgreifen. Das ist bei Basisfunktionen, wie dem Anzeigen der Position, leicht möglich. Es existieren beispielsweise offizielle Anleitungen als Videos von Mapbox, in denen manche Teile der API in Java ausgeführt werden.

Marker

Das Gebiet des späteren Flugs wird mit Markern und einem allgemeinen Polygon definiert. Der Benutzer soll die Möglichkeit haben individuelle Formen und Felder abzustecken. Zuerst werden vier initiale Marker hinzugefügt, mit welchen die grobe Form des Feldes umrissen wird. Danach kann man an jeder Strecke, welche die Marker miteinander verbindet, einen neuen verschiebbaren Marker hinzufügen und in die richtige Position bringen. Auf diese Weise können immer genauere Bereiche definiert werden.

Marker werden in Mapbox durch dessen Koordinaten (Längen- und Breitengrad) und einem Aussehen (Icon) festgelegt.

```
private void addMarker(LatLng position) {
    MarkerOptions markerOptions = new MarkerOptions().
        position(position);
    Icon icon = IconFactory.getInstance(getApplicationContext())
        .fromResource(R.drawable.marker_100x100);
    Marker marker = map.addMarker(markerOptions.icon(icon));
    markers.add(marker);
}
```

Programmausdruck 31: Die Funktion `addMarker` in der Datei `FlightPlanning1Fragment.java`

Die initialen Marker werden als Quadrat mit einer Länge von 20 m um die Position des Piloten gelegt. Mapbox inkludiert Funktionen zur Kartenprojektion, um die real dreidimensionale Oberfläche der Erde auf eine zweidimensionale Fläche zu übertragen. Dies ist nötig, um beispielsweise Distanzen in Meter in deren äquivalente Längen- und Breitengrade umzurechnen. Folgender Code kalkuliert den Abstand in Graden der initialen Marker.

```
LatLng distances = map.getProjection()
    .getLatLngForProjectedMeters(new ProjectedMeters(20, 20));
```

Programmausdruck 32: Meter in Latitude und Longitude umrechnen in der Datei `FlightPlanning1Fragment.java`

Für jede Strecke zwischen den vier Markern wird ein Zwischenmarker zum Hinzufügen neuer Marker in dessen Mitte hinzugefügt (dargestellt mit Plus-Icons). Die Positionen der Zwischenmarker werden durch das Addieren der halben Strecke zum ersten Punkt der Strecke errechnet. `point1` und `point2` sind die zwei Eckpunkte der Strecke und somit zwei der vier initialen Marker.

```
double distLong = point1.getLongitude() - point2.getLongitude();
double distLat = point1.getLatitude() - point2.getLatitude();
LatLng position = new LatLng(point2.getLatitude() + distLat / 2,
    point2.getLongitude() + distLong / 2);
```

Programmausdruck 33: Initiale Marker positionieren in der Datei `FlightPlanning1Fragment.java`

Für die bessere Visualisierung des Feldes unterlegt BambiGuard das durch die Marker definierte Polygon mit einer füllenden Farbe. Die Implementierung in Mapbox erfolgt durch das Erstellen einer Liste mit den Positionen aller Marker und Hinzufügen eines Polygons mithilfe von `PolygonOptions` und der Funktion `MapboxMap.addPolygon`.

```
List<LatLng> positions = new ArrayList<>();  
for (Marker marker : markers) {  
    positions.add(marker.getPosition());  
}  
  
PolygonOptions polygonOptions = new PolygonOptions()  
    .addAll(positions)  
    .fillColor(Color.parseColor("#29b17b"))  
    .alpha(0.5f);  
polygon = map.addPolygon(polygonOptions);
```

Programmausdruck 34: Marker hinzufügen in der Datei FlightPlanning1Fragment.java

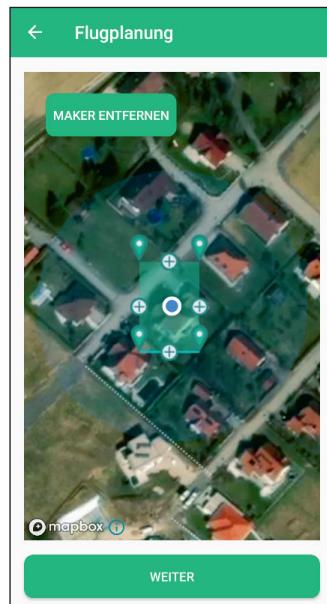


Abbildung 68: Initiale Marker in der Karte

Für die Funktionstüchtigkeit der Flugplanung müssen die Eckmarker verschoben und neue Marker hinzugefügt werden können. Der Event Listener `onTouch` wird ausgeführt, sobald die Karte mit dem Finger berührt wird. Durch die Position der Berührung kann festgestellt werden, ob ein Marker geklickt wurde und es kann basierend auf der weiterfolgenden Bewegung der Berührung der Marker verschoben werden. Wenn die Berührung losgelassen wird, entkoppelt sich der Marker von der Position der Berührung.

```

@Override
public boolean onTouch(View view, MotionEvent event) {
    if (dragMarkers(event)) {
        if (event.getAction() == MotionEvent.ACTION_UP)
            redraw();
        view.performClick();
        return true;
    }
    return false;
}

private boolean dragMarkers(MotionEvent event) {
    for (int i = 0; i < markers.size(); i++) {
        PointF touchLocation = new PointF(event.getX(), event
            .getY());
        PointF markerScreenLocation = map.getProjection()
            .toScreenLocation(markers.get(i).getPosition());
        if (getDistance(touchLocation, markerScreenLocation) < 40) {
            dragMarker(i, map.getProjection()
                .fromScreenLocation(touchLocation));
            return true;
        }
    }
    return false;
}

```

Programmausdruck 35: Marker verschieben in der Datei FlightPlanning1Fragment.java

Das Hinzufügen von Markern beim Klick auf ein Plus-Icon geschieht mit dem Event onMapClick. Die Überprüfung der Position der Berührung erfolgt wie beim verschieben der Marker (siehe Programmausdruck 35).



Abbildung 69: Individuell abgestecktes Feld mit Mapbox Markern

Ein Nachteil in dieser Methode der Absteckung des Feldes liegt darin, dass nur eine begrenzte Genauigkeit mit den Markern erzielt werden kann. Mapbox unterstützt einen gewissen Zoom-Faktor und eine Auflösung der Karte, auf welche man angewiesen ist.

Dieser Teil der BambiGuard-App entstand zu einem großen Teil auf Basis der Mapbox Dokumentation, welche wie erwähnt, keine Java-Implementierungen enthält. Einige komplexere Funktionalitäten, wie der Einsatz von eigenen Icons sind in Kotlin und Java anders umzusetzen und deswegen nicht in der Dokumentation beschrieben. Aus diesem Grund werden etwaige andere Ressourcen, wie Foren und öffentliche Repositories zur Hand genommen, wodurch die Entwicklung um einiges erschwert und verlangsamt wird.

Flugroute

Aus der Recherche zum DJI SDK wird angenommen, dass die Missionstypen, welche in der offiziellen App *DJI Pilot* enthalten sind, genauso im SDK stecken. In der Mapping-Mission wird ein Bereich auf einer Karte abgesteckt und dann ein Pfad für den Flug der Drohne aus dem gegebenen Polygon berechnet. Jedoch ist diese Art der Mission als solches nicht im DJI SDK implementiert. Aus diesem Grund muss auf die Wegpunkt-Mission zurückgegriffen werden. Die Berechnung des Pfades in Form von Wegpunkten aus einem definierten Polygon muss also getrennt geschehen. Diese Aufgabe wurde wegen der Fehlannahme in der Projektplanung nicht einberechnet und stellt eine deutliche Vertiefung in das Thema der Flugplanung dar.

Folgend muss nach einer Lösung für das Problem gesucht werden. Die Aufgabenstellung der Berechnung eines optimalen Pfades, welcher ein bestimmtes Feld (Polygon) abdecken soll, wird *Coverage Path Planning* (deutsch „Abdeckungspfadplanung“) genannt. Es existieren einige algorithmische Ansätze, um das Problem zu lösen, wie in etwa die sogenannte *Cellular Decomposition*. Dabei wird das Polygon in seine konvexe Teilpolygone aufgeteilt (konvex = jeder Winkel ist kleiner als 180°). Danach können diese Zellen durch den bekannten Algorithmus *Boustrophedon Cell Decomposition* mit gleichmäßig abdeckenden Pfaden bestückt werden.²

Eine zweite Methode des Coverage Path Planning hinterlegt dem Polygon ein quadratisches Raster mit einer Breite, welche die Kamera abdecken kann. Die ergebenden Punkte werden als Eingabe für die Berechnung des minimalen *Spanning Tree*³ verwendet. Der kleinste Weg ist demnach der Pfad, auf dem die Drohne danach fliegt. Diese Variante bringt jedoch mehr Nachteile mit sich. Die Implementierung braucht zwar im Vergleich zur Cellular Decomposition weniger Programmzeilen, ist aber

2 Vgl. Bäh 2019

3 Der *Minimum Spanning Tree* ist die kleinste Anzahl an Kanten, welche alle Punkte eines Graphen ohne Schleifen abdeckt.

weniger effizient beim Einsatz von Drohnen. Der Algorithmus zieht die Anzahl an Richtungswechsel nicht in die Berechnung mit ein. Während eines Richtungswechsel kann sich die Drohne jedoch nicht fortbewegen und deswegen vervollständigt sie die Mission langsamer. Außerdem ist durch das Raster ein Überlauf über das abgesteckte Polygon nicht zu vermeiden, es sei denn, auftretende Lücken an den Rändern werden für die Suche der Rehkitze vernachlässigt.

Es gibt einige Softwarereprodukte von Drittanbietern, welche die Aufgabenstellung in ihren Systemen in einer gewissen Weise gelöst haben. Beispiele dafür sind Drone Harmony⁴, DJIFlightPlanner⁵ oder Ardupilot Mission Planner⁶.

Zu sehen ist, dass Lösungen für das Coverage Path Planning existieren, jedoch entweder nicht öffentlich zugänglich oder nur in Prototypen oder generellen Programmiersprachen implementiert sind. Deswegen muss eine eigene Implementierung in der Java-Applikation erstellt werden.

Es wird ein einfacher Algorithmus entwickelt, welcher zwar nicht alle Grenzfälle abdeckt, jedoch im Sinne eines Prototypen für diesen Teil der Anwendung ausreicht. Die Berechnung nimmt den Punkt, welcher am weitesten unten rechts liegt als Ausgangspunkt für den Pfad. Es wird eine Linie nach links gezogen, so weit, bis das Ende dieser Linie außerhalb des Polygons (minus dem Sichtbereich der Kamera) ist. Die Kante des Polygons wird nach oben verfolgt und sobald der vertikale Abstand zur davor horizontal gezogenen Linie so groß, wie der Sichtbereich der Kamera ist, dreht sich die Linie nach rechts und zielt auf die rechte Kante des Polygons zu. Dieser Vorgang wiederholt sich, bis das Polygon zur Gänze abgedeckt wurde.

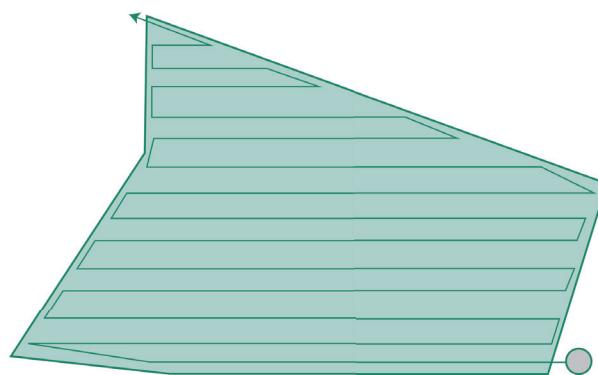


Abbildung 70: Optimaler Pfad des Algorithmus

4 <https://droneharmony.com/> 03.03.2022

5 <https://www.djiflightplanner.com/> 03.03.2022

6 <https://ardupilot.org/planner/> 03.03.2022

Die Ermittlung, ob die Linie an einem gewissen Punkt innerhalb des Polygons liegt, erfolgt durch folgende Berechnung. Wirft man einen Strahl ausgehend von dem Punkt, welcher überprüft werden soll, nach rechts, so kann man die Anzahl an Kreuzungen mit einem Segment des Polygons ermitteln. Ist die Anzahl gerade (0 eingeschlossen), dann ist der Punkt außerhalb des Polygons. Ist die Anzahl ungerade, liegt der Punkt innerhalb.

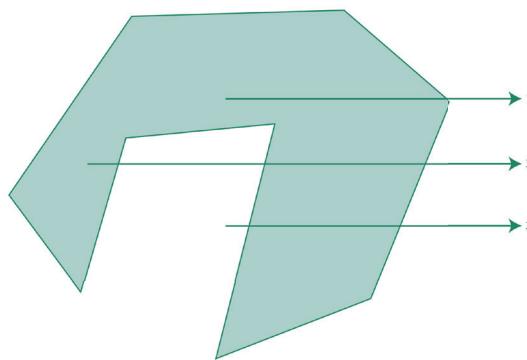


Abbildung 71: Anzahl der Kreuzungen eines Strahls mit den Kanten des Polygons

Der beschriebene Algorithmus funktioniert nur für konvexe Polygone, da Winkel über 180° einen Bereich generieren, der vom Pfad nicht überprüft wird. Bei dieser Bedingung sind nur Winkel betroffen, welche die Horizontale (x-Achse) überschreiten.⁷

Die Implementierung des Algorithmus erfolgt in der Klasse `BambiGuardCoveragePlanner` direkt in der Android App. Die ausführende Funktion `decomposePolygon` nimmt folgende Parameter: eine Liste von Punkten, welche das Polygon definieren und die Breite des Videos in Metern, welche die Kamera liefert. Zur Vereinfachung der Programmierung werden nicht alle Teile des Algorithmus optimal und vollkommen implementiert. Das hat den Grund, dass durch die Neuentwicklung dieses Teils der BambiGuard App ein großer Mehraufwand entstanden ist. Bei einer detaillierteren Ausarbeitung hätten andere Bereiche des Projektes in deren Qualität und Umsetzung gelitten, da das zu viel Zeit in Anspruch genommen hätte.

Das Polygon wird mitgegeben und die Ausgangslinie als die untere Linie der Bounding-Box (umschließendes Rechteck) des Polygons berechnet. Dann werden vom Polygon eingeschlossene horizontale Linien im Abstand der mitgegeben Videobreite gebildet. Die Genauigkeit der Überschneidungen mit der Kante des Polygons ist in der Klasse als Konstante definiert.

⁷ <http://www.paulbourke.net/geometry/polygonmesh/> 06.03.2021

```

public static ArrayList<Point> decomposePolygon(
    ArrayList<Point> polygon,
    double cameraCoverage) {
    ArrayList<Line> intersections = generateIntersections(
        polygon, cameraCoverage);
    return orderPoints(intersections);
}

private static ArrayList<Line> generateIntersections(
    ArrayList<Point> polygon, double cameraCoverage) {
    double[] bounds = calculatePolygonBoundingBox(polygon);
    Line startingLine = new Line(
        new Point(bounds[0], bounds[1]),
        new Point(bounds[0], bounds[3]));
    ArrayList<Line> lines = new ArrayList<>();
    lines.add(startingLine);

    int iterations = (int) Math.ceil((bounds[2] - bounds[0]) /
        cameraCoverage)

    for (int i = 0; i < iterations; i++) {
        Line line = startingLine
            .getParallelLine(-cameraCoverage * i);
        lines.addAll(getLinesIntersectingWithPolygon(polygon,
            line));
    }
    return lines;
}

```

Programmausdruck 36: Ausschnitt aus der Datei BambiGuardCoveragePlanner.java

3.5.5 Debugging und Testen mit der Drohne

Die Programmierung einer Android App kennt mehrere Arten des Debuggings. Entweder wird ein Simulator gestartet, welcher die wichtigsten Sensoren eines Smartphones nachstellen kann oder die kompilierte Anwendung wird über die Android Debug Bridge auf einem physischen Gerät installiert.

Die Fernsteuerung der DJI Mavic 2 Enterprise besitzt einen USB-Konnektor, um Smartphones anzuschließen und damit zu kommunizieren. Im Lieferumfang der Drohne sind mehrere Kabel, einschließlich eines USB-Typ C und Mikro USB für Android Geräte und eines mit dem Apple Lightning-Anschluss, beigelegt.

Alle Funktionen einer BambiGuard Android Applikation, welche keine Verbindung mit der Drohne brauchen, können im Simulator oder mit dem einfachen USB-Debugging am eigenen Smartphone getestet werden. Wenn die Drohne zum Einsatz kommt, muss WiFi-Debugging am Android Gerät in Verwendung sein, da der USB-Anschluss durch die Fernsteuerung belegt wird.

DJI Assistant 2

In beschriebener Weise kann die Drohne mithilfe des eigenen Smartphones angesteuert werden. Für das Testen und Debugging einer Applikation ist es jedoch nicht zumutbar, sich in geeignetem Terrain für einen Drohnenflug aufzuhalten. Deswegen stellt DJI einen Flugsimulator zur Verfügung, welcher alle Funktionen der Drohne virtualisiert. Der *DJI Assistant 2 für Mavic* ist kompatibel mit der DJI Mavic 2 Enterprise und für Windows und MacOS veröffentlicht.

Ein USB-Typ C Kabel verbindet den Rumpf der Drohne und den Computer mit der Simulations-Software. Obwohl durch die Simulation keine Hardware-Funktionen der Drohne, wie das Bewegen der Rotoren, funktionieren, werden die Rotorblätter trotzdem aus Gründen der Sicherheit abmontiert.

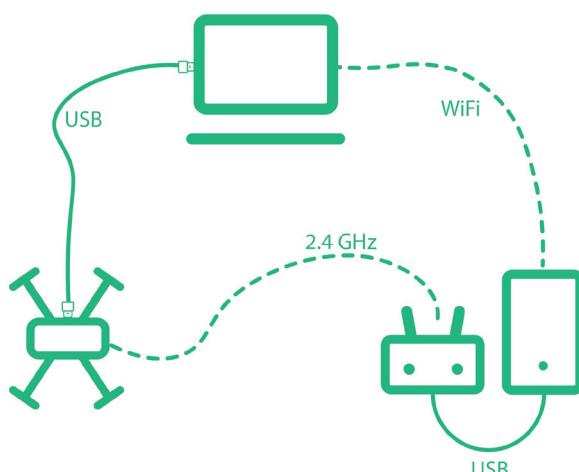


Abbildung 72: Aufbau des Debuggings

Die erste Installation des DJI Assistant auf Windows wies eine Inkompatibilität mit den Systemtreibern, welche die Drohnen ansteuern, auf. Dies führte dazu, dass obwohl alle Konfigurationen und Kabel richtig eingesetzt wurden, keine Verbindung zum Fluggerät aufgebaut werden konnte. Das Problem wurde dadurch gelöst, dass eine alte Hauptversion der Software installiert wurde, welche noch einen anderen Treiber verwendet. Danach kann die aktuelle Version heruntergeladen und die funktionstüchtigen Treiber verwendet werden. Dieser Bug sorgte für viel Verwirrung beim ersten Ausprobieren der Komponenten und wurde erst durch eine Suche in alten Beiträgen des DJI Forums gelöst.⁸

Abbildung 73 zeigt die Einstellungen des Simulators auf der rechten Seiten und das Fenster, in dem man die Drohne beobachten kann, auf der linken.

⁸ Siehe <https://forum.dji.com/forum.php?mod=viewthread&tid=134139> 02.03.2022

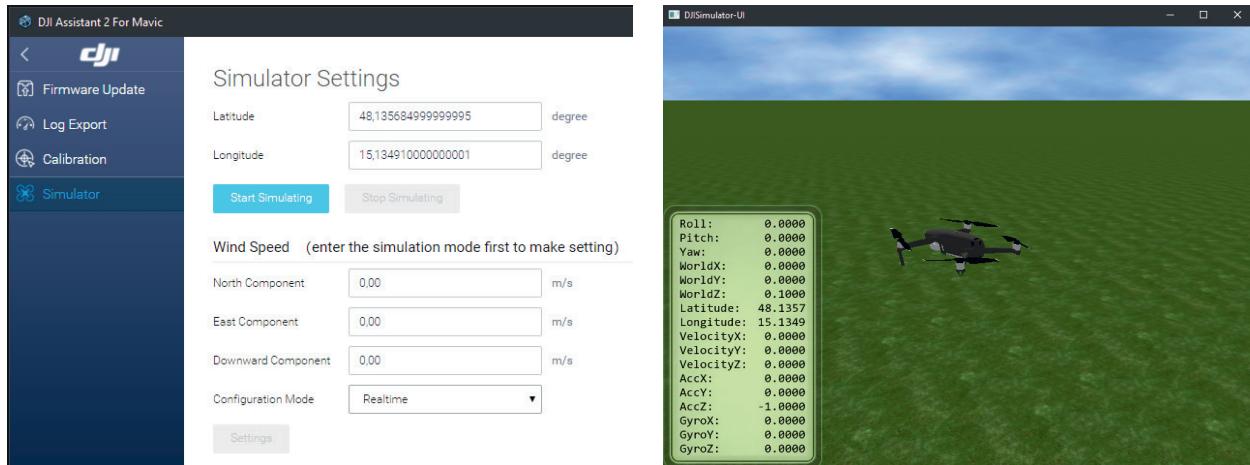


Abbildung 73: Flugsimulation im DJI Assistant 2

3.5.6 Arbeiten mit des DJI SDK

DJI stellt für die Entwicklung von Applikationen für Smartphones ein Software Development Kit (SDK) zur Verfügung. Es erlaubt den Zugriff auf diverse Produkte des Herstellers von iOS oder Android-Geräten aus. Dieses sogenannte Mobile SDK ist ein Teil der Entwicklungs-Werkzeuge von DJI, welche noch das UX SDK (vordefinierte UI-Elemente), Windows SDK, Payload SDK (für industrielle Einsätze) und das Onboard SDK (Computer der Drohne selbst) beinhalten. In der BambiGuard-App wird neben dem Mobile SDK für die Funktionalität auch das UX SDK für ein konsistentes Erscheinen im Sinne der offiziellen DJI-Software eingesetzt. Alle Bibliotheken des Herstellers stehen frei zur Verfügung, benötigen für die Entwicklung jedoch ein Hardware-Produkt, wie im Fall von BambiGuard die Drohne DJI Mavic 2 Enterprise.

Mobile und UX SDK

Die Verwendung des Mobile SDK bedingt die minimale Android API-Version 19 (Android 4.4) und eine geeignete Entwicklungsplattform, wie beispielsweise Android Studio 1.5. Zur Zeit der Entwicklung ist die aktuelle und die in BambiGuard verwendete Version des Android Mobile SDK die Version 4.15. Sie wurde am 24.06.2021 veröffentlicht.⁹ Folgende Funktionen werden von der Bibliothek abgedeckt:

- Flugsteuerung (Manuell oder durch vordefinierte Missionen)
- Flug- und Sensordaten auslesen
- Steuerung der Kamera und des Gimbals
- Echtzeit Videofeed der Kamera
- Zugriff auf Speichermedien der Drohne
- Status und Batteriestand der Fernsteuerung

⁹ <https://developer.dji.com/mobile-sdk> 03.03.2022

Mit der von DJI selbst entwickelten App *DJI Pilot* kann man mit Drohnen des Herstellers manuell fliegen, Missionen planen und durchführen, die Kamera steuern und auf die Speichermedien der Drohne zugreifen. Sie kann als vollständige Implementierung des SDK angesehen werden, wobei sie dem Piloten noch mehr Funktionen bietet. Bei der Verwendung der von DJI vordefinierten UI-Elemente (UX SDK) kann man sich an der offiziellen App orientieren, um die Benutzererfahrung anzupassen und dem Piloten Vertrautheit mit der Umgebung zu schaffen.

Hilfe und Support

Primärquelle beim Entwickeln einer Anwendung für eines der Produkte von DJI ist die offizielle API-Dokumentation. Sie beinhaltet einerseits Referenzen für alle Funktionen und Klassen der Bibliotheken und andererseits Erklärungen der wichtigsten Konzepte. Außerdem finden sich darin einige Beispiele, womit ein schneller Einstieg in die Entwicklung gefördert wird.

Existiert dennoch ein Problem bei der Programmierung oder tauchen weitere Fragen auf, kann man sich direkt an DJI mittels Developer Support¹⁰ wenden. Alternativ dazu gibt es ein belebtes GitHub-Repository und einen StackOverflow-Tag [dji-sdk]. Über diese beiden Wege kann mit anderen Entwicklern der Community kommuniziert und untereinander ausgetauscht werden.

Einrichtung

Für den Einsatz jedes SDK von DJI muss man sich auf dessen Entwickler-Website¹¹ registrieren und zuerst eine App erstellen, um einen API-Key zu erhalten. Dieser Schlüssel ist dazu da, dass der Hersteller das Projekt eines Entwicklers identifizieren und die Nutzung seines Produktes nachverfolgen kann.

CREATE APP	
SDK	Mobile SDK
APP Name	Demo App
Software Platform	Android
Package Name <small>(*)</small>	com.dji.demo
Category	Agricultural applications
Description	Eine Demo App
<input type="button" value="CANCEL"/> <input type="button" value="CREATE"/>	

Abbildung 74: Eine App im Developer-Tool von DJI erstellen

10 <https://sdk-forum.dji.net/hc/en-us> 05.03.2022

11 <https://developer.dji.com> 05.03.2022

Wichtig ist die Angabe des korrekten Package Namens. Durch diesen wird jede Android-App eindeutig identifiziert. Stimmt der eingegebene Wert nicht mit dem Namen der App zusammen, wird das SDK nicht freigegeben und ist nicht funktionstüchtig.

Nach der Erstellung wird der Schlüssel als `meta-data` in das Android-Manifest eingefügt.

```
<meta-data
    android:name="com.dji.sdk.API_KEY"
    android:value="a992f561342b29e506a****" />
```

Programmausdruck 37: DJI API-Schlüssel in der Datei `AndroidManifest.xml`

Architektur der Mobile SDK

Die oberste Instanz des SDK ist der **SDK Manager**. Dieser verwaltet die Registrierung der App bei DJI, sowie die Verbindung und den Zugang zum Produkt, welches eine Drohne oder dessen Steuerung sein kann. Jedes Produkt besteht aus mehreren Komponenten, wie beispielsweise der Kamera, dem Gimbal oder dem Flug-Controller. Parallel dazu werden über den SDK Manager auch die Objekte für die Missionsplanung, Durchführung und Überwachung aufgerufen (**Mission** und **Mission Control**).

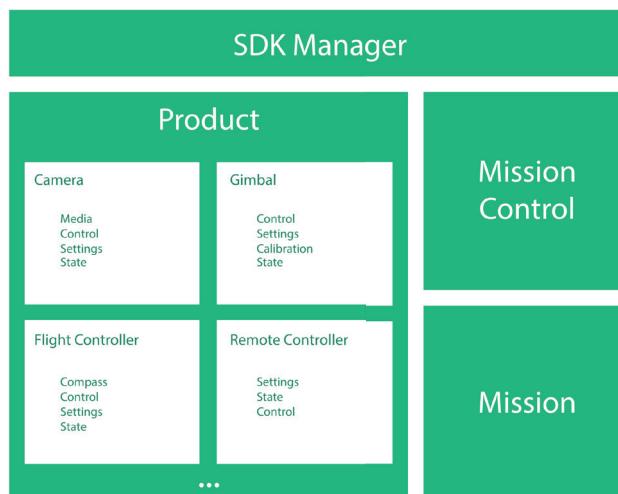


Abbildung 75: Mobile SDK Architektur¹²

Die Bibliothek beinhaltet vorgefertigte Missionen, auf welchen man zur weiteren Entwicklung aufbauen kann. Die wichtigste ist die Wegpunkt-Mission (englisch „*waypoint mission*“; in einem folgenden Absatz näher beschrieben), bei der eine Reihe von Koordinatenpunkten nacheinander abgeflogen wer-

12 Vgl. <https://developer.dji.com/document/45a1decb-e86c-4fcf-baf8-95420bf2acb8> 06.03.2021

den. Daneben gibt es noch eine Hot Point Mission (wiederholtes Rotieren um einen Punkt), Follow Me Mission (Folgen des GPS-Signals der Steuerung) und einige Missionsarten mehr.

Vor dem Starten mit der Entwicklung einer Applikation empfiehlt es sich die Beispiel-Projekte von DJI auszuprobieren. Diese finden sich auf einem eigenen GitHub-Repository¹³. Es sind diverse Anwendungen enthalten, wie beispielsweise das Anzeigen des Video-Streams oder eine Demo zur Registrierung der App bei DJI.

SDK Registrierung

Mithilfe des Gradle Build-Systems wird die Abhängigkeit des Mobile SDK zur App hinzugefügt. Dafür werden folgende Zeilen in der Datei `build.gradle` auf App-Ebene in der Direktive `dependencies` ergänzt.

```
implementation('com.dji:dji-sdk:4.15', {
    exclude module: 'library-anti-distortion'
    exclude module: 'fly-safe-database'
})
compileOnly 'com.dji:dji-sdk-provided:4.15'
```

Programmausdruck 38: DJI SDK Implementierung in der Datei `build.gradle` (Modul-Ebene)

Zur Aktivierung des SDK muss die Applikation beim erstmaligen Start mit dem Internet verbunden sein. Es wird eine Anfrage an DJI gesendet, welche unter anderem den davor generierten API-Key enthält, mit dem das Projekt identifiziert wird. Diese Aufgabe wird in BambiGuard von der Klasse `DJIApplication` erledigt, welche von `android.app.Application` vererbt. Sie ist ein Singleton und kann von allen Activities aus aufgerufen werden.

Die Funktion `registerApplication` wird beim Starten der Flugplanung ausgeführt, da ab diesem Zeitpunkt feststeht, dass der Benutzer die Drohne verwenden wird und somit das DJI SDK gebraucht wird. Bevor man über den SDK Manager die Methode `registerApp` aufruft, welche die Registrierung durchführt und entsprechende Rückmeldung gibt, müssen die Berechtigungen der App überprüft werden.

13 <https://github.com/DJI-Mobile-SDK-Tutorials> 04.03.2022

```

public class DJIApplication extends Application {
    public boolean registerApplication() {
        if (product != null)
            return true;

        SDKManagerCallback djiSdkManagerCallback =
            new SDKManagerCallback() {...};
        if (permissionsGranted()) {
            DJISDKManager.getInstance().registerApp(...);
            return true;
        }
        return false;
    }
    ...
}

```

Programmausdruck 39: Ausschnitt aus der Datei DJIApplication.java

Berechtigungen

Es sind eine Reihe an Berechtigungen nötig, um die Funktionen des SDK nutzen zu können. Zusätzlich zur Vervollständigung der Permissions im Manifest mithilfe des Tags <uses-permission>, verlangt Android eine Überprüfung und eine eventuelle Freigabe der Berechtigungen während der Laufzeit. Zu diesem Zweck wird in der Activity der Flugplanung ein Array mit allen notwendigen Permissions erstellt und vor dem Registrieren der Applikation bei DJI überprüft.

```

for (String eachPermission : REQUIRED_PERMISSION_LIST) {
    if (ContextCompat.checkSelfPermission(this, eachPermission)
        != PackageManager.PERMISSION_GRANTED) {
        missingPermission.add(eachPermission);
    }
}

if (!missingPermission.isEmpty()) {
    ActivityCompat.requestPermissions(
        this,
        missingPermission.toArray(new String[0]),
        REQUEST_PERMISSION_CODE);
}

```

Programmausdruck 40: Berechtigungen in der Datei FlightPlanningActivity.java

USB-Accessoire

Neben der Freigabe der Berechtigungen ist es essenziell die Verbindung zur Fernsteuerung via den USB-Ausgang des Smartphones zu deklarieren. Der Zugriff auf die Schnittstelle muss als Feature im Manifest eingefügt werden.

```

<uses-feature
    android:name="android.hardware.usb.host"
    android:required="false" />
<uses-feature
    android:name="android.hardware.usb.accessory"
    android:required="true" />
```

Programmausdruck 41: Deklaration des USB-Accessoires in der Datei `AndroidManifest.xml`

Außerdem wird ein *Intent-Filter* für USB-Accessoire in der Activity definiert, welche die SDK registriert. Beim Anstecken eines Geräts an den USB-Stecker wird der Hersteller, das Modell und die Version mit dem angegebenen Filter überprüft und bei Übereinstimmung direkt auf die Activity weitergeleitet.

```

<intent-filter>
    <action android:name="android.hardware.usb.action
        .USB_ACCESSORY_ATTACHED" />
</intent-filter>

<meta-data
    android:name="android.hardware.usb.action
        .USB_ACCESSORY_ATTACHED"
    android:resource="@xml/accessory_filter" />
```

Programmausdruck 42: Intent Filter und Meta-Tag der Activity `FlightPlanningActivity` in der Datei `AndroidManifest.xml`

Flugsteuerung

Das UX SDK definiert Widgets, welche die manuelle Steuerung und Flugüberwachung übernehmen. Die Elemente müssen im Layout der Activity, d.h. in dessen XML-Datei eingefügt werden. Sie initialisieren und aktivieren sich selbst, sodass wenn eine Verbindung zur Drohne besteht, die Widgets automatisch funktional werden.

```

<dji.ux.widget.BatteryWidget
    android:layout_width="96dp"
    android:layout_height="22dp"
    custom:excludeView="singleVoltage" />
```

Programmausdruck 43: DJI Widget in der Datei `activity_pilot_flight.xml`

Flugmission

Die im Abschnitt Flugplanung mit MapBox beschriebenen Wegpunkte, welche aus den Markern der Karte herausgelesen werden, bestehen aus Koordinaten mit Breitengrad, Längengrad und Flughöhe. Sie müssen nun in eine DJI Wegpunkt-Mission eingefügt werden.

Bei einer solchen Mission wird für jeden Punkt und die ihn verknüpfende Strecke eine Rotation und Geschwindigkeit festgelegt. Außerdem können auf jedem Wegpunkt Aktionen, wie das Schießen von Fotos ausgeführt werden. Während der Mission ist es dem Piloten jederzeit möglich mithilfe der Fernsteuerung die Kontrolle über die Drohne zu übernehmen. Am Ende der Ausführung kehrt die Drohne entweder zum Startpunkt zurück, landet oder bleibt am Endpunkt in der Luft stehen.

Zum Zweck der Erstellung und Verwaltung der Mission wird eine eigene Klasse BambiGuard-Mission geschrieben. Sie bekommt die Eingabeparameter des Piloten zur Konfiguration mit. Diese enthalten die gewünschte Aktion am Ende der Missionsausführung und die Wegpunkte, ausgelesen aus der Karte nach der Flugplanung. BambiGuard legt eine Höhe von 11 Metern und eine Geschwindigkeit von 3 fest, da diese Werte beim Testen der Drohne die besten Ergebnisse erzielten. Mithilfe eines Builders wird die WaypointMission erstellt und danach auf den OnBoard-Computer hochgeladen. Ein Event Listener definiert Methoden, welche beim Start, Stopp, Pausieren und weiteren Ereignissen ausgeführt werden. Dadurch bekommt man eine Rückmeldung welchen Status die Ausführung der Mission im Moment hat.

```
missionBuilder = new WaypointMission.Builder() .  
finishedAction(finishedAction)  
    .headingMode(headingMode)  
    .autoFlightSpeed(SPEED)  
    .maxFlightSpeed(SPEED)  
    .flightPathMode(WaypointMissionFlightPathMode.NORMAL);
```

Programmausdruck 44: MissionBuilder in der Datei BambiGuardMission.java

Video-Decoding

Für die Verwendung der optischen Bilddaten, sowie der Wärmebilddaten stellt DJI ein Demoprojekt zur Verfügung, welches die Rohdaten ausliest und mithilfe der Grafikbibliothek FFmpeg dekodiert und anzeigt. Die Klasse DJI CodecManager ist in der Mobile SDK enthalten und übernimmt die Aufgaben des Dekodierens und Ausgebens des Videostroms im Format YUV oder RGBA. Die Bilddaten werden von der Drohne mit dem verlustbehafteten Codec H.264 übertragen und in einer Android SurfaceTexture festgehalten. Das ist ein Speicher für Video-Streams aus einem Kamera-Preview oder dekodierten Videodaten.

Das Beispiel von DJI setzt eine `TextureView`, sowie eine `SurfaceView` zum Anzeigen des Bildes ein. Da in BambiGuard nicht die Bibliothek FFmpeg, sondern OpenCV zum Verarbeiten eingesetzt wird, müssen die Schnittstellen dieses Anwendungsteils anders gestaltet werden. Die Bilddaten des `CodecManager` können nur entweder mit einer der beiden erwähnten Methoden angezeigt werden oder als Byte Array ausgelesen werden. Dieser Designfehler des DJI SDK verhindert es ohne Umwege oder Workaround den Kamerafeed anzuzeigen und zu verarbeiten. Aus diesem Grund wird in BambiGuard folgende Problemlösung angewandt.

Eine wie im Demoprojekt mit dem `CodecManager` verbundene `TextureView` erhält jeden Frame des Video-Outputs der Wärmebildkamera. Periodisch wird ein Task (Java `Runnable`) ausgeführt, welcher aus der `TextureView` eine Bitmap erstellt, um diese in ein Byte Stream zu speichern. Dieser kann nun der NDK zur Bilderkennung übergeben werden (In Abschnitt Bilderkennung und NDK beschrieben). Der asynchron ausgeführte Task bestimmt somit die Frequenz der Bilderkennung. Das Auslesen und Konvertieren der Bitmap in ein Byte Array benötigt Rechenleistung, welche in einer optimalen Lösung gespart werden könnte. Diese Problemlösung ist jedoch durch das Design des DJI Mobile SDK erzwungen.

```
Bitmap bitmap = textureView.getBitmap();  
  
ByteArrayOutputStream stream = new ByteArrayOutputStream();  
bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);  
byte[] byteArray = stream.toByteArray();
```

Programmausdruck 45: Bitmap aus der `TextureView` erstellen in der Datei `PilotFlightActivity.java`

Dieser Teil der Entwicklung der Android-App warf einige Probleme auf, wodurch sich die Programmierung hinausgezögert hat und nicht wie geplant durchgeführt werden konnte. DJI bietet zwar eine vollständige Dokumentation ihrer API an, in der alle Klassen und Methoden aufscheinen. Sie werden jedoch schlecht bis gar nicht beschrieben und erklärt.¹⁴ Aufgrund dessen muss mit dem Prinzip des Trial-and-Error entwickelt und mehr Zeit investiert werden. Das erwähnte Demoprojekt enthält einen Lösungsansatz zum Verwenden des Video-Streams, dieser wird jedoch weder in der Dokumentation von DJI noch in Kommentaren im Code erklärt.¹⁵ Spezielle Anwendungen, wie es bei BambiGuard mit der Verwendung von OpenCV mit dem NDK der Fall ist, werden außerdem nicht in den zuvor erwähnten Foren und Hilfe-Möglichkeiten behandelt.

14 Siehe <https://developer.dji.com/api-reference/android-api/Components/CodecManager/DJICodecManager.html>
04.03.2022

15 Siehe <https://github.com/DJI-Mobile-SDK-Tutorials/Android-VideoStreamDecodingSample> 04.03.2022

Fehlerbehebungen

Eine Inkompatibilität bei der Verwendung der Mapbox SDK mit gleichzeitiger Implementierung der DJI Mobile SDK führt in Android zu einem Error (Application Not Responding, „ANR“). Bei genauerem Analysieren des Problems fällt auf, dass der Error nur beim Aufruf der Funktion `onDestroy` des Objektes `MapView` auftritt. Da der Anwendungsfall und diese Kombination an Technologien sehr spezifisch ist, ist dieser Fehler nicht in den Dokumentationen oder Protokollen beschrieben. In BambiGuard wurde er gelöst, indem die Funktion `onDestroy` nicht mehr manuell aufgerufen wird. Das ist kein Problem, da die genutzte Version der Mapbox SDK automatisch in den Activity-Lebenszyklus von Android eingebunden wird.

3.5.7 Bilderkennung und NDK

In den folgenden Abschnitten wird das Java Native Interface erklärt und an Beispielen dargestellt. Die Einbindung in eine Android App mithilfe des Native Development Kits wird danach beschrieben und mit der Implementierung in BambiGuard verdeutlicht.

Das **Java Native Interface** (JNI) bietet die Möglichkeit, native plattform-spezifische Methoden in Java aufzurufen. Als „nativ“ werden in diesem Kontext Methoden oder Programmiersprachen bezeichnet, welche generell maschinennäher im Gegensatz zu Java sind und deswegen höhere Performance oder manuelle Speicher-Verwaltung haben. Das JNI ist die Schnittstelle zwischen einem Java-Programm und in diversen anderen Sprachen programmierten Code, wie beispielsweise C/C++. Das Interface schafft den Zugriff auf Bibliotheken, welche anderweitig in Java nicht eingebunden werden könnten. Außerdem verwenden viele Anwendungen, das JNI, um auf Ein- und Ausgabe von Dateien oder Hardware-Funktionalitäten zuzugreifen. Umgekehrt ist es auch möglich, in nativen Methoden Java Programmen auszuführen.

Dazu erweiternd ist das Android Native Development Kit (NDK) eine Sammlung von Werkzeugen, welche das Implementieren von nativen Methoden in eine Android Applikation ermöglichen¹⁶. Native Programmcode wird kompiliert und mithilfe von *Gradle* in die von Android generierte APK (*Android Application Package*) verpackt. Standardmäßig verwendet Android Studio das Kompiliertool *CMake*, obwohl auch andere Werkzeuge eingesetzt werden können. Das NDK und dessen Werkzeuge können von der Android Developer Webseite heruntergeladen werden. Sie sind nicht in der Standard-Installation von Android Studio enthalten.

Im Fall von BambiGuard wurde aus Gründen der Performance und Wiederverwendbarkeit auf die NDK zurückgegriffen. Die in C++ programmierte Bilderkennung kann in iOS und Android gleichen Wege eingebunden werden und muss nicht in eine andere Sprache übersetzt werden.

16 Vgl. <https://developer.android.com/ndk/guides/concepts> 04.03.2022

Ansatz der JNI

Um native Methoden aufzurufen, muss auf der Seite des Java-Programms die Signatur einer Methode das Schlüsselwort `native` beinhalten. Es zeigt dem Compiler, dass die vorliegende Methode das JNI aufruft. Weiters muss das Statement mit einem Semikolon enden, da keine Implementierung in der Java-Klasse möglich sein kann.

```
private native String detectBambisInImage(int w, int h, byte[] img);
```

Programmausdruck 46: Beispiel einer JNI Funktion

Für eine besser Struktur bei der Entwicklung verwendet man auf der Seite der nativen Methode meist Header-Dateien, welche mit der in Java deklarierten Methode verknüpft sind. Dazu stellt der Java-Compiler eine Funktion zur Verfügung, die zu einer kompilierten Java-Klasse die passende h-Datei erstellt. JDK-Versionen bis inklusive JDK 9 verwenden dazu den dedizierten Befehl `javadoc`. Neuere Versionen verwenden `javac` mit der Option `-h`. Der zuvor erwähnte Command ist veraltet¹⁷.

```
javac -h BambiGuardDetector
```

Für Komfort und Geschwindigkeit während der Entwicklung besteht die Möglichkeit Werkzeuge – also Shortcuts für Befehle in der Kommandozeile – in Android Studio zu erstellen. Diese können entweder per Selektion im Kontextmenü einer Datei aufgerufen werden oder global über den Menüpunkt `Tools`. Unter `File/Settings/Tools/External Tools` können Werkzeuge erstellt werden. Dabei stehen diverse Makros wie `$FilePath$` für den Pfad der selektierten Datei oder `$ContentRoot$` für den absoluten Verzeichnispfad des Ordners `app`. In Abbildung 76 wurde ein Werkzeug für den Befehl `javadoc` und ein weiteres für einen folgend erklärten Command erstellt.

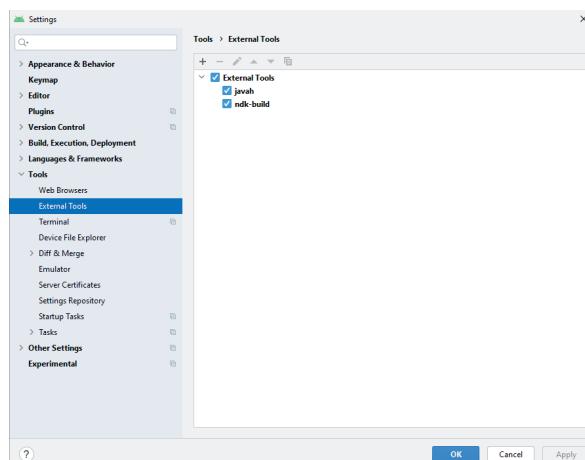


Abbildung 76: Android Studio External Tools

17 Vgl. <https://docs.oracle.com/javase/9/tools/javadoc.htm#JSWOR687> 02.03.2022

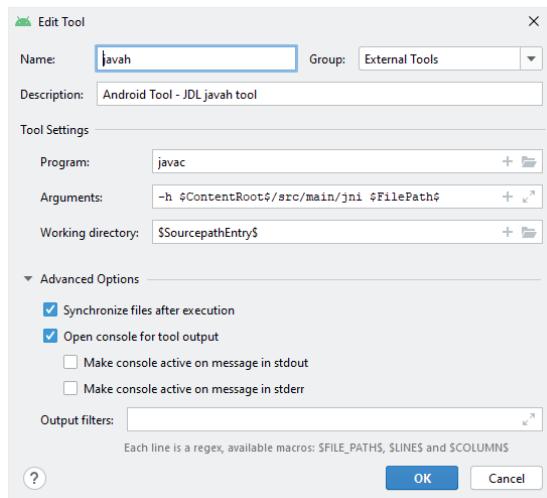


Abbildung 77: Android Studio Tool – javah

Eine mit `javac -h` erstellte Header-Datei sieht wie in Programmausdruck 47 aus.

```
...
JNIEXPORT jstring JNICALL Java_at_ac_szybbs_bambiguard_model_Bambi-
GuardDetector_detectBambisInImage
    (JNIEnv *, jobject, jint, jint, jbyteArray);
...
```

Programmausdruck 47: Beispiel einer Header-Datei erstellt mit dem Befehl `javac -h`

Die Deklaration von JNI-Methoden ist vordefiniert und kann in der offiziellen Dokumentation von Oracle nachgelesen werden. Sie haben immer, zusätzlich zu den Parametern der ursprünglichen Methode zwei weitere Attribute. Einen Pointer zur JNI-Umgebung, `JNIEnv *`, welche Funktionen für den Zugriff auf Java-Objekte bereitstellt und eine Referenz auf das Objekt selbst, `jobject`, vergleichbar mit dem Schlüsselwort `this` in C++. Die Implementierung der Methode schaut somit wie in Programmausdruck 48 aus.

```
...
extern „C“ {
JNIEXPORT jstring JNICALL Java_at_ac_szybbs_bambiguard_model_Bambi-
GuardDetector_detectBambisInImage
    (JNIEnv * env, jobject obj, jint width, jint height, jbyteArray
     byteArrayParam)
{ ... }
...
...
```

Programmausdruck 48: Mit dem Schlüsselwort `extern „C“` gekennzeichnete JNI Methode

Java Native Types

Um auf primitive Datentypen oder Objekte aus Java zuzugreifen, gibt es die Java Native Types. Für jeden primitiven Datentyp gibt es einen äquivalenten Typen in der nativen Methode. So wird aus einem `byte` in der JNI ein `jbyte` und einem `float` ein `jfloat`.

Auch Arrays können nicht direkt referenziert werden, sondern besitzen ihren eigenen Typ, das `jarray`. Die JNI Umgebung bietet eine Möglichkeit zum Auslesen des Arrays (`GetIntArrayElements`) und gängige Array-Operationen (z.B. `GetArrayLength`). Im folgenden Beispiel (Programmausdruck 49) wird ein mitgegebenes Array bestehend aus ganzzahligen Nummern aufsummiert.

```
...
JNIEXPORT jint JNICALL
Java_IntArray_sumArray(JNIEnv *env, jobject obj, jintArray arr)
{
    int i, sum = 0;
    jsize len = (*env)->GetArrayLength(env, arr);
    jint *body = (*env)->GetIntArrayElements(env, arr, 0);
    for (i=0; i<len; i++)
        sum += body[i];
    (*env)->ReleaseIntArrayElements(env, arr, body, 0);
    return sum;
}
...
```

Programmausdruck 49: Beispiel für Array Operationen mit Java Native Types

Konzept der NDK

Das Android Native Development Kit kompiliert und bindet den nativen Programmcode in die Android App ein. Dazu wird eine Datei `Android.mk` benötigt, welche die C/C++ Quelldateien, Parameter und einzubindende Bibliotheken angibt. Sie wird im Ordner `app/main/jni` gespeichert.

```
...
OPENCV_CAMERA_MODULES:=on
OPENCV_INSTALL_MODULES:=on
OPENCV_LIB_TYPE:=SHARED
OPENCVROOT := C:\Users\Clemens\OpenCV-android-sdk
include $(OPENCVROOT)\sdk\native\jni\OpenCV.mk

LOCAL_SRC_FILES := bambiGuardDetector.cpp
LOCAL_LDLIBS += -llog
LOCAL_MODULE := bambiGuardDetector
...
```

Programmausdruck 50: Ausschnitt aus der JNI Konfigurationsdatei `Android.mk`

In Programmausdruck 50 wird unter anderem die Bibliothek OpenCV inkludiert, dessen Shared Object-Dateien (.so) beim Build der App in die APK kopiert werden.

Eine zweite beschreibende Textdatei namens Application.mk kann erstellt werden, um Debugging-Optionen, Ziel-ABIs und STL zu definieren. Ein Application Binary Interface (ABI) ist die Definition von Instruktionen, Speicher verwaltung, unterstützt Formate etc. einer spezifischen CPU¹⁸. Die Angabe von ABIs schränkt den abzudeckenden Umfang der Implementierungsformen ein und ist beim Einbinden einiger Bibliotheken notwendig. Beispielsweise schränkt die DJI SDK die unterstützten ABIs auf armeabi-v7a und x86 ein (in Programmausdruck 51 in Zeile 1 zu sehen).

```
APP_ABI := armeabi-v7a x86
APP_PLATFORM := android-21
APP_CFLAGS += -frtti -fexceptions
```

Programmausdruck 51: Ausschnitt aus der Konfigurationsdatei Application.mk

Die wichtigsten Optionen der Application.mk werden in Tabelle 4 beschrieben.

APP_ABI	Unterstützte Architekturen
APP_PLATFORM	Minimale SDK-Version für die das Programm kompiliert wird
-frtti	Erlaubt das Auslesen von Informationen über den Typ von Variablen (Run-time type information)
-fexceptions	Aktivieren von C++ Exceptions

Tabelle 4: Wichtige Optionen in der Konfigurationsdatei Application.mk (NDK)

Die zwei Konfigurationsdateien werden vom Skript ndk-build verwendet, um den Quellcode basierend auf dem Build-System von CMAKE zu kompilieren. Es kann wie davor beschrieben mithilfe eines IDE Tools manuell oder in der gradle-Datei als Task ausgeführt werden, womit bei jedem Build in Android Studio der native Code automatisch miteingeschlossen wird (zu sehen in Programmausdruck 52).

```
task ndkBuild(type: Exec, description: 'Compile JNI source via NDK')
{
    commandLine "D:/Programme/AndroidStudioSdk/ndk/ndk-build.cmd",
    , NDK_PROJECT_PATH=build/intermediates/ndk',
    , NDK_LIBS_OUT=src/main/jniLibs',
    , APP_BUILD_SCRIPT=src/main/jni/Android.mk',
    , NDK_APPLICATION_MK=src/main/jni/Application.mk'
}
tasks.withType(JavaCompile) {
    compileTask -> compileTask.dependsOn ndkBuild
}
```

Programmausdruck 52: Task zum kompilieren des NDK-Quellcodes in der Datei build.gradle (Modul-Ebene)

Implementierung der Bilderkennung

Die im vorherigen Abschnitt Bilderkennung erstellte Header-Datei `BambiGuard.h` enthält den Algorithmus zur Bilderkennung. Sie muss in das Verzeichnis `/jni` kopiert und in die C++-Datei, welche mit der NDK aufgerufen wird, eingebunden werden. Nach der Übergabe des Byte-Arrays, welches das Bild enthält, muss eine OpenCV `Mat` daraus erstellt werden. Diese kann dann der Bilderkennung als Eingabeparameter mitgegeben werden.

Die Erstellung der `Mat` erzeugt jedoch einige Probleme, da die Datenformate erst aufeinander angepasst werden müssen. Die Bitmap, welche das Byte Array erstellt, speichert seine Bildpunkte in einem bestimmten Format – OpenCV unterstützt zwar einige Formate, wie YUV mit unterschiedlichen Unterabtastungen, RGB, BGR etc., jedoch ist keine Kompatibilität der beiden Komponenten garantiert. Es existieren keine Ressourcen um nachzuschlagen, in welcher Weise man die Bilder am besten überträgt.

Zuerst wurde angenommen, dass das Byte Array manuell in ein Integer Bild umgewandelt werden müsste. Nach einem gescheiterten Versuch eine Methode zur Umwandlung zu implementieren, wurde in der OpenCV Dokumentation genauer recherchiert. Dabei wurde erkannt, dass der `Mat`-Konstruktor standardmäßig Byte Arrays als Pointer akzeptiert. Folgend wurde die Erstellung der `Mat` auf diese Weise implementiert. Nachdem dieser Versuch mit der BambiGuard Header-Datei auch nicht funktioniert und keine Gründe herausgefunden werden konnten, wurde angenommen, dass es an dem Format der Daten liegt. Eine Konvertierung in andere Farbmodelle löst das Problem jedoch nicht.

Das Hauptproblem beim Entwickeln dieses Programmteils liegt darin, dass die Anwendung sehr speziell ist und keine Informationen über Möglichkeiten der Implementierung bereitstehen. Außerdem ist das Debugging innerhalb der NDK schwierig und aufwendig, da Informationen über Errors und Exceptions nur sporadisch ausgegeben werden.

Nach längerer Fehlersuche wurde entschieden, diesen Teil der Anwendung zu überspringen und in der fertigen App die Bilderkennung zu simulieren. Da die einzelnen Komponenten funktionstüchtig sind und nur die Schnittstelle zwischen dem Parsing des Videos mit dem DJI SDK und der Bilderkennung in C++ nicht funktioniert, ist der Kompromiss angesichts des Projektes verkraftbar.

3.5.8 Implementierung des Websockets

Socket.io ist eine JavaScript-Bibliothek, welche das Websocket Protokoll in einer um Metadaten erweiterten Version nutzt, um von Server zu Client zu kommunizieren. Das Serverprogramm wird mit Node.js ausgeführt und ist genauso wie die Client-Bibliothek eventbasiert. Socket.io ist unter anderem für die Sprachen Java, C++, Swift, Dart, Python und .Net verfügbar.¹⁹

19 Vgl. <https://socket.io/docs/v4/> 03.03.2022

Die BambiGuard Android-App nutzt die Java-Implementierung der Bibliothek, da sie somit ohne zusätzlicher Schnittstelle direkt in den Programmcode eingebunden werden kann.

Installation und Initialisierung

Socket.io wird in das Gradle-Build-System eingebunden, indem man den Code aus Programmausdruck 53 in die Datei `build.gradle` der App einfügt.

```
...  
implementation ('io.socket:socket.io-client:2.0.0') {  
    exclude group: 'org.json', module: 'json'  
}  
...
```

Programmausdruck 53: Implementierung der Bibliothek `socket.io` in die Datei `build.gradle` (Modul-Ebene)

Da diese Arbeit kein auslieferbares Produkt als Ziel hat und die Prioritäten von BambiGuard auf Bilderkennung und Automatisierung liegen, wurde die Verschlüsselung der über den WebSocket übertragenen Daten vernachlässigt. Um in Android einen Datenverkehr in Klartext, also unverschlüsselt zuzulassen, muss man das Attribut `usesCleartextTraffic` im Manifest setzen²⁰. Ansonsten lässt die Bibliothek Socket.io die Datenübertragung nicht zu.

```
<application  
    android:usesCleartextTraffic="true"  
    ...  
</application>
```

Programmausdruck 54: Das Attribut `usesCleartextTraffic` in der Datei `AndroidManifest.xml`

Implementierung

Es wird eine generelle Klasse `BambiGuardSocket` geschrieben, welche allgemeine Informationen und Methoden beinhaltet, die bei Pilot und Helfer gleich sind, wie z.B. die URI des Servers (der Server wird im Abschnitt 3.7 Node.js Server detailliert beschrieben). Davon abgeleitet sind je eine Unterklasse für Helfer und Pilot, welche für die Aussendung der spezifischen Daten zuständig sind.

20 Vgl. <https://developer.android.com/guide/topics/manifest/application-element#usesCleartextTraffic> 03.03.2022

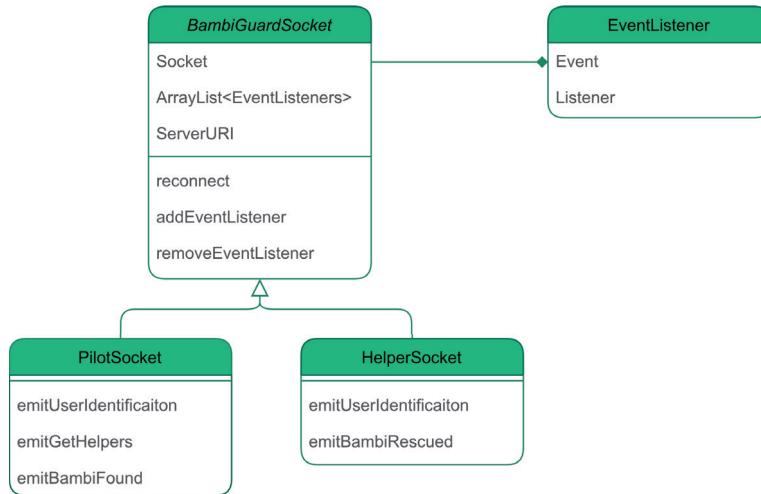


Abbildung 78: UML-Diagramm der Socket-Klassen

Pilot

Der Pilot versucht in der Activity der Flugplanung eine Verbindung zum Socket-Server aufzubauen. Gelingt dies nicht, so wird ein Dialog angezeigt, welcher den Benutzer auffordert, evtl. WLAN oder Mobile Daten einzuschalten. Bei einem Verbindungsfehler wird die Anfrage jede Sekunde erneut versendet.

Sobald die Verbindung aufgebaut wurde, wird dem Server mitgeteilt, dass es sich um einen Piloten handelt (`emitUserIdentification`). Dabei wird der eingegebene Name mitgeschickt, welchen die Helfer bei späterem Hinzufügen sehen können. Folgend liest der Pilot alle 2 Sekunden die aktuell verbundenen Helfer aus (In Programmausdruck 55 ist die beschriebene Funktion `requestHelpersRecurring` zu sehen).

FlightPlanningActivity.java

```

...
private void requestHelpersRecurring() {
    final Handler handler = new Handler(Looper.getMainLooper());
    handler.postDelayed(() -> {
        requestHelpers();
        requestHelpersRecurring();
    }, 2000);
}
...
  
```

Programmausdruck 55: Die Funktion `requestHelpersRecurring` aus der Datei `FlightPlanningActivity.java`

Ein Helper wird hinzugefügt, sobald der Pilot ihn in der Liste der Helper auswählt. Am Ende der Flugplanung wird für jeden hinzugefügten Helper ein entsprechendes Event mit dessen ID an den Server geschickt, welcher es weiterleitet (Event `acquire_helper`).

Während des Flugs sendet das Smartphone des Piloten eine `bambi_found` Nachricht mit den Koordinaten, wo das Rehkitz liegt (zu sehen in Programmausdruck 56). Dadurch bekommen Helper sofort mit, wenn ein Tier gefunden wurde. Das `ViewModel` speichert alle Rehkitze mit deren Position und einer fortlaufenden Identifikationsnummer.

```
...
public void emitBambiFound(Bambi bambi) {
    socket.emit("bambi_found", bambi.getId(),
    bambi.getPosition().getLatitude(),
    bambi.getPosition().getLongitude());
}
...
```

Programmausdruck 56: Emittieren des Events `bambi_found` in der Datei `PilotSocket.java`

Helper

Der Verbindungsaufbau der Helper findet ähnlich statt wie der des Piloten. Beim Starten der Activity versucht der Socket mit dem Server zu kommunizieren, sobald der Helper seinen Namen eingetragen hat, wird wieder das Event `UserIdentification` gesendet.

Wenn der Socket eine Nachricht `bambi_found` bekommt, leitet er die Suche ein und der Kompass zeigt zu der mitgegebenen Position. Ist das Rehkitz nahe genug, kann der Helper es als gerettet markieren, durch das Event `bambi_rescued` wird es dem Server mitgeteilt.

```
viewModel.getSocket().addEventListenner(onBambiFound, "bambi_found");
```

Programmausdruck 57: Auffangen des Events `bambi_found` in der Datei `HelperFlightActivity.java`

3.6 App für iOS

3.6.1 Entwicklungsumgebung Xcode

Apple Xcode

Als Entwicklungsumgebung der iOS-App wird die integrierte Entwicklungsumgebung (IDE) Apple Xcode verwendet. Die Entwicklungsumgebung ist ausschließlich für Apple-Betriebssysteme verfügbar und steht jedem Mac-User im Apple Store kostenlos zur Verfügung. Ein Upload von fertigen Apps auf den Apple Store ist hingegen kostenpflichtig und benötigt Mitgliedschaft im Developer-Programm. Vorrangig richtet sich Apple Xcode an Entwickler, mit den Sprachen Objektive-C oder Swift, jedoch werden auch andere Programmiersprachen wie C/C++ sowie JavaScript und Ruby unterstützt.

Bedeutende Features von Xcode sind der Interface Builder Version 4.0 zur Gestaltung grafischer Benutzeroberflächen (GUIs). Ebenso bietet Apple Xcode eine direkte Dokumentation der API, die einfach aus Xcode zu erreichen ist. Eine weitere Grundfunktion bildet der iPhone Simulator, der eine eingeschränkte Darstellung eines iPhone-Interfaces abbilden und eine Testung am virtuellen Gerät erlaubt. Ein Testen am realen iPhone ist mittels USB-Kabel oder WiFi möglich. Zum Debuggen und Testen stellt das integrierte Programm *Instruments* eine hervorragende Möglichkeit dar, um die Performance von Programmen (Analysefunktionen, CPU-Auslastung, Speicherverbrauch und weitere Funktionen) zu überwachen. Für die App „BambiGuard“ wurden beide Debugging-Methoden verwendet, der Simulator zum Erstellen der Benutzeroberfläche sowie der USB- und WiFi-Debugger für Testzwecke am eigenen iPhone.

Debuggen und Testen

Für die Simulation auf einem virtuellen iPhone sowie das Debuggen mit dem USB-Debugger wird das Gerät in der obersten Leiste ausgewählt. Bei USB-Debugging muss das iPhone mit dem Rechner verbunden sein, um eine Auswahl zu ermöglichen.



Abbildung 79: Auswahl USB-Debugging Xcode

Für WiFi-Debugging muss für neuere iPhones ab iPhone X ein Pass Code erstellt werden (Settings -> Face ID & Passcode-> Turn Pass Code On). Danach muss das iPhone mit Xcode gepaired werden. Dazu wird das iPhone zuerst mit USB-Kabel angeschlossen, danach werden die Simulator Einstellungen geöffnet. (Window -> Devices and Simulators). Um den Vorgang abzuschließen, muss die Checkbox „Connect via network“ ausgewählt werden, (siehe Abbildung 80) während des Vorgangs muss sich das iPhone im gleichen Netzwerk wie das MacBook oder der Mac befinden.



Abbildung 80: WiFi-Debugger Xcode iPhone

3.6.2 Benutzeroberfläche

Bei der Implementierung der iOS-Benutzeroberfläche baut man auf das App-Layout und die App-Struktur im davor gefertigten Wireframe- und MockUp-Design auf. Die Views werden direkt im Main.storyboard angefertigt.

Ressourcen

In Assets.xcassets werden alle relevanten Design-Elemente (Farben, Icons, Logo-Varianten) gespeichert, welche in der gesamten App verwendet werden können. Die Farben stammen aus dem teameigenen CICD-Manual, zu finden im Kapitel Repräsentation. Vorteil dieses gemeinsamen Design-Katalogs sind einheitliche, simple Gestaltungsmuster. Die Grafiken und Icons, welche für die App benötigt werden, können für die unterschiedlichen Größen und Bildformate aus Adobe Illustrator exportiert werden und direkt hineingezogen werden.

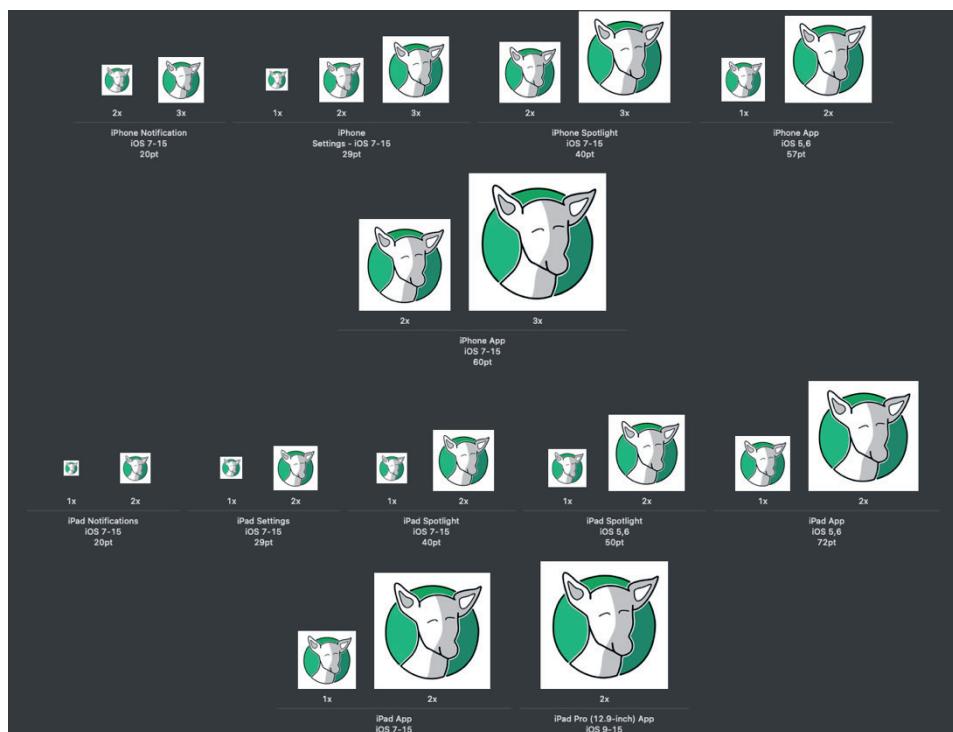


Abbildung 81: Das App Icon in verschiedenen Größen in Assets.xcassets

View Controller

Für jede Ansicht im MockUp wird in der App ein eigener Viewcontroller erstellt. Die Ansichten werden überwiegend im `Main.storyboard` über die grafische Ansicht erstellt und mit Properties und Constraints versehen.

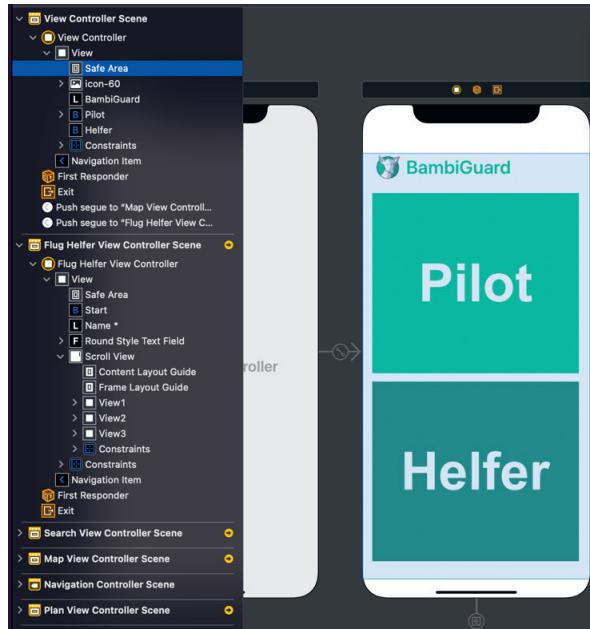


Abbildung 82: Safe Area im View Controller

Zur Navigierbarkeit als auch zur besseren Übersicht wurde über die gesamte App ein Navigation Controller hinzugefügt. Dieser kümmert sich um den reibungslosen Ablauf innerhalb der gesamten Ansichten undbettet selbstständig eine Titelleiste zur Rückkehr zur vorherigen Ansicht ein.

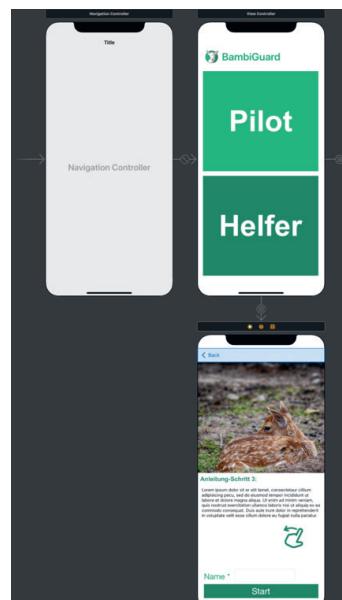


Abbildung 83: Navigation Controller – Ablaufsverwaltung

Map – und Plan View Controller

Durch das zeitversetzte Beginnen mit der iOS-App, hatte die Trennung der Flugplanungs-Ansicht in zwei eigenständige Ansichten keine Auswirkungen auf die iOS-App. Für den Map View Controller wird eine MKMapView verwendet, hier können später die Feldeckpunkte gesetzt werden. Im Plan View Controller befindet sich die Namenseingabe mit einem TextField und Label. Darunter, bei den verbundenen Helfern, wird eine Table View eingebettet, die später mit der Netzwerkkommunikation befüllt wird. Die Ansicht in Xcode wird leider falsch dargestellt, daher überdeckt die Table View-Ansicht im Screenshot das TextField und das Label. Weiters befinden sich noch zwei Switch-Buttons auf dem Controller sowie einige Labels.

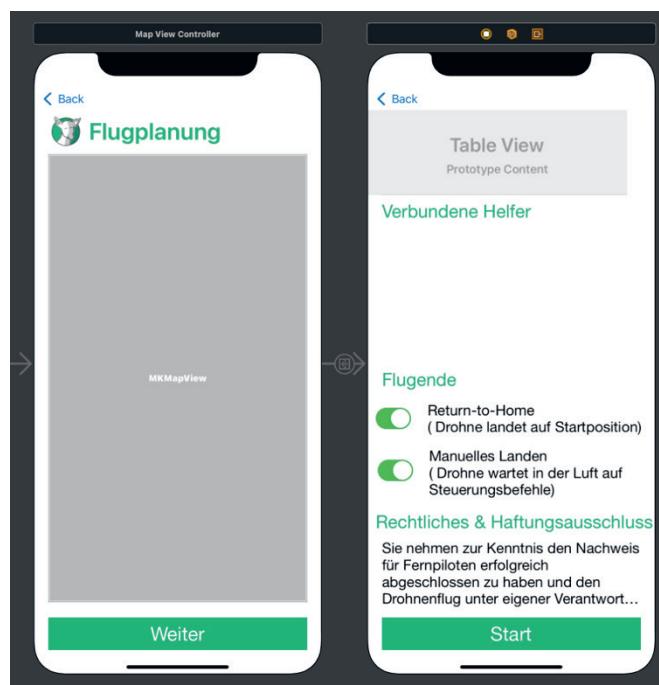


Abbildung 84: Map und Plan View Controller in Xcode

Flug View Controller

Beim Flug View Controller wechselt der Portrait-Mode automatisch in den Landscape-Mode. Bei der Ansicht sollte die DJI UX SDK verwendet werden. Bei der Einbindung der SDK trat in Xcode ein Fehler auf, wodurch die SDK bis zum Projektende keine Einbindung mehr zuließ.

Flug Helper View Controller

Das Helper-Tutorial wird mit einer ScrollView umgesetzt, welche mehrere Views enthält und daher durch „Swipen“ Informationen gesammelt werden können.

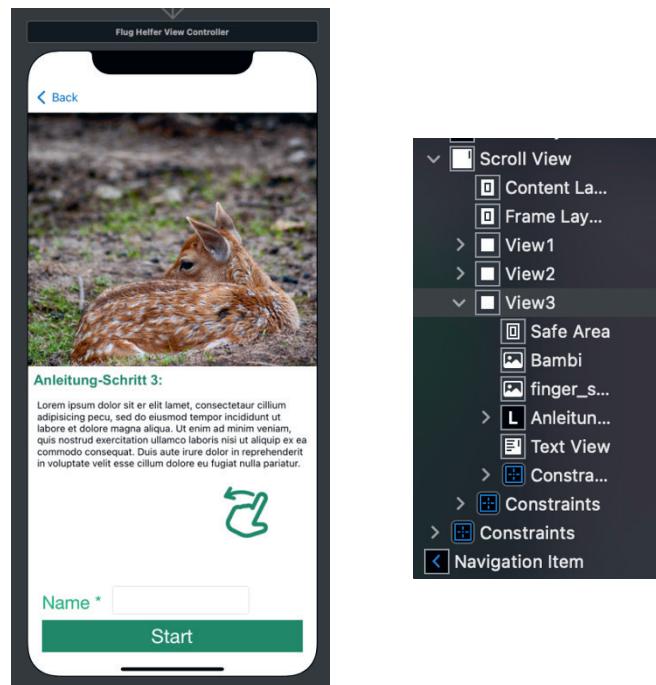


Abbildung 85: Flug Helper View Controller in Xcode

Search View Controller

Der Helper wird mittels Pfeil in die Richtung des Rehkitzes geleitet. Die Grafik rotiert, je nachdem in welcher Richtung das Tier liegt.

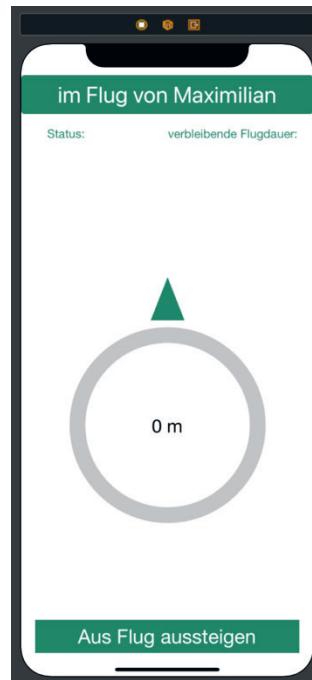


Abbildung 86: Search View Controller in Xcode

3.6.3 Arbeiten mit dem DJI SDK

Zur Flugsteuerung in der iOS BambiGuard-App wird das vom Hersteller kostenlos zur Verfügung gestellte DJI SDK (Software Development Kit) verwendet. Das unternehmenseigene Mobile SDK von DJI bietet Entwicklern eine Möglichkeit ihre eigenen Apps mit Basisfunktionen von DJI zu verwenden. Zur Programmierung wird nicht nur das Mobile SDK benötigt, sondern auch das UX SDK, die typische DJI-Icons beinhaltet, welche in der App genutzt werden können. Für die Nutzung des SDK wird ein Account und ein Produkt von DJI benötigt. Der Download ist für Apps im privaten, aber nicht kommerziellen Bereich kostenlos.

iOS Mobile SDK und UX SDK

Die Bereitstellung der Funktion ist für Android und iOS identisch, ebenso wie die Hilfe und der Support (Siehe 3.5.6 Arbeiten mit dem DJI SDK).

Einrichtung

Um die SDK in Xcode nutzen zu können, muss man sich auf der DJI-Developer-Website²¹ registrieren. Danach muss für jede App, die mit einer DJI-SDK verwendet wird, ein App-Key erstellt werden. DJI überprüft so die Anwender ihrer Produkte, um im Haftungsstreit eine identifizierbare Person zu erhalten.

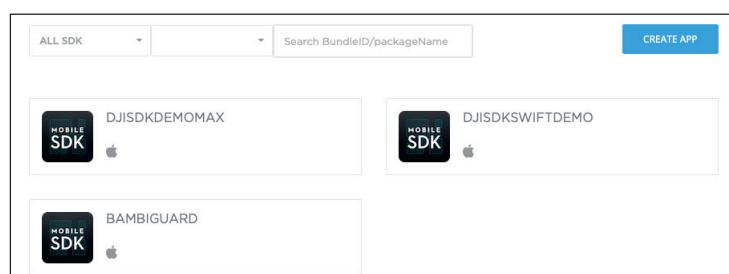


Abbildung 87: App-Keys für mehrere DJI-Apps (DJI Developer)

APP INFORMATION	
SDK Type	Mobile SDK
APP Name	BambiGuard
Software Platform	iOS
Bundle Identifier	maximilianstrobl.at.BambiGuard
App Key	813588888fcc9f8d26af6157
Category	Agricultural applications
Description	Wildlife rescue
EDIT	DELETE

Abbildung 88: iOS App-Key für die BambiGuard App

21 <https://developer.dji.com/user/apps/#all> 06.03.2022

Anhand des Bundle Identifier wird die Verifizierung durchgeführt. Der Bundle Identifier kann im Xcode Projekt unter General -> Identifier nachgelesen werden. Nach erfolgreicher Erstellung muss der erzeugte App.Key in dem File „info.plist“ eingetragen werden. (siehe Abbildung 89)

Key	Type	Value
Information Property List	Dictionary	(4 items)
> App Transport Security Settings	Dictionary	(1 item)
> Application Scene Manifest	Dictionary	(2 items)
Enable Multiple Windows	Boolean	NO
> Scene Configuration	Dictionary	(1 item)
DJISDKAppKey	String	813588888fcc9f8d26af6157
> Supported external accessory protocols	Array	(3 items)

Abbildung 89: App-Key in Xcode eintragen

Cocoapods DJI SDK

Die Installation und Einbettung der DJI SDK in Xcode wird mit dem Abhängigkeitsmanager „Cocoa-Pods“ durchgeführt. CocoaPods ist ein Abhängigkeitsmanager auf Anwendungsebene für Objective-C und Swift. Der Manager muss mit folgenden Befehlen im Terminal installiert werden.

```
sudo gem install cocoapods
nano Podfile
pod install
```

Nach erfolgreicher Installation der DJI SDK im Projekt sollte die Ordnerstruktur ähnlich dieser Abbildung aussehen.

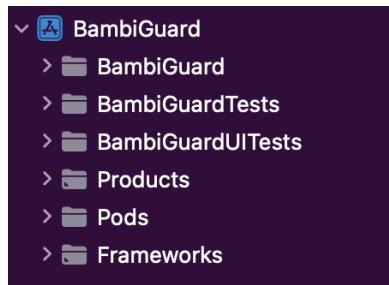


Abbildung 90: Ordnerstruktur nach CocoaPods-Installation

Die Einbindung in die App BambiGuard hat gut funktioniert, jedoch konnte man mit Verbindung der Drohne keine Simulation durchführen, da Xcode die Pod-Files eines nach dem anderen sperzte.

3.7 Node.js Server

Um das System von Smartphones der Helfer und des Piloten, wie in Abschnitt 3.2.2 Tech-Stack beschrieben, umzusetzen, wählte man einen Webserver als Zwischeninstanz. Die Plattform Node.js und deren Programmiersprache JavaScript wurde einerseits im Unterricht geübt und angewandt, andererseits ist diese Konfiguration gebräuchlich, sodass viele Ressourcen und Dokumentationen angeboten werden.

3.7.1 Node.js

Node.js-Implementierungen basieren auf Modulen, welche durch den *Node Package Manager* (npm) installiert werden. Man lädt erst bei der Erstellung eines Projekts und der Verwendung bestimmter Komponenten die Verbindlichkeiten herunter. Eine Vielzahl an Modulen, welche alle auf Node.js aufbauen, bieten diverse Einsatzmöglichkeiten.

Module

Für die Implementierung des Webservers wird das Modul Express verwendet. Es ist eine kleine Bibliothek für Web-Applikationen oder APIs. Dieses Modul ermöglicht eine einfache Verwaltung eines Web-Services. Die Definition der Abhängigkeiten ist in Programmausdruck 58 zu sehen.

```
{  
  "name": "bambiguard-server",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "dependencies": {  
    "express": "^4.17.1",  
    "socket.io": "^4.1.3"  
  },  
  "author": "Clemens Losbichler",  
  "license": "ISC"  
}
```

Programmausdruck 58: Definition der Abhängigkeiten in der Datei package.json

Initialisierung

Um eine Serverinstanz zu erstellen, importiert man die Module express, http und socket.io und erstellt ein socket.io-Objekt, welches auf der Instanziierung eines http-Servers basiert.

```
const express = require("express"),
http = require('http'),
app = express(),
server = http.createServer(app),
io = require('socket.io')().listen(server);
```

Programmausdruck 59: Importieren von Modulen in der Datei index.js

Mit dem Objekt `io` kann nun ein Event-Listener erstellt werden, welcher aufgerufen wird, sobald sich ein Client über den Websocket verbindet (Event „connection“ in Programmausdruck 60 zu sehen).

```
io.on("connection", (socket) => {
  console.log(`user connected`);
})
```

Programmausdruck 60: Auffangen des Events connection in der Datei index.js

Der mitgegebene Parameter `socket` ist die Instanz, über die man mit dem Client kommuniziert. Man kann damit auf weitere Events hören oder selbst welche senden.

Mit der Funktion `listen(port, callback)` des `http`-Objekts startet man den Server, bei Erfolg wird die definierte Callback-Funktion ausgeführt (Beschrieben in Programmausdruck 61). Es wurde Port 3000 gewählt, da dieser nicht „well-known“ ist und daher nicht die Gefahr besteht, dass er von anderen Anwendungen genutzt wird.

```
server.listen(3000, () => {
  console.log(`BambiGuard Server is running on port 3000`);
})
```

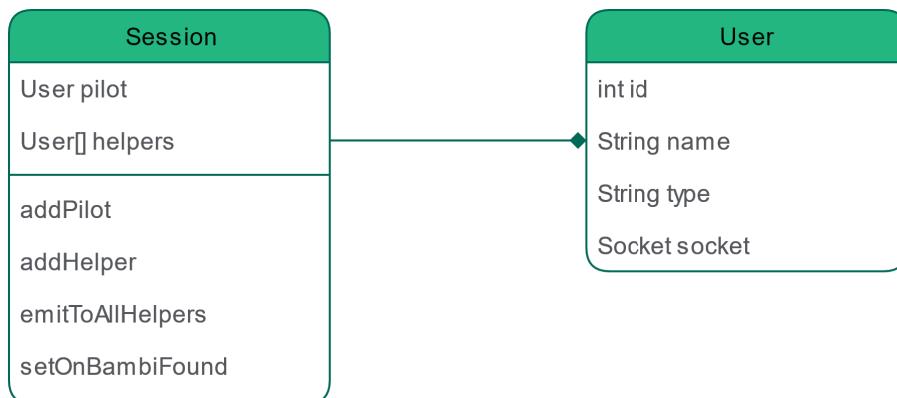
Programmausdruck 61: Mit dem Node.js Server auf einen bestimmten Port hören; Datei index.js

Um den Node.js-Server zu starten, setzt man den Befehl `node <source-file>` ab.

```
node index.js
```

3.7.2 BambiGuard Implementierung

Die Datenstruktur des Servers besteht aus Sessions, welche jeweils einen Piloten und eine Liste von Helfern enthalten. Alle Sessions werden global in einer Liste festgehalten. Sobald sich ein neuer Client als Pilot identifiziert, wird für ihn eine neue Session erstellt. Somit hat jede Session immer einen Piloten und jedem Piloten gehört immer eine Session. Wenn sich ein Helfer verbindet, wird dieser in eine globale Liste gespeichert, aus welcher alle Piloten ihre Helfer auswählen und ihrem Flug hinzufügen können.

**Abbildung 91:** UML-Diagramm des Servers

```

class Session {
    constructor(pilot) {
        this.addPilot(pilot);
        this.helpers = [];
    }
    ...
    emitToAllHelpers(event, args) {
        this.helpers.forEach((helper) => {
            helper.socket.emit(event, args);
        });
    }
}

```

Programmausdruck 62: Die Klasse Session in der Datei index.js

Testing

Da das Testen der Anwendung in einer authentischen Umgebung sehr aufwendig ist – man braucht mindestens zwei Smartphones, um einen Piloten und einen Helfer zu haben – werden die entsprechenden User am Server durch das Senden von Events simuliert. Die Funktion `testHelper` in Programmausdruck 59 fügt den Helfer dem virtuellen Piloten mit dem Namen „Wolfgang“ hinzu.

```

function testHelper(helper) {
    helper.socket.emit("join_flight", "Wolfgang");
    ...
}

```

Programmausdruck 63: Simulieren eines Piloten, um die Funktionalität des Helfers zu testen; Datei index.js

3.7.3 Distribution

Um nun eine kontinuierliche Kommunikation zu gewährleisten, muss der Webserver permanent online sein. Die Entwicklung auf einem Endgerät, wie einem Windows-PC ist rein für den Zweck des Testens und Debuggings. Aus diesen Gründen wurde der Server auf einem Raspberry Pi, einem Linux Debian Computer aufgesetzt. Dieser befindet sich im Heimnetzwerk von Clemens Losbichler, für welches Port-Forwarding eingesetzt wird, um den Dienst öffentlich zugänglich zu machen.

Weiterführend könnte beispielsweise ein Server in dafür dedizierten Umgebungen gemietet oder gekauft werden, um das System zu verbessern, also schneller und ausfallsicherer zu sein. Eine solche Notwendigkeit war für dieses Projekt jedoch nicht gegeben.

Installation

Aufgrund der Modularität von Node.js und dessen Packetmanager npm, kann Software besonders leicht auf neuen Geräten installiert werden. Dafür müssen nur der Anwendungscode und die Konfigurationsdateien `package.json` und `package-locks.json` auf die neue Maschine kopiert werden. Mit dem Befehl `npm install` werden alle Verbindlichkeiten, welche in den Konfigurationsdateien definiert werden, in das Verzeichnis `node_modules` installiert. Im Fall von Bambi-Gaurd sieht das Verzeichnis des Servers nun wie folgt aus.

```
pi@raspberrypi:~/BambiGuard $ ls
index.js  node_modules  package.json  package-lock.json
```

Port-Forwarding und DDNS

Da der Computer in einem Heimnetzwerk steht und keine eigene öffentliche IP-Adresse besitzt, muss Port-Forwarding am Gateway eingesetzt werden. Damit leitet in diesem Fall das Modem alle Pakete, welche einen bestimmten Port adressieren an den angegebenen Host weiter.

BambiGuard		<input checked="" type="checkbox"/> Bearbeiten	<input type="checkbox"/> Löschen
Weiterleitungsname:	BambiGuard		
Anwendung:	Port 3000		
Anwendung Portzuordnung hinzufügen			
Internal mode:	Host mode		
Interner Host:	raspberrypi-Ethernet1		
Gerät hinzufügen			
		<input type="button" value="Abbrechen"/>	<input type="button" value="Speichern"/>

Abbildung 92: Port-Forwarding des Modems

In Abbildung 92 werden alle Nachrichten des Ports 3000, auf welchem der Webserver betrieben wird an den Raspberry Pi weitergeleitet.

Üblicherweise vergeben Internet-Provider keine fixen IP-Adressen an private Haushalte, die öffentlichen Adressen können sich ändern. Deswegen setzt man sogenanntes Dynamisches DNS (DDNS) ein, um nicht mit der IP selbst, sondern mit einer Domain auf den Server zuzugreifen. Ein Host aus dem Netzwerk schickt regelmäßig eine Nachricht an den DDNS-Anbieter, welcher die Domains verwaltet, um die IP-Adresse bekanntzugeben. Damit stellt man sicher, dass ein Dienst immer über die entsprechende Domain erreichbar ist, egal welche Netzwerkadresse dieser hat.

Es wurde der DDNS-Anbieter No-IP und eine ddns.net-Domain verwendet. Helfer und Piloten können mit einer einfachen Internetverbindung von überall aus auf den Server zugreifen.

Hostname ▾	Last Update	IP / Ziel	Type	
.ddns.net <small>Active</small>	Jan 9, 2022 07:53 PST	192.164. .	A	Ändern

Abbildung 93: DDNS-Eintrag von No-IP

PM2 Daemon

Um das Programm nun permanent als Service auszuführen, verwendet man in Linux sogenannte Daemons, also Tasks, welche im Hintergrund abgearbeitet werden. Um diese nicht selbst erstellen und verwalten zu müssen, verwendet man Prozessmanager wie z.B. pm2. Dieser startet die Applikation neu, wenn sie abstürzt, und kann bei Bedarf auch Load-Balancing vornehmen. Der Vorteil von pm2 ist die einfache Handhabung und der daraus entstehende Komfort. Der folgende Befehl startet den Daemon für den BambiGuard-Server.

```
$ pm2 start index.js
```

3.8 Repräsentation

Ein wesentlicher Faktor zur Bekämpfung des Mähtodes ist es mehr Aufmerksamkeit auf dieses Thema zu lenken. Ziel der Bewusstseins- und Aufmerksamkeitsgenerierung ist es, mehr zu den betroffenen Personen durchzudringen und über dieses Problem aufzuklären und zu informieren, wie sie effektiv Mähtode vermeiden und zum Schutz von Wildtieren beitragen können.

Um die Menschen auf BambiGuard aufmerksam zu machen, werden mehrere Werbematerialien und Medien verwendet. Da viele Menschen in der heutigen Zeit online sind und auch oft über mehrere Stunden pro Tag die Sozialen Netzwerken nutzen, ist es sinnvoll, auf diesen ebenfalls die App BambiGuard zu bewerben.

3.8.1 Design der Marke BambiGuard

Der Name BambiGuard soll die Empfindlichkeit der Rehkitze schildern, welche durch das Konzept der Arbeit beschützt werden. Das Wort „Bambi“ wird stark emotional mit einem jungen Reh verbunden, welches sich selbst nicht schützen kann. „Guard“ bedeutet so viel wie „Wächter“ oder „Beschützer“ und symbolisiert den hohen Schutzfaktor, welchen das Konzept der Diplomarbeit bietet.

Die Marke BambiGuard als solche soll mit Tierschutz, Natur und Nachhaltigkeit verbunden werden. Aus diesen Gründen ergeben sich die in der Farbauswahl ermittelten Farben und es wird bei der Wahl von Herstellern für Druckprodukte auf Regionalität und Nachhaltigkeit geachtet.

Das BambiGuard Branding soll in den späteren Schritten und erstellten Produkten prägnant und repräsentativ wirken, um somit den Wiedererkennungsfaktor zu erhöhen.

Farben

Die Marke BambiGuard soll auch mit einer bestimmten Farbkombination verbunden werden. Da sich die Tiere oft im Gras verstecken, wurden verschiedene Grüntöne kombiniert und mit einem neutralen Grauton komplettiert, welche die Naturbeschaffenheit widerspiegeln sollen. Die Farben finden sich in allen Elementen der Diplomarbeit wieder und schaffen somit eine starke Bindung. Angefangen von der App bis zu den gedruckten Flyern zieht sich dieses Farbschema durch. Folgende Farben aus Tabelle 5 wurden ausgewählt.

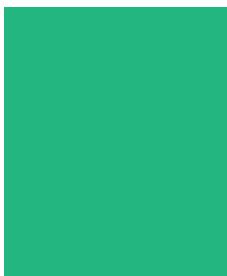
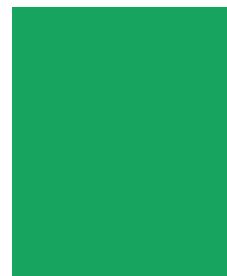
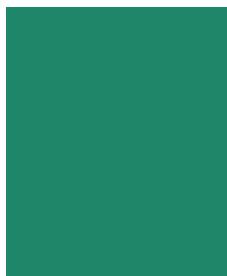
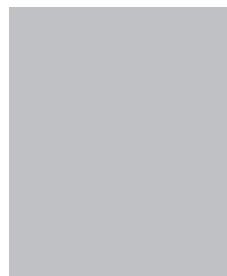
Pantone 2250 U	Pantone 355 U	Pantone 3415 U	Pantone Cool Gray 3 U
			
RGB 41 177 123	RGB 24 151 94	RGB 48 129 103	RGB 197 197 197
HEX #29b17b	HEX #18975e	HEX #308167	HEX #c5c5c5

Tabelle 5: BambiGuard Farben

Schriftarten

Es wurden zwei verschiedene Schriftarten ausgewählt, welche unterschiedliche Zwecke erfüllen sollen.

Die Schriftart **Recherche Regular** wird aufgrund ihres Stils für den Schriftzug „Bambi“ verwendet und soll die Verwundbarkeit der Tiere mit ihrer verspielten und feinen Art darstellen.



Beispieltext
1234567890

Abbildung 94: Schriftmuster „Recherche Regular“

Die Schriftart **Mr Eaves Mod OT Regular** wird für den Schriftzug „Guard“ verwendet und symbolisiert die Genauigkeit und Stärke des Systems BambiGuard. Zusätzlich wird diese Schriftart aufgrund ihrer einfachen Lesbarkeit für Fließtexte in den Infomaterialien verwendet.



Beispieltext
1234567890

Abbildung 95: Schriftmuster „Mr Eaves Mod OT Regular“

Logo

Das Logo wurde in Affinity Designer auf einem iPad erstellt, da dieses das schnelle und einfache Umsetzen einer Idee ermöglicht. Es wurden mehrere Ideen umgesetzt und fortschreitend verfeinert. Die Grundidee des Logos beinhaltet ein Reh, weil dieses Kern dieser Diplomarbeit ist und mit dem Betrachter eine Bindung schaffen soll. Das Logo soll zudem nicht zu kompliziert wirken und einen hohen Wiedererkennungswert besitzen. Der erste Entwurf des Logos soll die Grundidee darstellen und noch keine Farben beinhalten, weil diese erst für das finale Design relevant sind.

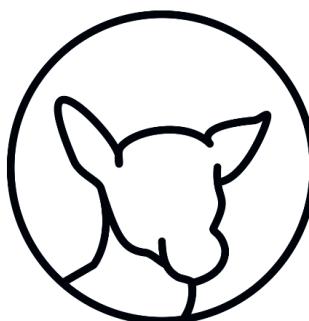


Abbildung 96: Erster Entwurf des Logos

Anschließend wurden alle Kritikpunkte der befragten Medientechnik Professoren und Schüler, welche die erste Version betrafen, aufgelistet und adressiert, folgende Punkte wurden angeführt:

- Kopf des Tieres ist zu klein
- Strichstärke ist zu hoch
- Verwechslungsgefahr mit anderen Tieren/ nicht erkennbar, dass es sich um ein Reh handelt
- Starke Ecke am linken Teil des Halses

Die daraus resultierende zweite Version des Logos stellt eine Erweiterung der ersten dar und soll alle Kritikpunkte beheben. Der Kopf wurde vergrößert und schmäler gemacht, um mehr dem eines Rehs zu ähneln. Zusätzlich wurden die Ohren leicht angepasst und stehen nun über den Kreis hinaus, um somit nicht fälschlicherweise darzustellen, dass das Tier eingesperrt wird oder es sich hierbei um ein Tier handelt, welches sich im Visier eines Jägers befindet.

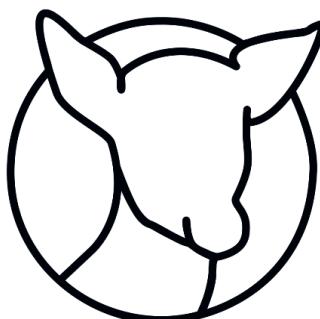


Abbildung 97: Korrektur des ersten Entwurfs des Logos

Anschließend wurden einige Farbkombinationen für das Logo gewählt, welche die Diplomarbeit repräsentieren sollen.



Abbildung 98: Verschiedene Farben am Entwurf des Logos

Versionen, welche Farbverläufe beinhalten, sollen die Charakteristiken einer Wärmebildkamera darstellen. Die Entscheidung fiel jedoch gegen eine solche Lösung, weil ein bunter Farbverlauf der Simplizität des Logos schadet und nicht die Farben beinhaltet, welche zuvor ausgewählt wurden.

Nachfolgend wurden kleine Änderungen vorgenommen, wie etwa die Anpassung des Kreises, dieser soll nicht die Ohren des Rehs berühren. Es wurden die inneren Teile der Ohren und Augen hinzugefügt, um den Erkennungswert weiter zu erhöhen. Zusätzlich wurde eine Teilung des Gesichts in Weiß und Grau vorgenommen, um dem Logo mehr Dynamik zu verleihen.



Abbildung 99: Logo Schattierung durch Grautöne

Anschließend wurden wiederrum Kritikpunkte gesammelt, welche daraufhin adressiert wurden. Folgende Mängel wurden beanstandet:

- Grüne Füllung des Kreises schneidet die Ohren
- Keine Verwendung der verschiedenen Grüntöne
- Falscher Grauton wurde verwendet
- Weitere Abgrenzung zwischen Kreis und Kopf nötig

Der Abstand zwischen Hintergrund und Kopf wurde geschaffen, indem eine größere Version des Kopfes erstellt und vom Hintergrund subtrahiert wurde. Anschließend wurde der Hintergrund in drei Teile unterteilt, um zu ermöglichen, dass drei verschiedene Farben verwendet werden können. In Abbildung 100 sind alle Ebenen des Logos sichtbar.



Abbildung 100: Das Logo in Affinity Designer

Die Füllung des Kreises besteht wie bereits zuvor erwähnt aus drei einzelnen Teilen in den verschiedenen Grüntönen. Der Kopf besteht aus drei Elementen, den äußeren Linien und der grauen, sowie der weißen Fläche. Dies wurde erzielt, indem der Kopf einmal gesamt erstellt, dreimal dupliziert und an der Stelle des Farbübergangs geteilt wurde. Die Augen und Ohren sind einfache Vektorkurven, welche mit einer Farbe gefüllt wurden.

Die finale Version des Logos beinhaltet nun alle zuvor festgelegten Farben und existiert für verschiedene Verwendungszwecke in den Formaten PDF, JPG, PNG und SVG. Zusätzlich wurde das Logo mit transparentem, schwarzem und weißem Hintergrund exportiert.



Abbildung 101: Finale Version des Logos

Die Entwicklung des Logos ist in Abbildung 102 zusammengefasst.

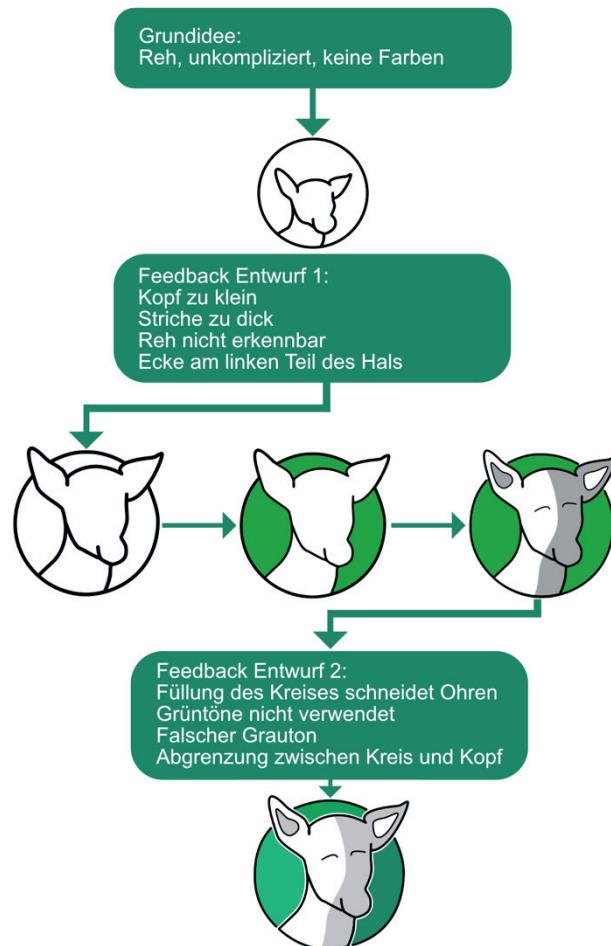


Abbildung 102: Logo Entwicklungsprozess

3.8.2 Website

Die BambiGuard-Website stellt in der Diplomarbeit eine Informationsplattform für Interessierte dar und ist ein zentraler Vermarktungsweg der Diplomarbeit. Die Website begrenzt sich auf vier Hauptseiten (Home / Produkt, Über uns, Kontakt sowie Impressum), welche eigene Aufgaben und Informationen enthalten. Entwickelt und umgesetzt ist die Website mit dem Web-Framework React.

Wireframe

Die Seite Home / Produkt bildet das Grundfundament für den Informationsfluss, hierbei beschäftigen sich die Inhalte mit den Fragen: „Was ist BambiGuard?“, „Welche Aufgaben und Ziele wurden gesetzt?“, „Welche Fakten gibt es zur Problematik?“ und „Wo erfahre ich mehr über die Diplomarbeit und das Projekt?“.

Grundsätzlich lässt sich das Wireframe in drei Bereiche teilen: dem Headerbereich mit Navigationsbar, dem Inhaltsbereich und der Fußzeile (Footer). Am Beginn der Website, für alle Seiten und Unterseiten gleich, bildet die Navbar mit links zu Beginn das Logo sowie auf der rechten Seite die Navigierpunkte zu den Unterseiten. Ebenfalls für alle gleich ist der Footer. Hierbei teilt sich der erste Teil des Footers in vier Blöcke. In diesen Blöcken ist ein Block für die Zusammenfassung von BambiGuard, von dort ausgehend gelangt man zu den Kernbotschaften. Der zweite Bereich ist unseren Partnern gewidmet (IT-HTL, VFI). Als dritten Bereich wird eine Kontaktmöglichkeit integriert, die auf das Kontaktformular weiterleitet. Der letzte und vierte Bereich in der Fußzeile dient zur Informationserweiterung und bietet Direktlinks auf unsere Social Media-Accounts. Im zweiten Abschnitt des Footers befindet sich das Logo, das Erstellungsjahr der Website sowie die beiden Social-Media Logos von Facebook und Instagram in einer Zeile.

Auf der Home-Seite entschied man sich eine möglichst hohe Aufmerksamkeitsgenerierung aufzubauen. Im Hintergrund der Headersektion wird ein Video in Dauerschleife laufen. Darüber liegt eine Überschrift mit einer untergeordneten Erklärwortgruppe. Unter diesen beiden sollen zum einen ein direkter Link auf die Willkommenssektion eingebunden werden, sowie der Button zum Abspielen des Infovideos. Nach der Headersektion wechseln sich Content-Bereiche mit Window-Pictures ab bis zum Footer (Ende der Seite). Siehe dazu die nachfolgenden Abbildungen zum Home-Wireframe.

Die Über uns-Seite gliedert sich wie auch die anderen Seiten in drei Bereiche, wobei der Navigations-Bereich sowie der Footer-Bereich bei allen Unterseiten gleichbleibt. Der Inhaltsbereich der Über uns-Seite teilt sich in drei Spalten, je eine für ein Teammitglied. In jeder Spalte eines Teammitglieds befindet sich ein Teammitgliedsfoto, sein Name, seine Arbeitsbereiche sowie seine genaue Ausformulierung seiner Bereiche in der Diplomarbeit. Siehe nachfolgende Abbildungen zum Über uns-Wireframe.

Auf der Kontakt-Seite im Inhaltsbereich befinden sich zum einen ein Kontaktformular und zum anderen eine Kartenansicht zum Schulstandort sowie die genaue Adresse und unsere gemeinsame BambiGuard E-Mailadresse. Eine genauere Übersicht hierzu können sie aus dem Kontakt-Wireframe nachfolgend erkennen.

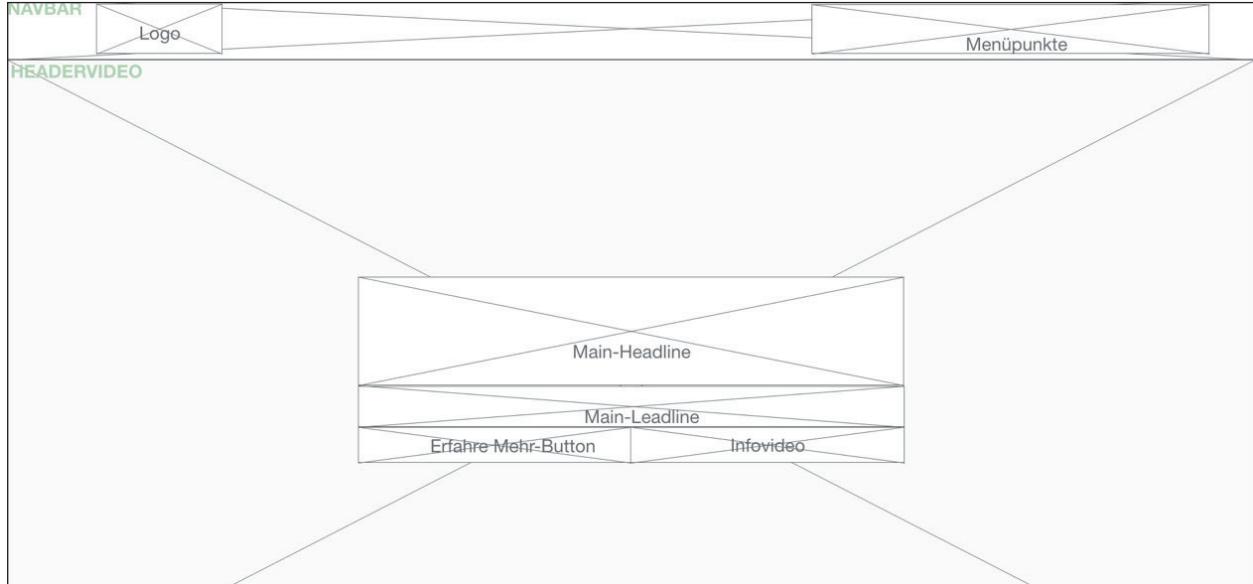


Abbildung 103: Website - Wireframe (Home; oberer Teil)

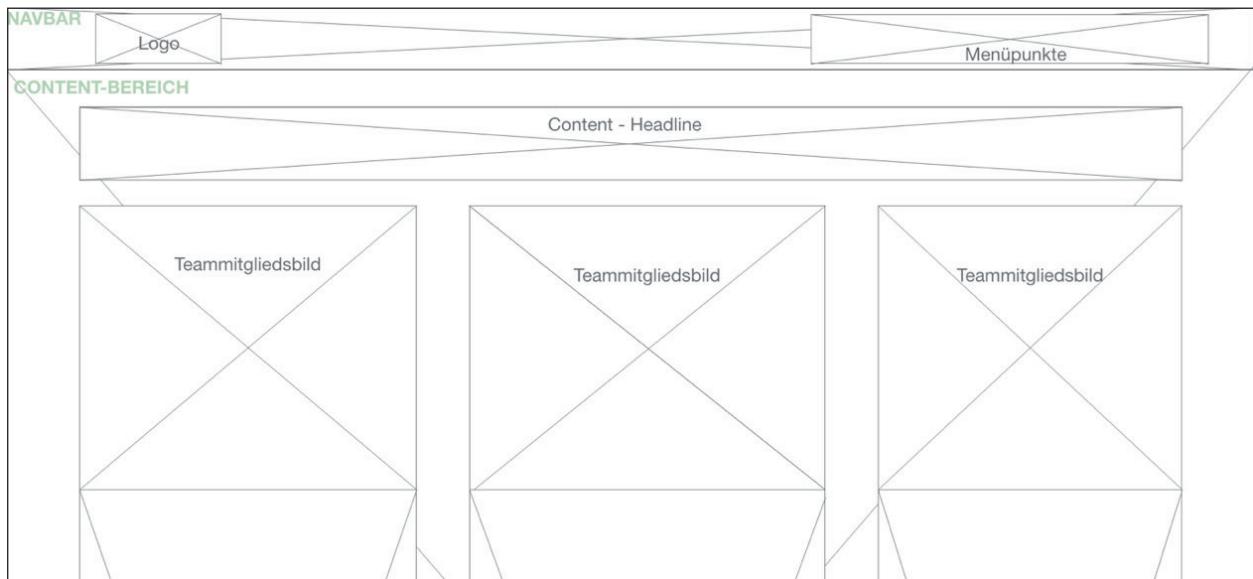


Abbildung 104: Website - Wireframe (Über uns; oberer Teil)

MockUp

Das Design der BambiGuard-Website wird in der primären CICD-Farbe (Grün) gestaltet. Bei der Entwicklung wird Augenmerk auf die CICD-Guideline der Marke BambiGuard gelegt und die Farben sowie die Bilder harmonisch ausgewählt und angepasst.

Der MockUp-Entwurf der Website baut auf den Wireframe-Strukturen auf, daher wird das Design der Navbar und des Footers bei allen Unterseiten einheitlich aussehen. Die Navbar wird in der CICD-Farbe Grün gehalten, auf der linken Seite der Navbar ist das Logo von BambiGuard eingebunden und

mit weißem Schriftzug versehen. Die Menüpunkte der Navigationsbar sind auf grünem Hintergrund mit weißer Schrift gestaltet, bei Auswahl des Menüpunkts, wird der untere Rahmen des Menüpunkts weiß hinterlegt. Sollte die Ansicht der Menüpunkte nicht passend oder zu klein sichtbar sein, wie etwa in der Smartphone-Ansicht, schieben sich die Komponenten zu einer Hamburgerübersicht zusammen. Die Hamburgerübersicht stellt sich durch drei parallele Striche als Icon dar und ermöglicht eine größere Ansicht der Menüpunkte am Smartphone.

Der Footer hält sich ebenfalls in den Farben Grün und Weiß. Der gesamte Footer-Container hält sich im grünen Hintergrund, die Schriftzüge, die Icons für Social Media, die Überschriften und Unterüberschriften erschienen in der Farbe Weiß. Zwischen den weißen Schriftelementen werden unterschiedliche Schriftgrößen und Schriftstärken verwendet, anhand dessen man die Bedeutung und Funktion eines Schriftelements unterscheiden kann.

Im MockUp-Entwurf der Homepage befinden sich beispielhafte Bilder für die Website. Die finalen Bilder und Videos werden vor der Veröffentlichung eingebettet. Nach der Navbar wird ein Eindrucks-video vom Verfahren von BambiGuard eingebettet. Dies soll dazu dienen, dem Besucher der Website einen direkten Einstieg in die Thematik der Wildtierabsuche zu geben. Das Video soll sich im Hintergrund platzieren, wodurch die weiße Überschrift und die Wortgruppe mehr ins Auge stechen. Die Inhaltsbereiche werden schlicht gehalten, als Hintergrundfarbe wird Weiß gewählt. Der Inhalt und die Überschrift werden in Schwarz gestaltet, um Lesbarkeit zu vereinfachen. Die Inhaltsbereiche werden durch aussagekräftige Bilder zum nächsten Inhaltbereich unterbrochen, wodurch der Besucher einen besseren Überblick über diese Bereiche bekommt.

Auf der Über uns-Seite gliedert sich der Inhaltsbereich in drei Spalten (Blöcke), die jeweils mit Schlagschatten versehen sind, um harte Kanten und Ecken zu vermeiden. Bei diesen drei Blöcken wurden die Ecken abgerundet. Der obere Teil eines Blockes bildet ein rundes Portrait des Teammitglieds, vor einem grünen Hintergrund. Wird mit dem Mauszeiger über das Portrait gezogen vergrößert sich das Portrait und füllt den gesamten vertikalen Bereich aus. Nach dem Portrait wird der Name und die Zuständigkeit im Projektteam mit dicker Schrift platziert, wobei die Namen im CICD-Grün und die Zuständigkeit in Schwarz gehalten wird. Danach folgen für jedes Projektteam eine in Absätze gegliederte Kurzbeschreibungen ihrer Tätigkeiten, um eine grafische Unterscheidung darzustellen, werden die Kurzbeschreibungen in grau gefärbt.

Die Kontakt-Seite umfasst einen Inhaltsbereich und wird daher nicht wie die Home-Seite durch Bilder getrennt. Ebenso wie auf den Inhaltsbereichen auf der Home-Seite herrschen auch hier die Farben Weiß als Hintergrundfarbe und schwarze Schrift vor. Im unteren Bereich wird eine Google Maps Karte in Buntansicht eingefügt sowie die Adressdaten der Schule und unsere Kontaktmail in grünen Blöcken.

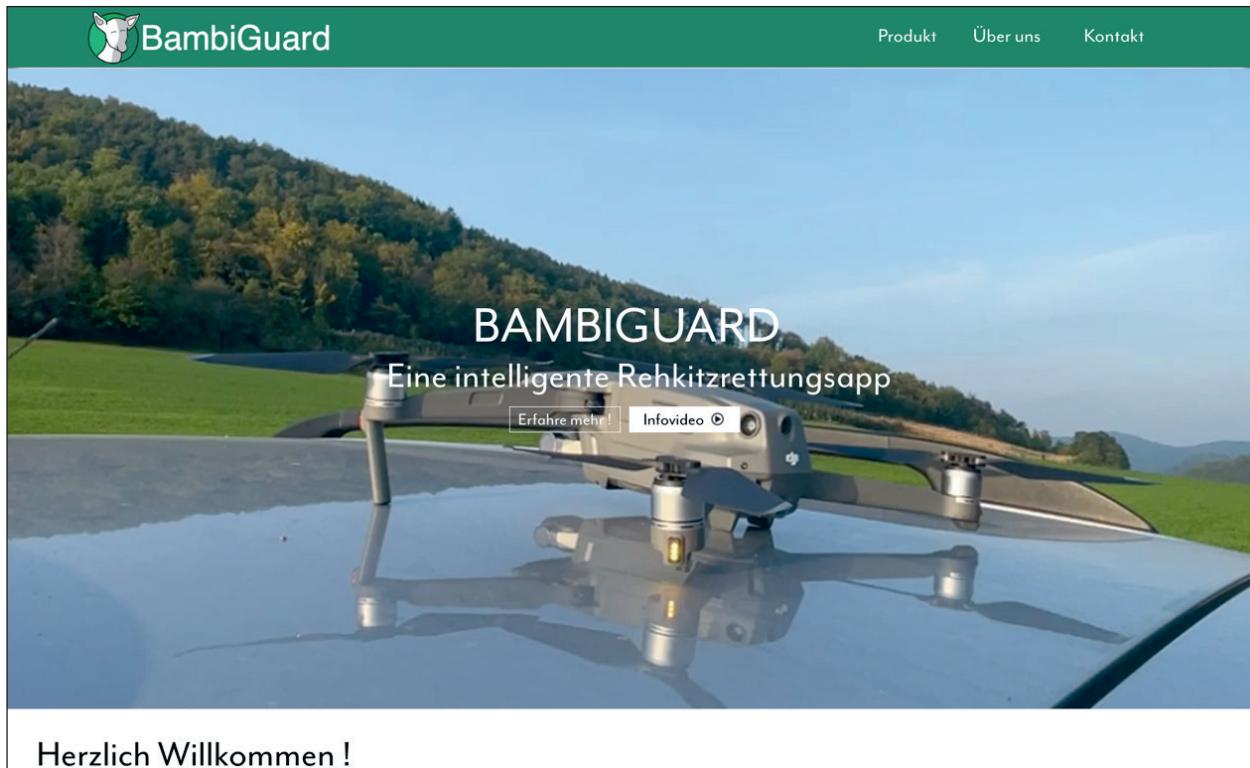


Abbildung 105: Website - MockUp (Home; oberer Teil)

Team Member	Role
Felix Teufel	Corporate Identity/ Design, Social Media Marketing
Maximilian Strobl	Projektleitung, UI/UX Design, Frontend Entwicklung, Webentwicklung
Clemens Losbichler	Bilderkennung, Backend Entwicklung

Abbildung 106: Website - MockUp (Über uns; oberer Teil)

Entwicklungsumgebung Visual Studio Code

Für die Entwicklung der Website wurde die Entwicklungsumgebung Visual Studio Code von Microsoft verwendet. Visual Studio Code ist plattformunabhängig und für Windows, MacOs und Linux kostenlos verfügbar. Der kostenlose Quelltext-Editor ermöglicht unter anderem Syntaxhighlighting, Code-Faltung (Kennzeichnung zusammengehöriger Teile), Debugging, Auto vervollständigung sowie eine Versionsverwaltung. Im Unterschied zu der IDE Visual Studio arbeitet Visual Studio Code nicht mit Projektdateien, sondern auf Basis von Quelltexten und Ordnern. Ebenso bietet Visual Studio Code unzählige Extensions, wie etwa ein XSL Transformation, einen PHP Debugger oder einen XML-Editor. Die Extensions können einfach installiert und in das Projekt eingebunden werden. Für die Website wurde die Extension ES7+ React/Redux/React-Native snippets verwendet.

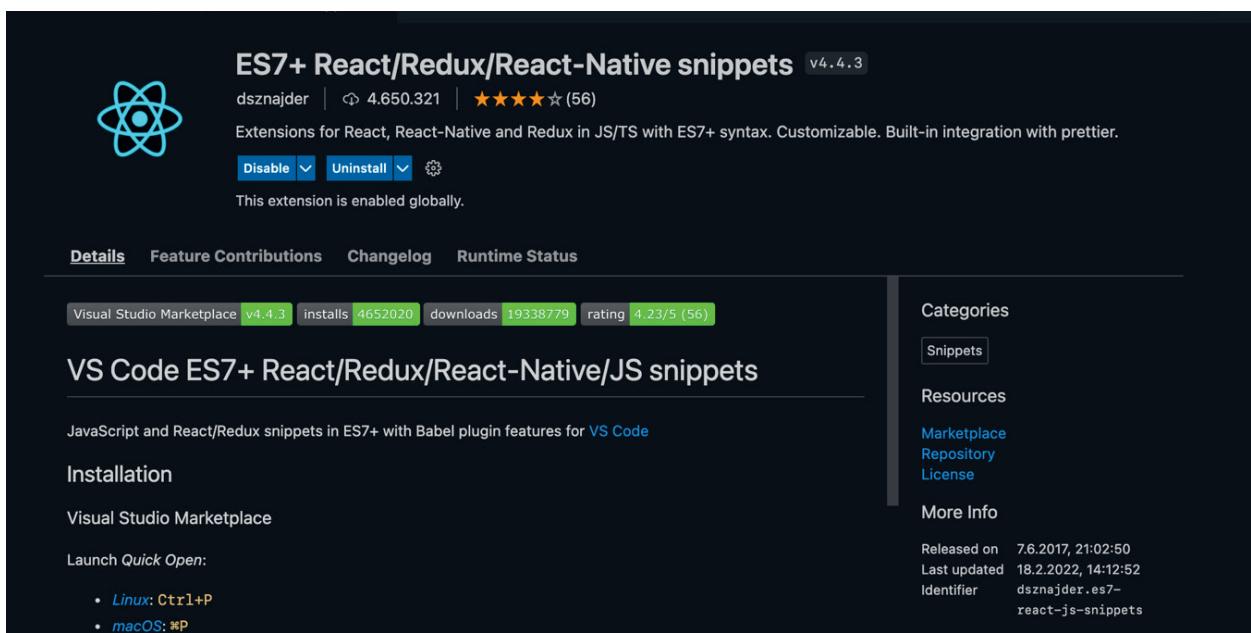


Abbildung 107: Verwendete Extension Visual Studio Code

Erstellen der React-Webapp

Zu Beginn muss für die Entwicklung die JavaScript-Laufzeitumgebung NodeJS²² installiert werden, um im Projekt den Node Package Manager (npm) nutzen und Packages hinzuzufügen zu können. Zur Installation und Erzeugung unserer React-App werden folgende Befehle in den Terminal eingegeben

```
npx create-react-app bambiguard
cd bambiguard
npm start
```

22 <https://nodejs.org/de/>

Mit dem ersten Befehl wird eine React-App erzeugt mit dem Namen „bambiguard“, danach wechselt man in den Projektfolder und startet die JavaScript-Entwicklungsumgebung und erhält eine Web-Ansicht.

Nachfolgend werden die Komponenten und Seiten der BambiGuard-Website näher beschrieben. Es wird hauptsächlich auf die Logik Wert gelegt Die Beschreibung des genauen Design-Aufbaus würde den Rahmen unserer Diplomarbeit sprengen Die Umsetzung des Designs kann dem MockUp oder der Website im Browser entnommen werden.

Komponenten

Die REACT-Website gliedert sich in mehrere Komponenten, die auf die vier Seiten eingebunden werden.

Navbar

Die Navbar-Komponente dient zum Navigieren in der Web-App und baut auf die Routen des React-Routers im JavaScript-File „App“ auf. Dieser wird in folgenden Absätzen genauer beschrieben.

```
const [click, setClick] = useState(false);

const handleClick = () => setClick(!click);
const closeMobileMenu = () => setClick(false);
...

<Link to='/' className='navbar-logo' onClick={closeMobileMenu}>
    <img src='../192x192.png' alt="BambiGuard Logo" width="50"
        height="50"/>
    BambiGuard
</Link>
<div className='menu-icon' onClick={handleClick}>
    <i className={click ? 'fas fa-times' : 'fas fa-bars'} />
</div>
<ul className={click ? 'nav-menu active' : 'nav-menu'}>
    <li className='nav-item'>
        <Link to='/' className='nav-links'
            onClick={closeMobileMenu}>
            Produkt
        </Link>
    </li>
    ...
</ul>
```

Programmausdruck 64: Die Navbar-Komponente in der Datei Navbar.js

Der Link-Tag vermittelt beim Klick auf das Logo oder die Schrift „BambiGuard“ mithilfe des Attributes zur root-Seite (to="/""). Wird auf eine andere Seite referenziert, so steht nach dem Lash der Sei-

tenname. Im Zusammenspiel mit den drei Konstanten, wird entschieden, ob die Navigation in einer Reihe angezeigt wird oder zu einem Hamburger-Menü für eine mobile Ansicht wird. Entscheidend für diese Darstellung ist das className-Attribut im ul-Tag.

HeroSection

In der HeroSection-Komponente wird der erste Eindruck der Website geprägt, diese Komponente enthält das Headervideo, sowie die Haupt- und Nebenunterschrift und die beiden Hauptbuttons.

```
const [buttonPopup, setButtonPopop] = useState(false);
...
<div className='hero-container'>
    <video src='/videos/Headervideo.mp4' autoPlay loop muted />
    <h1>BAMBIGUARD</h1>
    <p>Eine intelligente Rehkitzrettungsapp</p>
    <div className='hero-btns'>

        <Button
            className='btns'
            buttonStyle='btn--outline'
            buttonSize='btn--large'
        >
            Erfahre mehr!
        </Button>

        <Button
            className='btns'
            buttonStyle='btn--primary'
            buttonSize='btn--large'
            onClick={() => setButtonPopop(true)}
        >
            Infovideo
        </Button>
        <Popup trigger={buttonPopup} setTrigger={setButtonPopop}>
            <h3>Infovideo in Arbeit</h3><br><br>
            <p>Das Werbevideo für ...</p>
        </Popup>
    </div>
</div>
```

Programmausdruck 65: Die HeroSection-Komponente in der Datei HeroSection.js

Der HeaderSection-Quelltext gliedert sich in zwei Hauptbereiche, der Header-Container und die Headerbuttons. Im Headercontainer befindet sich das Headervideo, welches immer beim Öffnen der Website den kompletten Screen füllt, sowie die Haupt- und Nebenüberschrift. Im Headerbutton-Bereich, der sich im Containerbereich befindet, sind die beiden Buttons sowie der Trigger für das PopUp. Wird auf den Button „Erfahre mehr“ gedrückt, wird der Besucher zur ersten Überschrift geleitet (genaue Abarbeitung in der Komponente Button). Wird auf den zweiten Button „Infovideo“ gedrückt,

sollte sich das Werbevideo in diesem PopUp ändert. Da dieses jedoch nicht fertiggestellt wird, wurde hier nur Text integriert.

Button

Die Button-Komponente beschäftigt sich mit der konstanten Zuweisung für die Button-Komponente, sowie der Weiterleitung und Referenzierung des ersten Buttons in der HeroSection.

```
export const Button = ({  
  children,  
  type,  
  onClick,  
  buttonStyle,  
  buttonSize  
) => {  
  const checkButtonStyle = STYLES.includes(buttonStyle)  
    ? buttonStyle  
    : STYLES[0];  
  
  const checkButtonSize = SIZES.includes(buttonSize) ?  
    buttonSize : SIZES[0];  
  
  return (  
    <HashLink to='/#maintop' className='btn-mobile'>  
      <button  
        className={`btn ${checkButtonStyle} ${checkButtonSize}`}  
        onClick={onClick}  
        type={type}  
      >  
        {children}  
      </button>  
    </HashLink>  
  );  
};
```

Programmausdruck 66: Die Button-Komponente in der Datei Button.js

Cards

Die Cards-Komponente besteht aus mehreren CardItems, die für den direkten Inhalt der CardItems bestimmt sind. In Programmausdruck 67 werden die Properties für die Komponente CardItem genau befüllt.

```
<CardItem
  src='images/Team/Teufel.png'
  leadtext='Corporate Identity/ Design, Social Media Marketing'
  text1='Felix befasst ...'
  text2='Druckprodukte wie etwa Flyer ...'
  text3='Die Erstellung von Werbeartikeln ...'
  name='Felix Teufel'
  path='/'>
```

Programmausdruck 67: Eine definierte CardItem-Komponente in der Datei Cards.js

CardItem

Da in der Komponente Cards bereits die Attribute für die CardItems gesetzt werden, kümmert sich die Komponente CardItem lediglich um den Aufbau eines CardItems.

```
<li className='cards__item'>
  <Link className='cards__item__link' to={props.path}>
    <figure className='cards__item__pic-wrap'
      data-category={props.label}>
      <img
        className='cards__item__img'
        alt='Teambild'
        src={props.src}
      />
    </figure>
    <div className='cards__item__info'>
      <h4 className='cards__item__name'>{props.name}</h4>
      <h5 className='cards__item__leadtext'>
        {props.leadtext}
      </h5>
      <p className='cards__item__text'>{props.text1}</p>
      <p className='cards__item__text'>{props.text2}</p>
      <p className='cards__item__text'>{props.text3}</p>
    </div>
  </Link>
</li>
```

Programmausdruck 68: Die Card-Item-Komponente in der Datei CardItem.js

PopUp

Das PopUp-Fenster ist ursprünglich als Anzeigefeld für das Werbevideo entstanden, durch die Nicht-Fertigstellung wurde eine Textnachricht als Ersatz konzipiert. Der Text wird direkt aus der HeaderSection für das PopUp verwendet.

```

function Popup(props) {
  return (props.trigger) ? (
    <div className='popup'>
      <div className='popup-inner'>
        <button className='fas fa-window-close'
          className='close-button' onClick={() =>
            props.setTrigger(false)}>X</button>
        {props.children}
      </div>
    </div>
  ) : "";
}

```

Programmausdruck 69: Die PopUp-Komponente in der Datei Popup.js

ImpressText

In der ImpressText-Komponente befindet sich der Text für die Offenlegungspflicht der Website mit simplen HTML-Tags. Diese Komponente wird in die Impress-Seite eingebettet.

```

<div class="impressText">
  <h1>Impressum</h1>
  <p>Informationspflicht laut §...</p><br/>
  <p>Verantwortlich für die Inhalte</p><br/>
  <p><strong>BambiGuard</strong> - Team</p>
  <p>Projektleiter</p><br/>
  <p>Maximilian Strobl</p>
  ...
</div>

```

Programmausdruck 70: Die ImpressText-Komponente in der Datei ImpressText.js

Seiten der Website

Die Seitenweiterleitung im Rahmen der BambiGuard-Website wird mithilfe von Routing umgesetzt. Der React Router vermittelt anhand der URL zu den unterschiedlichen Komponenten. Stimmt eine Route mit der URL genau überein, liefert der Router diese Route zurück. Mithilfe der Switch-Komponente und der exact-Property liefert der Router immer nur genau eine Route zum Rendering zurück und nicht mehrere. In Programmausdruck 71 ist der Routing-Ablauf der BambiGuard-Website zu sehen.

```

    return (
      <>
      <Router>
        <Navbar />
        <Switch>
          <Route path='/' exact component={Home} />
          <Route path='/AboutUs' component={AboutUs} />
          <Route path='/Contact' component={Contact} />
          <Route path='/Impress' component={Impress} />
        </Switch>
      </Router>
    ) ;
  );

```

Programmausdruck 71: Routing-Ablauf in der Datei App.js

Wird eine Seite neu geladen oder auf eine andere Seite weitergeleitet, wird ein Ladeicon (in unserem Fall eine Animation) für eine gewisse Zeit (Millisekunden) vor dem Inhalt angezeigt. Umgesetzt wird diese Animation mit Hooks (State Hooks, Effect Hooks). Hooks sind Erweiterungen von React und bieten die Möglichkeit, funktionale Komponenten zu nutzen. Vor 2018 war dies nur Klassenkomponenten vorbehalten. Mit dem State Hook ist es möglich einen Array zurückzuliefern, der mittels einer Array Destrukturierung in zwei Konstanten aufgeteilt wird. Der erste Wert repräsentiert den aktuellen Status, der zweite Wert bildet die dazugehörige Funktion. Die useEffect-Methode überwacht den gesamten Lifecycle-Prozess und wird erst aktiv, wenn die Komponente geupdated wird. Nachfolgender Programmcode stellt die Ladeanimation dar. Ändert sich der Status, so ändert sich die Konstante „loading“ und die Funktion „setLoading“ wird ausgeführt. Dadurch wird die useEffect-Methode aktiv und wartet die 2,5 Millisekunden ab und zeigt durch den return der If-Verzweigung die Animation an. Wird „loading“ während des Prozesses nicht mehr geändert, so wird der Inhalt in der else-Verzweigung dargestellt und zurückgeliefert. Dieser Ladeicon-Prozess wird auf allen Seiten der BambiGuard-Website angewandt.

```

const [loading, setLoading] = useState(true);
useEffect(() => {
  const loadData = async () => {
    await new Promise((r) => setTimeout(r, 2500));
    setLoading((loading) => !loading);
  };
  loadData();
}, [])

if (loading) {
  return 
} else {
  return ()
}

```

Programmausdruck 72: Das Ladeicon in den Seiten der BambiGuard-Website; in den Dateien des pages-Ordner

Home

Die Home-Seite dient als Landing Page und wird daher mit dem Hauptinhalt gefüllt. In der Home-Seite befinden sich die Komponenten HeroSection (zuständig für die Einbettung des Header-Videos sowie die Einbettung des Werbevideos und der Hauptüberschrift), die Inhaltsbereiche und den Footer.

Über uns

Die Über uns-Seite beinhaltet zwei Komponenten: die Cards- und die Footer-Komponente.

Kontakt

Die Kontakt-Seite beinhaltet zwei Komponenten die Form-Komponente (beinhaltet das Formular und die Kartenansicht) und die Footer-Komponente.

Impressum

Die Impressum-Seite beinhaltet zwei Komponenten: die ImpressText-Komponente (beinhaltet das gesamten Inhalt der Offenlegungspflicht) und die Footer-Komponente.

3.8.3 Werbevideo

Um die Möglichkeit zu besitzen, bei Veranstaltungen oder auf Webseiten die Diplomarbeit einfach und effizient zu bewerben, wird ein Video angefertigt, welches alle wichtigen Informationen bereitstellt.

Zum Zeitpunkt der Realisierung sorgten schlechte Wetterbedingungen und zu niedriges Gras dafür, dass kein Video erstellt werden konnte, welches das Funktionsprinzip BambiGuard realistisch darstellt. Es wurden Aufnahmen erstellt, jedoch können diese aufgrund der kalten Temperaturen, welche im Winter vorherrschten, nicht wie geplant verwendet werden.



Abbildung 108: Werbevideo Aufnahme bei niedrigen Temperaturen

3.8.4 Infopaket

Das Infopaket beinhaltet Druck-Elemente wie Flyer und Visitenkarten, welche bei Veranstaltungen verteilt werden können, um viele Menschen zielgruppengerecht zu erreichen. Eine mögliche Veranstaltung, bei welcher diese Artikel verteilt werden können, ist etwa eine Landwirtschaftsmesse.

Visitenkarten

Als Hersteller der Visitenkarten wurde Druck.at gewählt. Dieses Unternehmen bietet eine Vielzahl von Druckprodukten an, welche in Österreich produziert werden. Für die Auswahl wichtige Faktoren waren etwa die individuelle Gestaltungsmöglichkeit oder das Format und die Papierart. Sie wurden mit bereits vorhandenen Merkmalen der Diplomarbeit kombiniert.

Es wurde ein Modell mit individuellem Farbkern gewählt. Das bedeutet, dass die Visitenkarte aus 3 Schichten besteht: der bedruckten Vorder- und Rückseite sowie einem grünen Karton in der Mitte. Diese Eigenschaft erhöht die Dicke und Wirkung der Karte. Die Grammatur für dieses Material beträgt 755 g/m^2 und ist ein vergleichsweise hoher Wert, Briefpapier besitzt im Vergleich dazu eine Grammatur von rund 80 bis 100 g/m^2 . Je höher die Grammatur eines Papiers ist, desto schwerer und steifer ist es. Eine hohe Grammatur ist bei Visitenkarten gewünscht, da dadurch die Entstehung von Falten oder Knicken verringert wird.

Die Abmessungen der Karte sind $85 \times 55 \text{ mm}$ sind Standardwerte für Visitenkarten. Sie wurden gewählt, um zu garantieren, dass die Karten in Brieftaschen und Ordnern, welche für Visitenkarten ausgelegt sind, passen, da diese meist von diesem Format ausgehen. Die Karte ist beidseitig bedruckt, wobei auf der Vorderseite Logo und Schriftzug von BambiGuard zu sehen sind. Auf der Rückseite finden sich die individuellen Kontaktdaten der Teammitglieder. Aufgrund dieser Eigenschaften wurde ein $4c/4c$ -farbig beidseitiger CMYK Aufdruck gewählt. Dieser legt fest, dass auf beiden Seiten ein farbiger Druck möglich ist, welcher in diesem Fall auch benötigt wird, da das Logo Farben beinhaltet.

Der Preis der Visitenkarten mit Farbkern, dem Format $85 \times 55 \text{ mm}$ und $4c/4c$ CMYK Bedruck belaufen sich bei drei Sujets und einer Anzahl von 100 Stück pro Sujet auf $210,57 \text{ €}$ inkl. USt. Drei Sujets bedeutet, dass drei verschiedene Motive bzw. die Karte für drei verschiedene Personen individuell gedruckt werden kann.

Erstellung der Visitenkarten

Die Visitenkarten wurden in Adobe Illustrator erstellt. Es wurde für jedes Teammitglied eine eigene Karte mit Namen, Telefonnummer und E-Mail-Adresse ausgearbeitet.

Der Erste Schritt bei der Erstellung der Visitenkarte ist das Betrachten des Datenblatts, welches von der Druckerei bereitgestellt wird. Darin finden sich Merkmale wie Sicherheitsabstand oder Druckformat wieder. Diese sind wichtig, da beim Design darauf geachtet werden muss, dass im späteren Produktionsprozess keine wichtigen Grafikelemente abgeschnitten werden. Da es sich hier um unbeschichtetes Papier handelt, muss in der Erstellungssoftware auch ein dementsprechendes Farbprofil gewählt werden – in diesem Fall wird Uncoated FOGRA 29 (ISO 12647-2:2004) verwendet.



Abbildung 109: Visitenkarte Randdefinitionen

Die Vorderseite soll wie bereits erwähnt eine Kombination aus Logo und Text bilden. Dazu wurde in der Visitenkarte das Logo mittig links platziert, um noch genug Platz für den Text zu schaffen. Für diesen wurden zwei verschiedene Schriftarten gewählt. Wie bereits bei der Auswahl der Schriftarten erwähnt, wurden die Schriftarten Mr. Eaves Mod OT Regular und Recherche Regular verwendet. Für die Vorderseite wurden keine weiteren Elemente verwendet, da diese einfach wirken und schnell erkennbar sein soll.



Abbildung 110: Visitenkarte Vorderseite

Die Rückseite beinhaltet Informationen zu den jeweiligen Mitgliedern, wie etwa Name, E-Mail und Telefonnummer. Zusätzlich wird auf jeder Karte die E-Mail von BambiGuard angeführt. Hier werden ebenfalls wieder beide Schriftarten verwendet, jedoch wird die Schriftart Recherche Regular nur für den Namen der Mitglieder genommen, da Informationen wie die Telefonnummer einfach zu lesen sein sollen. Zusätzlich wird zur besseren Stilisierung eine Illustration eines Rehs hinzugefügt.



Abbildung 111: Visitenkarte Vorderseite (Felix Teufel)

Microsoft Word und PowerPoint Vorlagen

Um bei Präsentationen oder Dokumenten immer dieselbe Formatierung zu erreichen, wurden Vorlagen für Microsoft Word und PowerPoint erstellt.

In PowerPoint können Vorlagen unter dem Punkt Folienmaster erstellt werden, dieser ist im Ansicht Menü zu finden. Es können Schriftarten, Farben, Effekte, Layout, Hintergrundbild und Farbe geändert und festgelegt werden. Die Formatvorlage enthält die typischen Farben BambiGuards und verwendet die Schriftart Mr Eaves Mod OT für alle Textelemente. Zusätzlich wurde das Logo eingefügt. Auf einen Hintergrund wird in diesem Fall verzichtet, um Probleme der Lesbarkeit bei unterschiedlichen Schriftfarben sowie anschließend bei den Präsentationen zu vermeiden. Projektoren weisen oft einen zu geringen Kontrast auf, wodurch der gesamte Bildinhalt verschmelzen und schwer erkennbar sein würde. Die erstellte Vorlage muss anschließend als PowerPoint-Vorlage (*.potx) gespeichert werden.

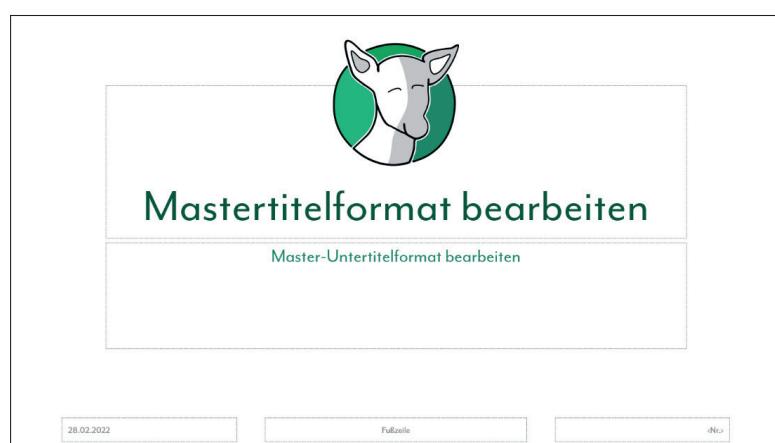


Abbildung 112: Microsoft PowerPoint Vorlage Titelfolie

In Microsoft Word kann unter dem Punkt Formatvorlagen eine neue Vorlage erstellt werden. Für die Vorlage BambiGuards werden mehrere Überschriftenformate, sowie Textformate erstellt, welche ebenso wie die PowerPoint Vorlage die Schriftart Mr Eaves Mod OT verwendet. Für die Überschriften

kommen die bekannten Farben zum Einsatz. In der Kopfzeile werden Seitenzahl und Veröffentlichungsdatum angezeigt. An der Unterseite wurde das Logo befestigt. Um diese Datei wiederum als Vorlage zu speichern, muss im Menü das Format Word-Vorlage (*.dotx) gewählt werden.

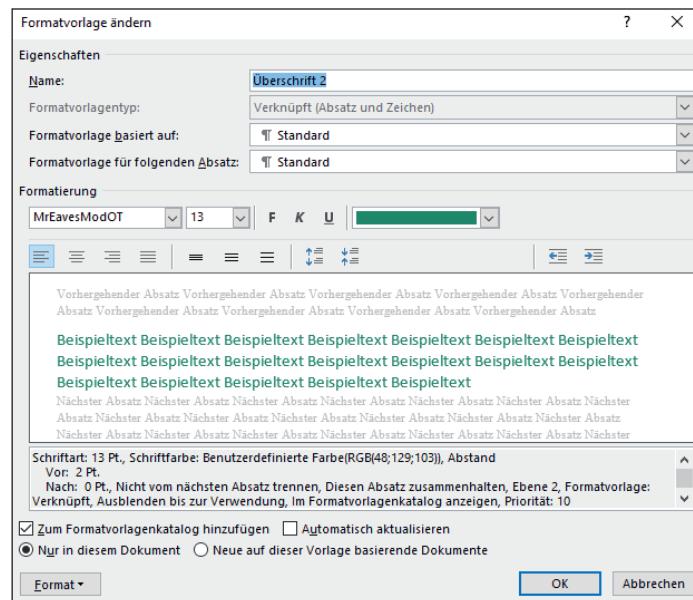


Abbildung 113: Microsoft Word Formatvorlage ändern

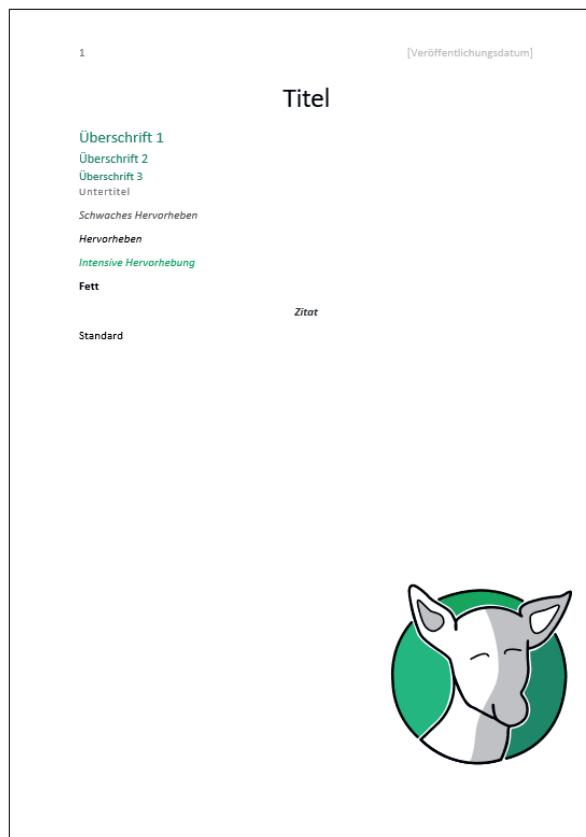


Abbildung 114: Microsoft Word Vorlage

Flyer

Der Flyer soll wichtige Informationen zu BambiGuard beinhalten und über die Funktionsweise der App und Methoden zur Rehkitzrettung informieren.

Als Hersteller des Flyers wird erneut Druck.at gewählt und wird auf die individuellen Gestaltungsmöglichkeiten sowie das Format und die Papierart geachtet.

Es wird ein Folder gewählt, welcher einen Z-Falz mit 6 Seiten vorweist. Das bedeutet, dass der Bogen an zwei Stellen zu einer Z-Form gefaltet wird. Die Abmessungen des Folders betragen im ausgefalteten Zustand 445,5x210 mm. Es wird ein 300 g/m² Volumenkarton ohne Cellophanierung gewählt, welcher einen 4c/4c Digitaldruck vorweist, also einen beidseitigen Farbaufdruck. Gegen Aufpreis ist es möglich das Herkunftssiegel „Printed in Austria“ aufdrucken zu lassen. Zusätzlich können Optionen wie die Verwendung von FSC zertifiziertem Papier (FSC Mix 70%-HFA-COC-100222) angedacht werden. Das bedeutet, dass das Papier zu mindestens 70% aus Holz von FSC stammenden Wäldern oder recyceltem Papier besteht. Auch Optionen, wie das Einschleifen der Flyer zu bestimmten Stückzahlen ist möglich und es können Flyerständer für Messen bestellt werden.



Abbildung 115: Zertifizierungssiegel

Der Preis des Flyers zu einer Stückzahl von 100 und ohne zusätzlichen Siegeln oder Zertifizierungen beläuft sich auf 146€ inkl. USt.

Erstellung des Flyers

Der Flyer wird in Adobe InDesign erstellt, für den Inhalt werden lizenzzfreie und eigens erstellte Bilder verwendet. Da bei dem ausgewählten Flyer keine Beschichtung vorgenommen wird, muss wiederum ein passendes Farbprofil gewählt werden. In diesem Fall wird das Farbprofil Uncoated FOGRA29 (ISO 12647-2:2004) angewandt. Da die Abmessungen des Folders im ausgeklappten Zustand 445,5x210 mm betragen, muss die Breite durch drei geteilt werden, um auf die Breite einer einzelnen Seite zu kommen. Anschließend kann das Projekt erstellt werden. Hier werden 6 Seiten mit der Größe A5 (zu finden unter Vorlagen-->Druck) erstellt. Zusätzlich kann noch ein Seitenrand festgelegt werden.



Seite 1



Seite 2



Seite 3



Seite 4



Seite 5



Seite 6

Abbildung 116: Flyer Seitenansichten

Flyer Seite 1

Die Erste Seite soll, ähnlich zur Visitenkarte, das Logo gemeinsam mit dem BambiGuard Schriftzug zeigen. Da dies allein jedoch dafür sorgen würde, dass die Seite unvollkommen und leer wirkt, wurden noch einige farbliche Akzente in den für BambiGuard typischen Grüntönen gesetzt. Diese sollen sich im gesamten Flyer wieder finden, um somit eine bessere Trennung zwischen Elementen zu schaffen. Die eingefügten Bilder und Dateien können unter dem Reiter Verknüpfungen gefunden und aktualisiert werden, falls sie im Dateiverzeichnis verschoben oder ersetzt wurden. Der Schriftzug wurde in unterschiedlichen Schriftarten gehalten, um die bereits bei der Visitenkarte erwähnten Eigenschaften zu vermitteln.

Flyer Seite 2

Die zweite Seite soll grundlegende Fragen wie „Worum geht es bei BambiGuard?“, „Wie funktioniert die BambiGuard App?“ und „Warum müssen Rehkitze geschützt werden?“ beantworten. Die Infotexte werden von Bildern begleitet, welche einen Bezug zum enthaltenen Textinhalt schaffen sollen.

Flyer Seite 3

Die dritte Seite befasst sich mit der genauen Zielsetzung BambiGuards. Es soll darauf eingegangen werden, warum uns der Schutz von Wildtieren am Herzen liegt und worin unser Konzept besteht.

Flyer Seite 4

Die vierte Seite stellt die Teammitglieder der Diplomarbeit vor. Es soll gezeigt werden, welche Tätigkeiten von welchem Teammitglied umgesetzt werden. Die Bilder sollen von einem farbigen Akzent umrandet werden. Die Bilder werden in Adobe Photoshop bearbeitet und werden rund ausgeschnitten, um der Seite mehr Dynamik zu verleihen.

Flyer Seite 5

Die fünfte Seite soll Informationen über die Funktionsweise der App liefern. Es soll genauer darauf eingegangen werden, wie die Koppelung zwischen App, Smartphone und Drohne abläuft. Zudem sollen die verschiedenen Ansichten erklärt werden, da zwischen der Piloten- und Helfer-Ansicht starke Unterschiede bestehen.

Flyer Seite 6

Die sechste Seite soll wichtige Informationen für den Piloten auflisten, da viele rechtliche Faktoren beim Steuern einer Drohne zu berücksichtigen sind und bei Missachtung zu rechtlichen Konsequenzen, Personen oder Sachschäden führen können. Zusätzlich werden wichtige Informationen vor dem Kauf einer Drohne angeführt.

Flyer Gesamtansicht

In Photoshop wurde ein Prototyp erstellt, welcher den fertigen Flyer in Echt darstellen soll, dazu wurde ein gefaltetes Stück Papier mit den Proportionen einer A5 Seite fotografiert und anschließend die einzelnen Seiten des Flyers nachträglich eingefügt. Der erstellte Flyer würde ausgedruckt wie folgt aussehen.



Abbildung 117: Flyer Gesamtansicht

3.8.5 Social Media

Die Social-Media-Kanäle werden auf Instagram und Facebook erstellt und sollen zwei Zwecke erfüllen:

- Information der an der Diplomarbeit interessierten Personen
- Bewerbung der Diplomarbeit um mehr Menschen über die Lösung, welche BambiGuard bereitstellt, zu informieren

Die beiden Kanäle beinhalten dieselben Beiträge, da keine Differenzierung zwischen ihnen hinsichtlich des Contents stattfindet. Sinn und Zweck der Kanäle ist es, verschiedene Personengruppen zu erreichen, da auf Facebook im Vergleich zu Instagram eine eher ältere Altersgruppe vorherrscht.

Die Beiträge wurden so konzipiert, dass sie optisch miteinander verbunden sind und somit ein Gesamtbild erzeugen. Sie wurden in Adobe Photoshop und Adobe Illustrator erstellt und beinhalten lizenzenfreie Bilder, welche zu den enthaltenen Texten einen Bezug schaffen sollen.

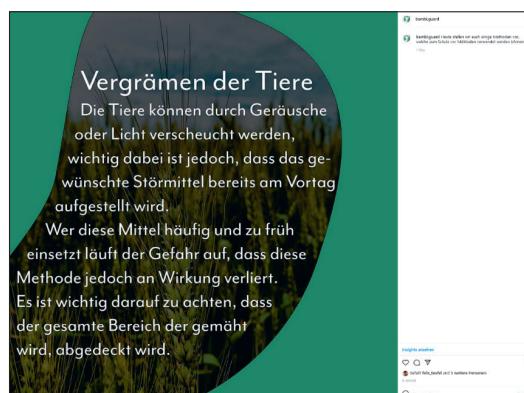


Abbildung 118: Ein Post aus dem Instagram Account

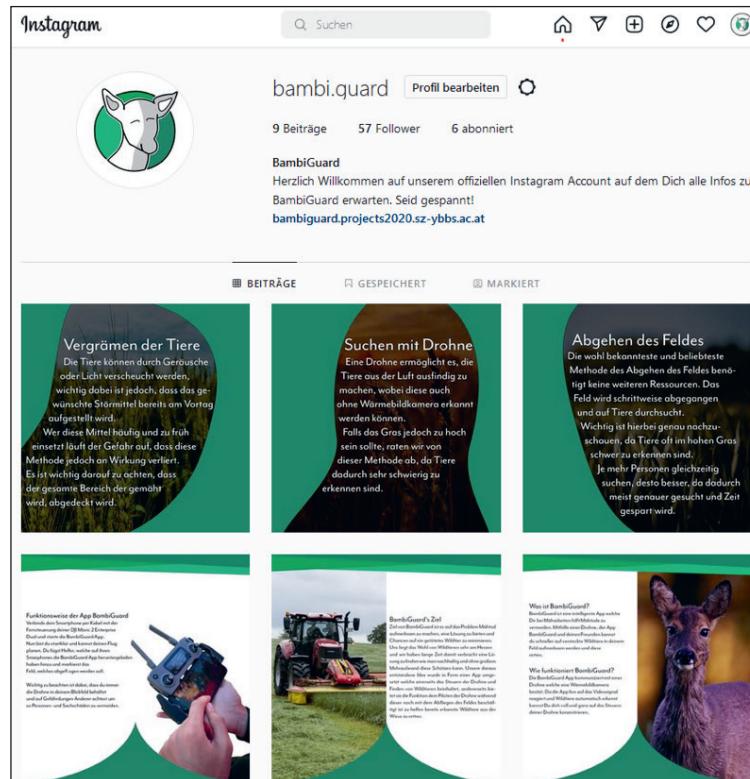


Abbildung 119: Instagram Account von BambiGuard

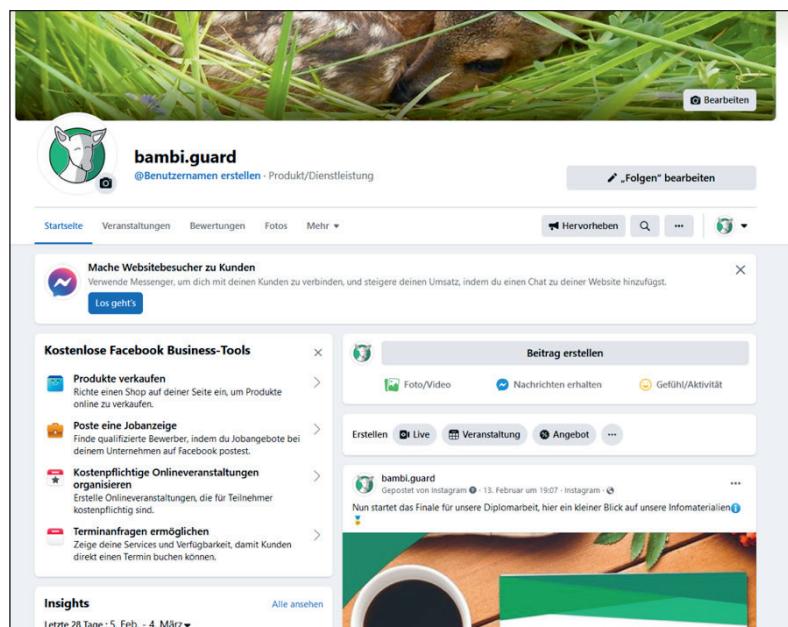


Abbildung 120: Facebook Seite von BambiGuard

4. Evaluierung

4.1 Projektmanagement

Das Projektmanagement des Projekts war von der Größe und Ausführlichkeit adäquat durchgeführt. Formale Fristen und Anforderungen am Projektstart sowie während der Diplomarbeit wurden eingehalten. Außerdem war die Genauigkeit des Controllings gut gewählt, da genug Informationen erfasst wurden, jedoch in keinem unpassenden großen Ausmaß.

Ein Hindernis im Projektmanagement war die Suche einer geeigneten Software zur Auflistung, Verwaltung und Darstellung der Projektplanung. Obwohl es einige Werkzeuge und Programme gibt, sind die meisten von ihnen kostenpflichtig oder nicht für die beiden Betriebssysteme Windows und macOS verfügbar. Letztendlich war die Wahl der Software Quire.io gut und die längere Recherche stellte sich im späteren Verlauf des Projekts als lohnenswert heraus.

4.2 Planung und Konzeption

Die grobe Planung und anfängliche Konzeption des Projekts legte ein gutes Fundament für die Umsetzung der Aufgabenstellung. Die Recherche im Zuge des Abschnitts „Grundlagen und Methoden“ erfolgte ausführlich und die Beschäftigung mit Fachliteratur schaffte benötigtes Basiswissen.

Untrüglich ist die Gesamtheit der Aufgabenstellung zu umfangreich gewählt worden, um im Zuge der Diplomarbeit komplett erfüllt zu werden. Es haben einige geplante Ergebnisse, wie die Fertigstellung der iOS App und die ausführliche Testung des Systems in echten Bedingungen gestrichen werden müssen. Der Grund für diese Kompromisse liegt in der anfänglichen Neuheit der Themengebiete für das Projektteam. Obwohl die Lösungsansätze unter Beachtung der Vorkenntnisse gewählt wurden, waren große Teile der Umsetzung wie die Bilderkennung und die Flugplanung für die Projektmitglieder erstmalig. Wie in der Ergebnisdokumentation beschrieben, führten unerwartete Erweiterungen der Umsetzung (z.B. Coverage Path Planning und Android NDK) zu einem deutlich höheren Arbeitsaufwand. Nachträglich ist zu erkennen, dass die Vermeidung dieser Probleme beispielsweise durch ausführlichere Prototypen gewährleistet würde. Bei der Erstellung der Benutzeroberfläche hätte außerdem durch genaueres Analysieren der spezifischen Programmierung (Android, iOS) Zeit und Aufwand gespart werden können.

Nichtsdestotrotz wurden die Kernpunkte der Aufgabenstellung, also die Bilderkennung und die App, als einzelne Komponenten erreicht. Die Auseinandersetzung mit den Themengebieten und die detaillierte Ausarbeitung der Problemlösungen sorgte für eine Erweiterung des theoretischen und prakti-

tischen Wissens aller Projektmitglieder. Des Weiteren ist das Durchführen eines Projektes in diesem Ausmaß mit wertvollen Erfahrungen verknüpft, welche in späteren beruflichen Laufbahnen der Teammitglieder positive Auswirkungen zeigen werden.

4.3 Verbesserung des Systems BambiGuard

Die gewählten Methoden und Werkzeuge, vor allem die Implementierung der Bilderkennung in C++ wurden so gewählt, dass sie die besten Ergebnisse erzielen. Durch die Datenübertragung von Pilot und den Helfern über das Internet und einen Server ist der Einsatz von BambiGuard skalierbar und die App kann mit leichten Anpassungen (z. B. von Auswahl der Helfer) für große Missionen optimiert werden.

Die Realisierung dieser Bereiche hätte beispielsweise durch den Einsatz von künstlicher Intelligenz anstelle der klassischen Bilderkennung beschleunigt werden können. Außerdem wäre eine Aufteilung der Back-end-Programmierung (Nodejs Server, Flugplanung und Bilderkennung) auf mehrere Entwickler für die Programmierung von Vorteil gewesen.

Die Entwicklung der Android App wurde in Bezug auf das Design, das Layout und die Grundfunktionen mit nur kleinen Hindernissen (z. B. bei der Einbindung des DJI SDK) gestört. Das lag unter anderem daran, dass diese Kenntnisse einerseits im Unterricht besprochen wurden und andererseits eine Affinität der bearbeitenden Projektmitglieder zu dem Themengebiet besteht.

4.4 Repräsentation

In den Themengebieten der Repräsentation sind mit Ausnahme des Werbevideos alle Bereiche wie erwartet erfüllt worden. Während der Planung der einzelnen Arbeitspakete wurde nicht berücksichtigt, dass ein Dreh des Werbevideos im Winter nicht möglich ist. Aufgrund der oft noch sehr kalten Temperaturen konnte dies nicht erwartungsgemäß erstellt werden. Retrospektiv betrachtet hätte der Drehermin für ein optimales und realistisches Video bereits im Sommer stattfinden müssen.

Bei der Erstellung des Logos wurde das Programm Affinity Designer verwendet, welches im Unterricht nicht behandelt wurde. Trotz geringer Erfahrung mit dieser Umgebung wurden ein Logo und einige Infografiken erstellt, welche durchaus gut gelungen sind und den Erwartungen der Teammitglieder entsprechen. Erfahrungsgemäß hätte die Erstellung dieser Grafiken in einem Programm wie etwa Adobe Illustrator länger gedauert, da die Bedienung von Affinity Designer von einem Tablet aus stattfinden kann und sich somit simpler und einfacher gestaltet.

Aufgrund der im Unterricht angeeigneten Kenntnisse in den Programmen der Adobe Creative Cloud ist die Arbeit mit den Programmen Photoshop, Illustrator, InDesign und Lightroom gut abgelaufen. Bei Fragen konnte problemlos nach Rat bei den unterrichtenden Lehrkräften gefragt werden.

Ein unerwartetes Hindernis war, dass innerhalb der Repräsentation oft mehr Zeit für das Planen der Elemente aufgebracht werden musste als für die Erstellung der Elemente selbst. So bestand etwa ein Großteil der Erstellung der Social-Media Beiträge aus der Planung dieser. Folgende Fragen wurden meist vor der grafischen Umsetzung gestellt:

- Welches Thema wird behandelt?
- Wie sprechen wir die Betrachter am Besten an?
- Welche Bilder oder Grafiken sollen erstellt oder verwendet werden?
- Wie werden die neuen Beiträge mit den Alten verbunden?

Diese Fragen konnten oft nicht so einfach beantwortet werden. Das Finden von passenden Bildern gestaltete sich als schwierig, da darauf geachtet werden musste, dass diese frei nutzbar sind und den gegebenen Kontext des Inhalts wiederspiegeln.

Ein tatsächlicher Ausdruck der erstellten Infomaterialien von einer externen Druckerei wie etwa dem Flyer oder der Visitenkarten war aufgrund der zu hohen Auflagenzahl und den damit verbundenen Kosten leider nicht möglich beziehungsweise sinnvoll. Der Druck von nur 10 Flyern hätte einen Preis von 8-10€ pro Flyer bedingt und bei Visitenkarten hätten pro Person mindestens 100 Stück bestellt werden müssen.

Im Bezug auf die Website wurde der Aufwand zur Erstellung unterschätzt. Trotz anfänglicher Probleme mit der Einrichtung des Frameworks konnte die Website jedoch mit geringer Verzögerung fertiggestellt werden. Die Daten zur Befüllung wurden bis auf das geplante Werbevideo zeitgerecht fertiggestellt und in die Website übertragen.

5. Abbildungsverzeichnis

Abbildung 1: Schwellenwert 128 bei einer 7-Bit Skala	18
Abbildung 2: Strukturelemente	19
Abbildung 3: Erosion (Strukturelement 3x3 Box)	20
Abbildung 4: Dilation (Strukturelement 3x3 Box).	20
Abbildung 5: Morphologisches Öffnen	21
Abbildung 6: Morphologisches Schließen	21
Abbildung 7: Isolation von Objekten durch eine Kombination der morphologischen Transformationen	21
Abbildung 8: Konturen eines Binärbildes	22
Abbildung 9: Einschichtiges neuronales Netzwerk.	23
Abbildung 10: Convolution eines CNN (Die Werte der Eingabe, sowie des Kernels in diesem Beispiel sind beliebig gewählt)	24
Abbildung 11: Abbildung 11: UI/UX Honigwabe nach Morville	26
Abbildung 12: UI/UX Prinzipbeispiel iPhone Einhandmodus.	28
Abbildung 13: UI/UX Prinzipbeispiel Gegenüberstellung alte/neue TV Bedienungen.	29
Abbildung 14: UI/UX Prinzipbeispiel MailChimp	30
Abbildung 15: UI/UX Prinzipbeispiel Prototyp	31
Abbildung 16: Gesetz der Nähe	36
Abbildung 17: Gesetz der einfachen Gestalt.	36
Abbildung 18: Gesetz der Ähnlichkeit.	37
Abbildung 19: Gesetz der Geschlossenheit	37
Abbildung 20: Gesetz der Erfahrung	38
Abbildung 21: Gesetz der Kosntanz	38
Abbildung 22: Gesetz der Fortsetzung.	38
Abbildung 23: Adobe InDesign Benutzeroberfläche	39
Abbildung 24: Adobe Illustrator Benutzeroberfläche	41
Abbildung 25: Adobe Photoshop Benutzeroberfläche	42
Abbildung 26: Affinity Designer Benutzeroberfläche	44
Abbildung 27: Komponentendiagramm.	51
Abbildung 28: Kommunikation zwischen Pilot, Server und Helfer	53
Abbildung 29: Landing Screen (Modell)	55

Abbildung 30: Flugplanung (Modell)	55
Abbildung 31: Pilot – Flugende (Modell)	55
Abbildung 32: Pilot – im Flug (Modell)	56
Abbildung 33: Pilot – Einzelrettung (Modell)	56
Abbildung 34: Helfer – Flug beitreten (Modell)	56
Abbildung 35: Helfer – im Flug (Modell)	56
Abbildung 36: Landing Screen (Wireframe)	59
Abbildung 37: Pilot – Flugplanung (Wireframe)	60
Abbildung 38: Pilot – im Flug (Wireframe)	61
Abbildung 39: Pilot – Ende Flug	62
Abbildung 40: Helfer – Flug beitreten (Wireframe)	63
Abbildung 41: Helfer – im Flug (Wireframe)	64
Abbildung 42: Landing Screen (MockUp)	65
Abbildung 43: Pilot – Flugplanung (MockUp)	66
Abbildung 44: Pilot – im Flug (MockUp)	66
Abbildung 45: Pilot – Ende Flug (MockUp)	67
Abbildung 46: Helfer – Flug beitreten (MockUp)	68
Abbildung 47: Helfer – im Flug (MockUp)	68
Abbildung 48: Gesamter App-Ablauf in Adobe XD	69
Abbildung 49: Graustufen Wärmebild mit platziertem Hund	70
Abbildung 50: Zusätzliche Includeverzeichnisse in Visual Studio	72
Abbildung 51: Zusätzliche Abhängigkeiten in Visual Studio	72
Abbildung 52: Bildbereich der Wärmebildkamera (nicht maßstabsgetreu)	76
Abbildung 53: Bildbreite bei einer Flughöhe von 11m	76
Abbildung 54: Bildwinkel der Wärmebildkamera (nicht maßstabsgetreu)	76
Abbildung 55: Bildausschnitt bei einer Flughöhe von 11m	76
Abbildung 56: Präpariertes Wärmebild mit zwei eindeutigen Tieren.	77
Abbildung 57: Einzeichnung eines detektierten Rehkitzes.	77
Abbildung 58: USB-Debugging am Smartphone zulassen.	78
Abbildung 59: Gesamter Klassenbaum der BambiGuard Android App	80
Abbildung 60: Icons zur Steuerung während des Flugs (activity_pilot_flight.xml) . .	81

Abbildung 61: activity_flight_end.xml im Layout-Editor	82
Abbildung 62: Erstes Fragment des ViewPagers (fragment_helper_tutorial1.xml).	84
Abbildung 63: Kompass im Layout-Editor	85
Abbildung 64: AlertDialog, wenn keine Verbindung zum Server aufgebaut werden kann	86
Abbildung 65: Dialog beim Verbinden mit der Drohne	86
Abbildung 66: API-Schlüssel in der Web-Oberfläche von Mapbox	88
Abbildung 67: Mapbox Karte mit Position	91
Abbildung 68: Initiale Marker in der Karte	93
Abbildung 69: Individuell abgestecktes Feld mit Mapbox Markern	94
Abbildung 70: Optimaler Pfad des Algorithmus.	96
Abbildung 71: Anzahl der Kreuzungen eines Strahls mit den Kanten des Polygons	97
Abbildung 72: Aufbau des Debuggings	99
Abbildung 73: Flugsimulation im DJI Assistant 2.	100
Abbildung 74: Eine App im Developer-Tool von DJI erstellen	101
Abbildung 75: Mobile SDK Architektur	102
Abbildung 76: Android Studio External Tools.	109
Abbildung 77: Android Studio Tool – javah.	110
Abbildung 78: UML-Diagramm der Socket-Klassen	115
Abbildung 79: Auswahl USB-Debugging Xcode	117
Abbildung 80: WiFi-Debugger Xcode iPhone.	118
Abbildung 81: Das App Icon in verschiedenen Größen in Assets.xcassets	118
Abbildung 82: Safe Area im View Controller	119
Abbildung 83: Navigation Controller – Ablaufsverwaltung	119
Abbildung 84: Map und Plan View Controller in Xcode	120
Abbildung 85: Flug Helper View Controller in Xcode.	121
Abbildung 86: Search View Controller in Xcode	121
Abbildung 87: App-Keys für mehrere DJI-Apps (DJI Developer)	122
Abbildung 88: iOS App-Key für die BambiGuard App	122
Abbildung 89: App-Key in Xcode eintragen.	123
Abbildung 90: Ordnerstruktur nach CocoaPods-Installation.	123
Abbildung 91: UML-Diagramm des Servers	126
Abbildung 92: Port-Forwarding des Modems	127

Abbildung 93: DDNS-Eintrag von No-IP	128
Abbildung 94: Schriftmuster „Recherche Regular“	130
Abbildung 95: Schriftmuster „Mr Eaves Mod OT Regular“	130
Abbildung 96: Erster Entwurf des Logos	131
Abbildung 97: Korrektur des ersten Entwurfs des Logos	131
Abbildung 98: Verschiedene Farben am Entwurf des Logos	132
Abbildung 99: Logo Schattierung durch Grautöne	132
Abbildung 100: Das Logo in Affinity Designer	133
Abbildung 101: Finale Version des Logos	133
Abbildung 102: Logo Entwicklungsprozess	134
Abbildung 103: Website - Wireframe (Home; oberer Teil)	136
Abbildung 104: Website - Wireframe (Über uns; oberer Teil)	136
Abbildung 105: Website - MockUp (Home; oberer Teil)	138
Abbildung 106: Website - MockUp (Über uns; oberer Teil)	138
Abbildung 107: Verwendete Extension Visual Studio Code	139
Abbildung 108: Werbevideo Aufnahme bei niedrigen Temperaturen	146
Abbildung 109: Visitenkarte Randdefinitionen	148
Abbildung 110: Visitenkarte Vorderseite	148
Abbildung 111: Visitenkarte Vorderseite (Felix Teufel)	149
Abbildung 112: Microsoft PowerPoint Vorlage Titelfolie	149
Abbildung 113: Microsoft Word Formatvorlage ändern	150
Abbildung 114: Microsoft Word Vorlage	150
Abbildung 115: Zertifizierungssiegel	151
Abbildung 116: Flyer Seitenansichten	152
Abbildung 117: Flyer Gesamtansicht	154
Abbildung 118: Ein Post aus dem Instagram Account	154
Abbildung 119: Instagram Account von BambiGuard	155
Abbildung 120: Facebook Seite von BambiGuard	155
Abbildung 121: Projektstrukturplan	165
Abbildung 122: Projektstrukturplan Abschnitt 3 Realisierung	165
Abbildung 123: GANTT-Diagramm	168

6. Tabellenverzeichnis

Tabelle 1: Datenvergleich verschiedener Drohnenmodelle	6
Tabelle 2: Testergebnisse DJI Maivc 2 Enterprise Dual	9
Tabelle 3: Wichtige Events der Schnittstelle zwischen Pilot, Server und Helfer	54
Tabelle 4: Wichtige Optionen in der Konfigurationsdatei Application.mk (NDK)	112
Tabelle 5: BambiGuard Farben	130
Tabelle 6: Verfasserverzeichnis	164
Tabelle 7: Meilensteine	166
Tabelle 8: Projektphasen	167
Tabelle 9: Stundenliste Maximilian Strobl	169
Tabelle 10: Stundenliste Clemens Losbichler	169
Tabelle 11: Stundenliste Felix Teufel	170

7. Programmausdruckverzeichnis

Programmausdruck 1: OpenCV Funktion cvtColor	17
Programmausdruck 2: OpenCV Funktion adaptiveThreshold	18
Programmausdruck 3: OpenCV Funktion findContours	22
Programmausdruck 4: Inkludierung der C++ Bibliothek OpenCV	72
Programmausdruck 5: Funktion detectBambisInImage aus der Datei BambiGuard.h	73
Programmausdruck 6: Bildbearbeitende Funktinoen aus der Datei BambiGuard.h	74
Programmausdruck 7: Konturenfindung in der Datei BambiGuard.h	74
Programmausdruck 8: Kontrolle der Erkennung in der Datei BambiGuard.h	75
Programmausdruck 9: Berechnung des Pixel-zu-Meter Quotienten in BambiGuard.h	76
Programmausdruck 10: Einlesen und Konvertieren in BambiGuard.cpp	77
Programmausdruck 11: Verschiedene Funktionen zur Ausgabe eines Bildes; BambiGuard.h	77
Programmausdruck 12: Auszug aus der Datei .gitignore für das Android-Projekt	79
Programmausdruck 13: Farbdefinitionen in colors.xml	81
Programmausdruck 14: Eine Style-Definition in styles.xml	81
Programmausdruck 15: Toolbar in den Layout-Dateien der BambiGuard App	82
Programmausdruck 16: Ausschnitt aus FlightPlanningActivity.java	83
Programmausdruck 17: Ausschnitt aus JoinFlightActivity.java	83
Programmausdruck 18: PilotFlightActivity im Android Manifext (AndroidManifest.xml)	84
Programmausdruck 19: Alert Dialog in der Datei FlightPlanningActivity.java	85
Programmausdruck 20: Ausschnitt aus ConnectDroneDialogFragment.java	86
Programmausdruck 21: ListView Initialisierung in der Datei FlightPlanning2Frag- ment.java	87
Programmausdruck 22: API-Schlüssel in der Datei strings.xml	88
Programmausdruck 23: Maven Definition in der Datei build.gradle (Projekt-Ebene)	88
Programmausdruck 24: Mapbox Implementierung in der Datei build.gradle (Modul- Ebene)	89
Programmausdruck 25: Mapbox-Tag in der Datei activity_flight_planning1.xml	89
Programmausdruck 26: Karte übersetzen in der Datei FlightPlanning1Fragment. java	89
Programmausdruck 27: Definition der Berechtigung für die genauer Position in der Datei An-	

droidManifest.xml	90
Programmausdruck 28: Position Erfassen in der Datei FlightPlanning1Fragment.java	90
Programmausdruck 29: Funktion initLocationEngine in der Datei FlightPlanning1Fragment.java	90
Programmausdruck 30: Die Funktion initLocationLayer in der Datei FlightPlanning1Fragment.java	91
Programmausdruck 31: Die Funktion addMarker in der Datei FlightPlanning1Fragment.java	92
Programmausdruck 32: Meter in Latitude und Longitude umrechnen in der Datei FlightPlanning1Fragment.java	92
Programmausdruck 33: Initiale Marker positionieren in der Datei FlightPlanning1Fragment.java	92
Programmausdruck 34: Marker hinzufügen in der Datei FlightPlanning1Fragment.java	93
Programmausdruck 35: Marker verschieben in der Datei FlightPlanning1Fragment.java	94
Programmausdruck 36: Ausschnitt aus der Datei BambiGuardCoveragePlanner.java	98
Programmausdruck 37: DJI API-Schlüssel in der Datei AndroidManifest.xml	102
Programmausdruck 38: DJI SDK Implementierung in der Datei build.gradle (Modul-Ebene)	103
Programmausdruck 39: Ausschnitt aus der Datei DJIAplication.java	104
Programmausdruck 40: Berechtigungen in der Datei FlightPlanningActivity.java	104
Programmausdruck 41: Deklaration des USB-Accessoires in der Datei AndroidManifest.xml	105
Programmausdruck 42: Intent Filter und Meta-Tag der Activity FlightPlanningActivity in der Datei AndroidManifest.xml	105
Programmausdruck 43: DJI Widget in der Datei activity_pilot_flight.xml	105
Programmausdruck 44: MissionBuilder in der Datei BambiGuardMission.java	106
Programmausdruck 45: Bitmap aus der TextureView erstellen in der Datei PilotFlightActivity.java	107
Programmausdruck 46: Beispiel einer JNI Funktion	109
Programmausdruck 47: Beispiel einer Header-Datei erstellt mit dem Befehl javac -h	110
Programmausdruck 48: Mit dem Schlüsselwort extern „C“ gekennzeichnete JNI Methode	110
Programmausdruck 49: Beispiel für Array Operationen mit Java Native Types	111

Programmausdruck 50: Ausschnitt aus der JNI Konfigurationsdatei Android.mk	111
Programmausdruck 51: Ausschnitt aus der Konfigurationsdatei Application.mk	112
Programmausdruck 52: Task zum kompilieren des NDK-Quellcodes in der Datei build.gradle (Modul-Ebene)	112
Programmausdruck 53: Implementierung der Bibliothek socket.io in die Datei build.gradle (Modul-Ebene)	114
Programmausdruck 54: Das Attribut usesCleartextTraffic in der Datei AndroidManifest.xml	114
Programmausdruck 55: Die Funktion requestHelpersRecurring aus der Datei FlightPlanningActivity.java	115
Programmausdruck 56: Emittieren des Events bambi_found in der Datei PilotSocket.java	116
Programmausdruck 57: Auffangen des Events bambi_found in der Datei HelperFlightActivity.java	116
Programmausdruck 58: Definition der Abhängigkeiten in der Datei package.json	124
Programmausdruck 59: Importieren von Modulen in der Datei index.js	125
Programmausdruck 60: Auffangen des Events connection in der Datei index.js	125
Programmausdruck 61: Mit dem Node.js Server auf einen bestimmten Port hören; Datei index.js	125
Programmausdruck 62: Die Klasse Session in der Datei index.js	126
Programmausdruck 63: Simulieren eines Piloten, um die Funktionalität des Helfers zu testen; Datei index.js	126
Programmausdruck 64: Die Navbar-Komponente in der Datei Navbar.js	140
Programmausdruck 65: Die HeroSection-Komponente in der Datei HeroSection.js . .	141
Programmausdruck 66: Die Button-Komponente in der Datei Button.js	142
Programmausdruck 67: Eine definierte CardItem-Komponente in der Datei Cards.js . . .	143
Programmausdruck 68: Die Card-Item-Komponente in der Datei CardItem.js	143
Programmausdruck 69: Die PopUp-Komponente in der Datei Pupup.js	144
Programmausdruck 70: Die ImpressText-Komponente in der Datei ImpressText.js . . .	144
Programmausdruck 71: Routing-Ablauf in der Datei App.js	145
Programmausdruck 72: Das Ladeicon in den Seiten der BambiGuard-Website; in den Dateien des pages-Ordner	145

8. Anhang

8.1 Verfasserverzeichnis

Kapitel	Verfasser
1. Einleitung	Gemeinsam
2.1 Etablierte Lösungsansätze	Gemeinsam
2.2 Drohne	Gemeinsam
2.3 Mobile Applikationen	Clemens Losbichler
2.4 Bilderkennung	Clemens Losbichler
2.5 User Interface Design	Maximilian Strobl
2.6 Web-Frameworks	Maximilian Strobl
2.7 Gestaltgesetze	Felix Teufel
2.8 Werkzeuge und Wissen	Gemeinsam
3.2 Software-Architektur	Clemens Losbichler
3.3 App-Design	Maximilian Strobl und Clemens Losbichler
3.4 Bilderkennung	Clemens Losbichler
3.5 App für Android	Clemens Losbichler
3.6 App für iOS	Maximilian Strobl
3.7 Node.js Server	Clemens Losbichler
3.8 Repräsentation	Felix Teufel und Maximilian Strobl

Tabelle 6: Verfasserverzeichnis

8.2 Beigelegtes Speichermedium

Pfad	Inhalt
/App	Projektdateien der BambiGuard Android App (zu Öffnen mit Android Studio), der iOS App (zu Öffnen in Xcode) und Projektdateien sowie Export von Wireframe, Mockup und Prototyp der App
/Bilderkennung	Projektdateien der Bilderkennung (zu Öffnen mit Visual Studio)
/Projektmanagement	Besprechungsprotokolle, Stundenliste, Arbeitspaketliste, Projektstartdokumente und XML-Export aus der Planungs-Software Quire.io (die Datei ist in Quire.io zu importieren, um die Daten korrekt anzeigen zu lassen)
/Repräsentation	Projektdateien und Export von Infopaket (Flyer, Visitenkarten, Formatvorlagen), Logo und Website (Wireframe, Mockup und Programmierung)
/Schriftliche Arbeit	InDesign Buch der Diplomarbeit BambiGuard
/Server	Programmierung des Node.js Servers und Angabe der Abhängigkeiten (zu Installieren mit npm)

8.3 Projektstrukturplan

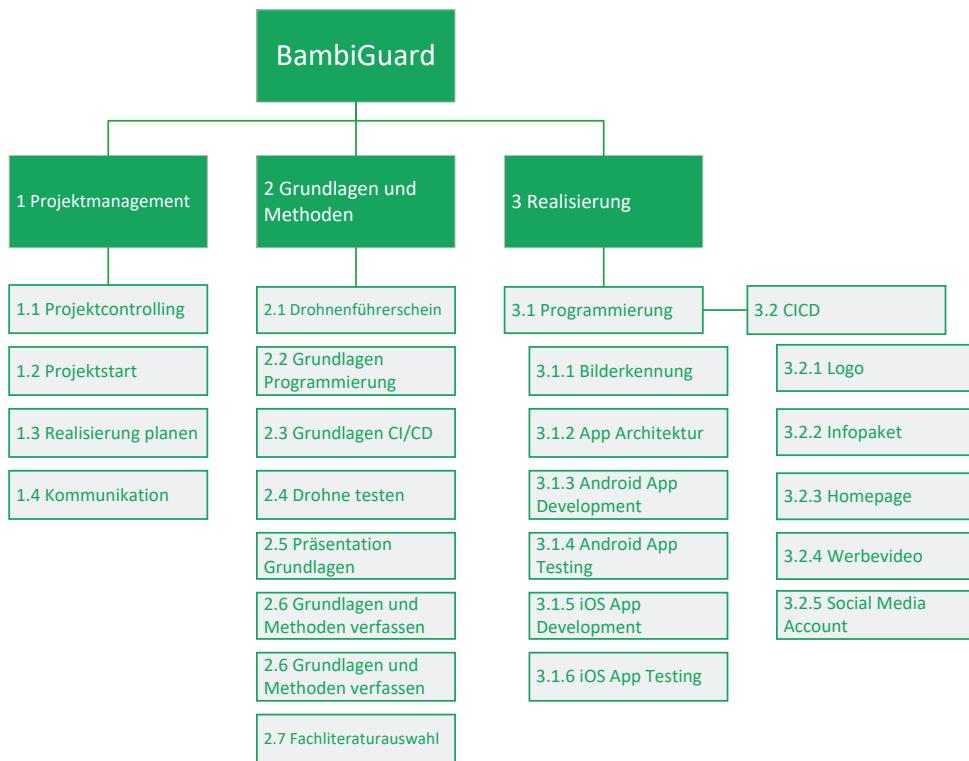


Abbildung 121: Projektstrukturplan

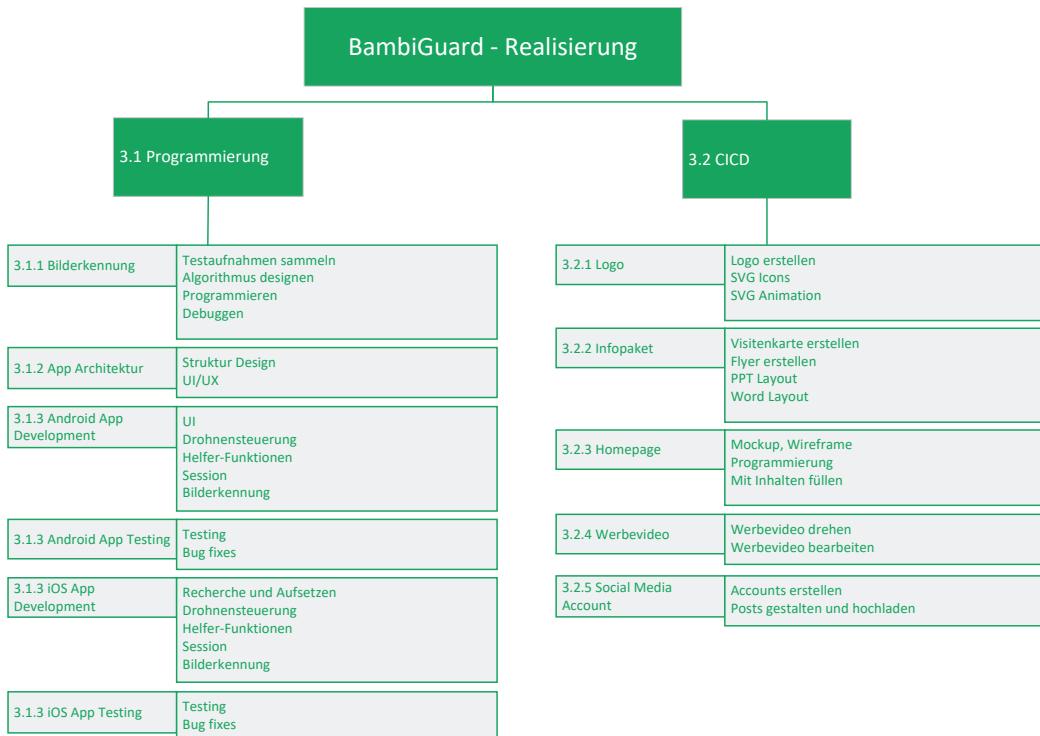


Abbildung 122: Projektstrukturplan Abschnitt 3 Realisierung

8.4 Meilensteine

PSP-Code	Name	Projektschnitt	Soll-Datum	Ist-Datum
3.1.1	Bilderkennung fertig	Bilderkennung	26.10.2021	31.10.2021
3.1.2.2	Front-End fertig	UI/UX	24.10.2021	01.11.2021
3.1.3.2	Pilot kann fliegen	Drohnensteuerung	09.11.2021	28.01.2022
3.1.3.4	Session funktioniert	Session	24.10.2021	28.01.2022
3.1.4	Android App fertig	Android App	11.12.2021	nicht erreicht
3.1.6	iOS App fertig	iOS App	31.12.2021	nicht erreicht
3.2.1	Logo fertig	Logo	11.10.2021	11.10.2021
3.2.2	Infopaket fertig	Infopaket	31.10.2021	29.10.2021
3.2.3	Homepage fertig	Homepage	26.11.2021	29.11.2021
3.2.4	Werbevideo fertig	Werbevideo	17.12.2021	nicht erreicht
3.2.5	Social Media Accounts fertig	Social Media	31.12.2021	04.03.2022
3.2	CI/CD fertig	CI/CD	10.01.2022	04.03.2022

Tabelle 7: Meilensteine

8.5 Projektphasen

PSP-Code	Name	Verantwortlicher	Soll-Stunden	Ist-Stunden
1.1	Projektcontrolling	M, C, F	30	28,5
1.2	Projektstart	M, C, F	34	34,6
1.3	Realisierung Planung	M, C, F	16	18,3
1.4	Kommunikation	M, C, F	50	51,7
2.1	Drohnenführerschein	M, C, F	15	12,1
2.2	Grundlagen Programmierung	C	13	14,2
2.3	Grundlagen CI/CD	F	16,5	15,4
2.4	Drohne testen	M	14	14,7
2.5	Präsentation Grundlagen	M, C, F	5,2	8,3
2.6	Grundlagen und Methoden Verfassen	M, C, F	61,5	51,4
2.7	Fachliteratur	M, C, F	3	0,6
3.1.1	Bilderkennung	C	18	19,6
3.1.2	App-Architektur	M, C	17,7	18,8
3.1.3	Android App Entwicklung	C	54	86,1
3.1.4	Android App Testing	C	10	4,5
3.1.5	iOS App Entwicklung	M	65	31,9
3.1.6	iOS App Testing	M	10	0,2
3.2.1	Logo	F	13	9,5
3.2.2	Infopaket	F	14	18,7
3.2.3	Website	M	36	42,6
3.2.4	Werbevideo	F	10	8
3.2.5	Social Media	F	12	12,5

Tabelle 8: Projektphasen

M Maximilian Strobl

C Clemens Losbichler

F Felix Teufel

8.6 GANTT-Diagramm

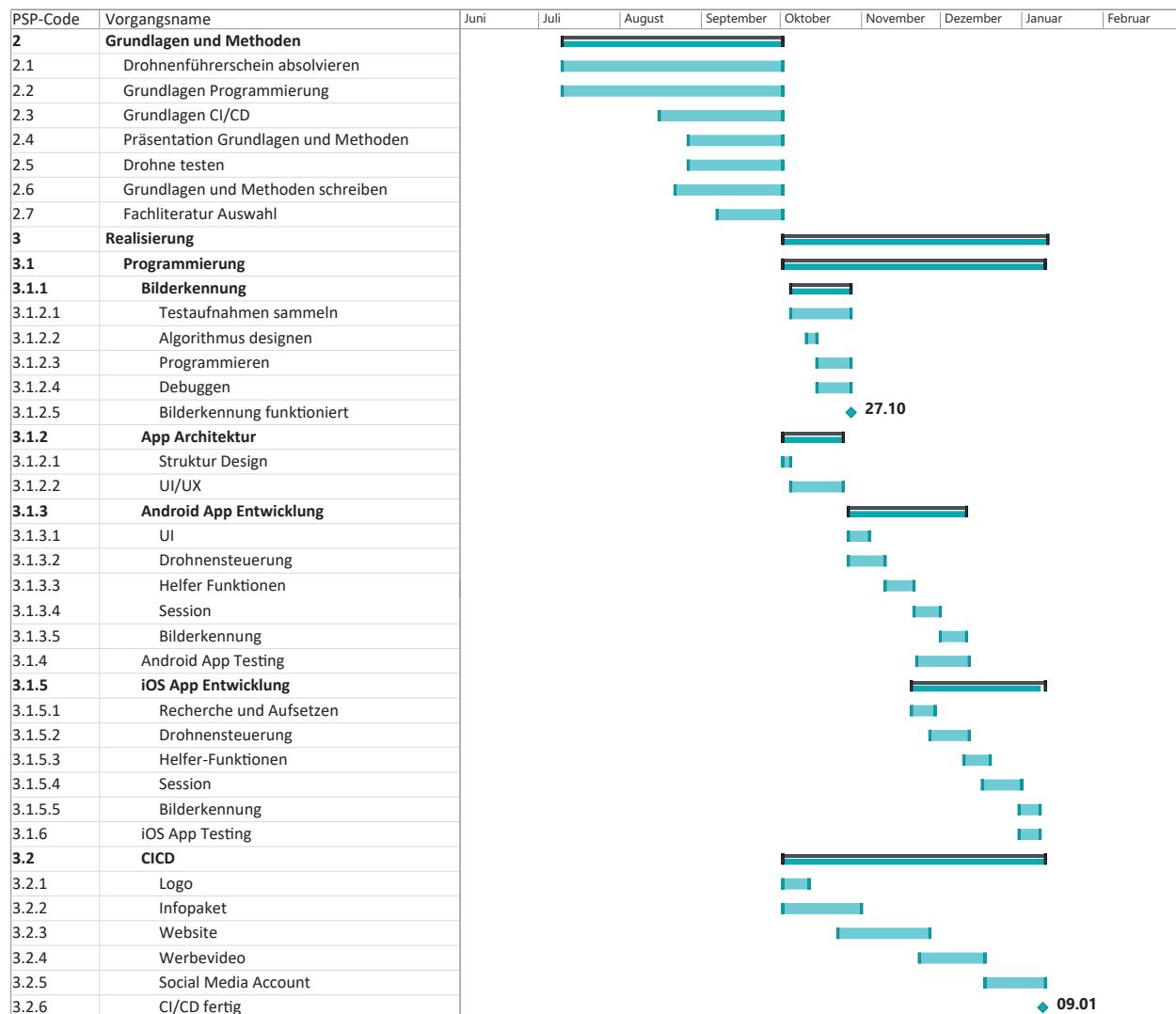


Abbildung 123: GANTT-Diagramm

8.7 Stundenlisten

Die genauen Aufzeichnungen sind auf dem Speichermedium zu finden (siehe 7.2 Beigelegtes Speichermedium).

Maximilian Strobl

Monat	Tätigkeiten	Aufwand in Stunden
06.2021	Projektstartdokumente; Arbeitspakete; Namensdefinition	3,9
08.2021	Pflichtenheft; Drohnenführerschein; Recherche Fachliteratur; Testen der Drohne	21,4
09.2021	Grundlagen Recherche; Planung Realisierung; Diplomarbeitsdatenbank; Grundlagen und Methoden	22,7
10.2021	App Modell; Wireframe; Mockup; Website Einrichtung	19,9
11.2021	Website Programmierung; App User Stories; Drohne Testmaterial; Beschäftigung mit Fachliteratur	34,6
12.2021	Werbefideo Dreh; Website überarbeitung; Mockup Fertigstellung	6,8
01.2022	Website Content; Wettbewerbe Einreichungen; iOS App Einrichtung und Layout	38,8
02.2022	iOS App DJI SDK; Schriftliche Arbeit verfassen	37,2
03.2022	Schriftliche Arbeit verfassen; Projektmanagement Abschluss	24,4
Summe		209,7 Stunden

Tabelle 9: Stundenliste Maximilian Strobl

Clemens Losbichler

Monat	Tätigkeiten	Aufwand in Stunden
06.2021	Projektstartdokumente; Arbeitspakete; GitLab Verzeichnis	4,5
07.2021	Arbeitspakete; Projektmanagement Definitionen	3
08.2021	DJI SDK Recherche und Einrichtung; Grundlagen Recherche; Grundlagen und Methoden; Drohne Testflug	23,8
09.2021	Grundlagen und Methoden; Android Demoprojekte; Drohnenführerschein; Planung Realisierung	23,7
10.2021	Bilderkennung; DJI Demoprojekte; Android App Einrichtung	22,3
11.2021	Android App Layout; Android Flugplanung	24,3
12.2021	Werbefideo Dreh; Android Flugplanung; Socket.io	31,7
01.2022	Node.js Server; Android Flugsteuerung; Android NDK Recherche und Implementierung; Android Helper Kompass; Coverage Path Planning	40,9
02.2022	Android App Testing; Schriftliche Arbeit verfassen	27,5
03.2022	Schriftliche Arbeit verfassen; Projektmanagement Abschluss	25
Summe		226,7 Stunden

Tabelle 10: Stundenliste Clemens Losbichler

Felix Teufel

Monat	Tätigkeiten	Aufwand in Stunden
06.2021	Projektstartdokumente; Arbeitspakete	2,3
07.2021	Arbeitspakete	1,5
08.2021	Drohnenführerschein; Planung Realisierung	10,7
09.2021	Drohnenführerschein; Planung Realisierung; Grundlagen und Methoden; Drohne testen	17,6
10.2021	Logo; SVG Icons; Visitenkarte; Flyer; Social Media Einrichtung; Formatvorlagen	25,5
11.2021	Flyer; Website Content	14,2
12.2021	Werbefideo Dreh; SVG Animation	3,1
01.2022	Werbefideo Bearbeitung; Social Media Posts; Wettbewerbe Dokumente	18,7
02.2022	Social Media Posts; Schriftliche Arbeit verfassen	23,5
03.2022	Schriftliche Arbeit verfassen; Wettbewerbe Dokumente	28,9
Summe		146 Stunden

Tabelle 11: Stundenliste Felix Teufel

9. Quellen und Literatur

Android: *Android's Kotlin-first approach.* Online in Internet: URL: <https://developer.android.com/kotlin/first>, 18.02.2022.

AustroControl: *Welche Regelungen sind beim Betrieb meiner Drohne einzuhalten?* Online in Internet: URL: <https://www.dronespace.at/open>, 19.02.2022.

Autel: *Spezifikationen Autel EVO II Dual.* Online in Internet: URL: <https://www.dronivo.de/Autel-EVO-II-Dual-640-Rugged-Bundle-V2>, 20.02.2022.

Bähnemann Rik: *Revisiting Boustrophedon Coverage Path Planning as a Generalized Traveling Salesman Problem.* Online in Internet: URL: <https://jenjenchung.github.io/anthropomorphic/Papers/Baehnemann2019revisiting.pdf>, 18.02.2022

Bergen L.; Burkhardt H.: *Morphologische Bildverarbeitung.* Online in Internet: URL: https://lmb.informatik.uni-freiburg.de/lectures/praktika_brox/bvpraktikum/BVAnl_morphologie.pdf, 18.02.2022

Budiu, Raluca: *Mobile: Native Apps, Web Apps, and Hybrid Apps.* Online in Internet: URL: <https://www.nngroup.com/articles/mobile-native-apps/>, 18.02.2022

Deutsche Wildtier Stiftung: *Praxisratgeber Mähtod.* Online in Internet: URL: <https://lebensraum-brache.de/wp-content/uploads/2019/07/Praxisratgeber-M%C3%A4htod-2019-2.pdf>, 21.02.2022

DJI Mavic: *Technische Daten Mavic 2 Enterprise.* Online in Internet: URL: <https://www.dji.com/at/mavic-2-enterprise/specs>, 20.02.2022

DJI Zenmuse: *Technische Daten Zenmuse H20 Serie.* Online in Internet: URL: <https://www.dji.com/at/zenmuse-h20-series/specs>, 20.02.2022

Duden: *Rechtschreibung Bilderkennung.* Online in Internet: URL: <https://www.duden.de/rechtschreibung/Bilderkennung>, 18.02.2022

EASA: *The European UAS Regulation.* Online in Internet: URL: https://www.easa.europa.eu/sites/default/files/dfu/EU_UAS_Regulation_introduction_and_open-category.pdf, 19.02.2022

ECMA Standard: *Dart programming language specification.* Online in Internet: URL: <https://www.ecma-international.org/publications-and-standards/standards/ecma-408/>, 18.02.2022

Empatic: *Was ist eigentlich UX-Design?* Online in Internet: URL: <https://empatic-ux.com/blog/was-ist-ux/>, 19.02.2022

Ermigotti, Lorenzo: *17 JavaScript Frameworks that You Should Know in 2021 – A Comprehensive Guide.* Online in Internet: URL: <https://www.codemotion.com/magazine/frontend/javascript-javascript-frameworks-guide/#1-angular>, 19.02.2022

Farley, Patrick: *What is Image Analysis?* Online in Internet: URL: <https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-image-analysis>, 18.02.2022

Flutter: *Flutter architectural overview.* Online in Internet: URL: <https://docs.flutter.dev/resources/architectural-overview>, 18.02.2022

Flutter: *Introduction to widgets.* Online in Internet: URL: <https://docs.flutter.dev/development/ui/widgets-intro>, 18.02.2022

Flutter: *Dart overview.* Online in Internet: URL: <https://dart.dev/overview>, 18.02.2022

Google Developers: *Nerual Networks: Structure.* Online in Internet: URL: <https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/anatomy>, 18.02.2022

Hmara, Anna: *Swift vs Objective-C: Which is Better to Chosse?* Online in Internet: URL: <https://key-ua.org/blog/swift-vs-objective-c-which-is-better/>, 18.02.2022

IONOS: *Webframeworks – Überblick und Klassifizierung.* Online in Internet: URL: <https://www.ionos.at/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/>, 19.02.2022

Liferay: *Was ist User Experience?* Online in Internet: URL: <https://www.liferay.com/de/resources/l-user-experience>, 19.02.2022

Ludwig, Jamie: *Image Convolution.* Online in Internet: URL: https://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf, 18.02.2022

Mallik, Satya: *Image Recognition and Object Detection: Part 1.* Online in Internet: URL: <https://learnonopencv.com/image-recognition-and-object-detection-part1/>, 18.02.2022

Meyerhofer, Markus: *Introduction to Artificial Intelligence.* IT-HTL Ybbs an der Donau, 2021

Microsoft: *Was ist Xamarin?* Online in Internet: URL: <https://docs.microsoft.com/de-de/xamarin/get-started/what-is-xamarin>, 18.02.2022

Microsoft: *Xamarin Aufbau.* Online in Internet: URL: <https://docs.microsoft.com/de-de/xamarin/android/internals/architecture>, 18.02.2022

Microsoft: *Xamarin Binden einer Java-Bibliothek.* Online in Internet: URL: [https://docs.microsoft.com/de-de/xamarin/android/platform/binding-java-library/](https://docs.microsoft.com/de-de/xamarin/android/platform/binding-java-library), 18.02.2022

OpenCV: *OpenCV Introduction.* Online in Internet: <https://docs.opencv.org/4.5.5/d1/dfb/intro.html>, 18.02.2022

OpenCV: *Morphological Transformations.* Online in Internet: URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html, 18.02.2022

Parrot: *Anafi Thermal Datenblatt.* Online in Internet: URL: <https://www.parrot.com/assets/s3fs-public/2021-02/anafi-thermal-product-sheet-white-paper-en.pdf>, 20.02.2022

Pirringer, Jürgen: *Wahrnehmung und Erkenntnis.* IT-HTL Ybbs an der Donau, 19.02.2022

React: *Getting Started.* Online in Internet: URL: <https://reactjs.org/docs/getting-started.html>, 19.02.2022

Rechtsinformationssystem des Bundes: *Gesamte Rechtsvorschrift für Tierschutzgesetz.* Online in Internet: URL: <https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20003541&ShowPrintPreview=True>, 10.03.2022

Rohles, Björn: *Mediengestaltung Der Ausbildungsbegleiter.* Rheinwerk Design Verlag, 2019

Schürmann, Tim: *Von Dotnet über Novell-Suse zu Xamarin und mobilen Devices: Mono.* Online in Internet: URL <https://www.linux-magazin.de/ausgaben/2011/11/mono/>, 18.02.2022

Simran, Kaur Arora: *10 Best JavaScript Frameworks to Use in 2022.* Online in Internet: URL: <https://hackr.io/blog/best-javascript-frameworks>, 19.02.2022

Swift: *About Swift.* Online in Internet: URL: <https://www.swift.org/about/>, 18.02.2022

TensorFlow: *Grundlegende Dokumentation.* Online in Internet: URL: <https://www.tensorflow.org/guide>, 18.02.2022

Thesemann, Stephan: *Interface Design.* Springer Verlag, 2022

Tiroler Jagdverband: *Rehkitzrettung.* Online in Internet: URL: <https://rehkitzrettung.at/>, 21.02.2022

Tiroler Jagdverband: *Rehkitzrettung: Gemeinsam gegen den Mähtod.* Online in Internet: URL: <https://www.tjv.at/rehkitzrettung-gemeinsam-gegen-den-maehtod/>, 10.03.2022

Wagner, Johann: *Schach dem Mähtod.* Online in Internet: URL: [https://jagdwirt.at/DesktopModules/ContentList/Uploads/Schach dem Maehtod_Wagner J_final.pdf](https://jagdwirt.at/DesktopModules/ContentList/Uploads/Schach%20dem%20Maehtod_Wagner%20J_final.pdf), 10.03.2022

Webzeile: *Was ist ein Frontend-Framework?* Online in Internet: URL: <https://www.webzeile.com/de/blog/was-ist-ein-frontend-framework>, 19.02.2022

Yablonski, Jan: *Laws of UX.* O'REILLY, 2020