Abteilung: Höhere Lehranstalt für Informatik

Diplomarbeit

Höhere Abteilung für Informatik

Thema:	JagdEizendorf - Abschussplaner
eingereicht von:	Max Kühberger ‹max.kuehberger@gmail.com›
eingereicht am:	17. August 2025
Betreuer:	Dipl.-Ing. Michael Romani
In Zusammenarbeit mit:	Jagdgenossenschaft Eizendorf

1 Eidesstattliche Erklärung

Die unterfertigten Kandidaten / Kandidatinnen haben gemäß § 34 (3) SchUG in Verbindung mit § 22 (1) Zi. 3 lit. b der Verordnung über die abschließenden Prüfungen in den berufsbildenden mittleren und höheren Schulen, BGBl. II Nr. 70 vom 24.02.2000 (Prüfungsordnung BMHS), die Ausarbeitung einer Diplomarbeit mit der umseitig angeführten Aufgabenstellung gewählt.

Die Kandidaten / Kandidatinnen nehmen zur Kenntnis, dass die Diplomarbeit in eigenständiger Weise und außerhalb des Unterrichtes zu bearbeiten und anzufertigen ist, wobei Ergebnisse des Unterrichtes mit einbezogen werden können.

Die Abgabe der Diplomarbeit hat bis spätestens 18.08.2025 beim zuständigen Betreuer / der zuständigen Betreuerin zu erfolgen.

Die Kandidaten / Kandidatinnen nehmen weiters zur Kenntnis, dass gemäß § 9 (6) der Prüfungsordnung BMHS nur der Schulleiter bis spätestens Ende des vorletzten Semesters den Abbruch einer Diplomarbeit anordnen kann, wenn diese aus nicht beim Prüfungskandidaten (bei den Prüfungskandidaten) gelegenen Gründen nicht fertiggestellt werden kann.

Kandidaten / Kandidatinnen inkl. Unterschrift:

Saxen, 17.08.2025
Ort, Datum

Max Kühberger

2 Danksagung

An dieser Stelle möchte ich mich herzlich bei all jenen bedanken, die mich während der Anfertigung dieser Diplomarbeit unterstützt, begleitet und motiviert haben.

Mein besonderer Dank gilt der Jagdgenossenschaft Eizendorf, die mir als Auftraggeber die Möglichkeit gegeben hat, den Abschussplaner JagdEizendorf zu entwickeln. Die stets offene und hilfsbereite Zusammenarbeit hat maßgeblich zum Gelingen dieses Projekts beigetragen und die Arbeit daran nicht nur lehrreich, sondern auch sehr angenehm gemacht.

Ein großes Dankeschön geht auch an Herrn Dipl.-Ing. Michael Romani für die fachliche Betreuung und Begutachtung dieser Arbeit. Seine konstruktive Kritik, seine wertvollen Anregungen und seine Unterstützung haben wesentlich zur Qualität dieser Diplomarbeit beigetragen.

Ebenso danke ich Cora Hofstetter für das sorgfältige Korrekturlesen und die hilfreichen Hinweise zur sprachlichen Gestaltung der Arbeit.

Mein aufrichtiger Dank gilt auch meiner Familie und meinen Freunden. Ihre Geduld, ihr Verständnis und ihre beständige emotionale Unterstützung haben mir in jeder Phase Mut gemacht – besonders in den anspruchsvollen Momenten dieses Projekts. Es ist ein großes Geschenk, Menschen an seiner Seite zu wissen, die immer ein offenes Ohr und ein gutes Wort parat haben.

Vielen Dank!

Max Kühberger

Inhaltsverzeichnis

3 Kurzfassung

Die Jagdgenossenschaft Eizendorf nutzt aktuell ein traditionelles Protokollbuch, in dem die Abschüsse der Wildtiere eingetragen werden. Diese Abschüsse müssen während des gesamten Jagdjahres regelmäßig und zeitgerecht an die Bezirkshauptmannschaft Perg gemeldet werden. Obwohl es in der Theorie ideal wäre, die Meldungen unmittelbar nach dem Erlegen eines Tieres vorzunehmen, gibt es in der Praxis verschiedene Schwierigkeiten, die diesen Prozess erschweren. Häufig treten Probleme wie etwa eine schwer leserliche Schrift oder Blutrückstände auf den Seiten des Eintragungsbuches auf, was zu Unsicherheiten und Fehlern bei der Erfassung führt.

Darüber hinaus müssen auch Tiere, die in Lebendfallen gefangen wurden, sowie Fallwild – also Tiere, die durch Verkehrsunfälle oder andere unvorhergesehene Ereignisse zu Tode kommen – kontinuierlich gemeldet werden. Da es bisher keine zentrale, strukturierte Möglichkeit zur Erfassung dieser Tiere gab und die Meldungen auf anderem Wege, etwa über eine Whatsapp-Gruppe, erfolgen mussten, bestand immer die Gefahr, dass wichtige Informationen übersehen oder vergessen wurden.

Ziel dieser Diplomarbeit ist es, eine Applikation zu entwickeln, die es den Jägern ermöglicht, ihre Abschüsse direkt und ohne Umwege zu melden. Dies würde nicht nur die unübersichtliche Anzahl an Dokumenten eliminieren, sondern auch eine genauere und effizientere Erfassung der Wildtierabschüsse ermöglichen. Die Jäger können ihre Meldungen sofort und digital durchführen, wodurch Fehlerquellen wie unleserliche Handschrift oder Verzögerungen bei der Weiterleitung von Informationen vermieden werden. Gleichzeitig erhält der Jagdleiter die Möglichkeit, die Abschussliste jederzeit zu exportieren und detaillierte Statistiken zu erstellen, um eine genauere Auswertung der Jagdaktivitäten und der gejagten Tierarten zu erhalten.

Diese Applikation soll somit einen wesentlichen Beitrag zur Optimierung der Dokumentation und Meldung von Abschüssen und anderen relevanten Jagdereignissen leisten und den Arbeitsaufwand der Jäger sowie des Jagdleiters deutlich reduzieren.

4 Abstract

The Eizendorf Hunting Association currently uses a traditional logbook to record hunting kills. These kills must be reported regularly and in a timely manner to the district authority of Perg throughout the hunting year. While it would ideally be best to report the kills immediately after an animal is taken, there are several practical issues that complicate this process. Common problems include illegible handwriting or blood stains on the pages of the logbook, which lead to uncertainties and errors in recording.

In addition, animals trapped in live traps and roadkill (animals that have been killed in traffic accidents or other unforeseen events) also need to be continuously reported. Since there has not been a centralized or structured way to record these incidents, such reports have previously been made through a WhatsApp group. However, this method posed the risk of important information being overlooked or forgotten.

The aim of this thesis is to develop an application that will allow hunters to directly report their kills. This will not only eliminate the cumbersome ”paperwork,”but also enable a more accurate and efficient record-keeping process. Hunters will be able to make their reports immediately and digitally, thereby avoiding errors caused by illegible handwriting or delays in transmitting information. At the same time, the hunting leader will have the ability to export the current kill list at any time and generate detailed statistics to better understand where and which species of animals were killed. This application will therefore contribute significantly to optimizing the documentation and reporting of kills and other relevant hunting events, while also reducing the workload of both the hunters and the hunting leader.

5 Einleitung

5.1 Ausgangssituation

5.1.1 Aktuelle Situation der Jagdgenossenschaft Eizendorf

Die Jagdgenossenschaft Eizendorf verwaltet ihre Abschussplanung und -dokumentation derzeit noch manuell in einem Protokollbuch. Diese Vorgehensweise erfolgt überwiegend mittels handschriftlicher Notizen und unsystematischer Dokumentation. Diese Praxis führt häufig zu Missverständnissen, Unklarheiten sowie einem erhöhten organisatorischen Aufwand, insbesondere hinsichtlich der Nachvollziehbarkeit und Abstimmung innerhalb der Jagdgemeinschaft.

5.2 Zielsetzung

Ziel des vorliegenden Projekts ist die Entwicklung und Einführung einer digitalen Anwendung, die die bislang manuelle und papierbasierte Abschussverwaltung innerhalb der Jagdgenossenschaft Eizendorf ablösen soll. Die Applikation ermöglicht eine zentrale Erfassung, übersichtliche Darstellung sowie jederzeitige Abrufbarkeit aller relevanten Abschussdaten für die beteiligten Akteure. Hierdurch sollen Missverständnisse sowie organisatorische Unklarheiten vermieden werden. Darüber hinaus trägt die Anwendung zur Verbesserung der internen Kommunikation, zur Erleichterung der Abstimmung innerhalb der Jagdgemeinschaft sowie zur Steigerung der Effizienz bei Planung und Dokumentation jagdlicher Maßnahmen bei

6 Grundlagen

6.1 Technische Grundlagen

6.1.1 Verwendete Technologien

React Native

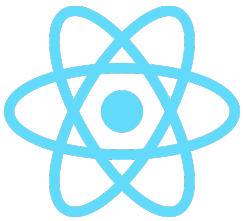


Abbildung 6.1: React Native Logo

React Native gehört zu den beliebtesten Frameworks für die plattformübergreifende App-Entwicklung. Die zugrunde liegende Programmiersprache ist JavaScript, wobei häufig auch TypeScript verwendet wird, um eine bessere Strukturierung und Typisierung des Codes zu ermöglichen. React Native ist ein Community-Framework, das kontinuierlich weiterentwickelt wird und zahlreiche beliebte Bibliotheken aus dem JavaScript-Ökosystem integriert. Dadurch entsteht ein sehr leistungsfähiges und entwicklerfreundliches Framework, das sich besonders gut für moderne App-Projekte eignet.
Die wohl größte Stärke von React Native ist die Möglichkeit, mit nur einer Codebasis gleichzeitig Apps für Android und iOS zu entwickeln. Dadurch entfällt die Notwendigkeit, zwei separate Anwendungen zu pflegen, was Zeit und Ressourcen spart. Zusätzlich kann die App auf Wunsch mit nativen Modulen erweitert werden, wenn spezielle Funktionen oder Gerätezugriffe erforderlich sind. Ein weiterer Vorteil ist die große Auswahl an bereits existierenden Komponenten und UI-Bibliotheken, die sich flexibel anpassen und erweitern lassen.
Auch das Erlernen von React Native gestaltet sich vergleichsweise einfach. Das Framework ist sehr gut dokumentiert, und durch die große weltweite Community existiert eine Vielzahl an Tutorials, Forenbeiträgen und Blogartikeln, die bei der Entwicklung unterstützen. Besonders für kleinere bis mittelgroße Projekte mit klar definierten Anforderungen – wie in diesem Fall die Entwicklung einer App zur digitalen Abschussverwaltung – bietet React Native eine leistungsstarke und effiziente Lösung.

Google Firebase

Abbildung 6.2: Firebase Logo

Firebase ist eine Plattform von Google, die eine Vielzahl an Backend-Diensten für die Entwicklung moderner Web- und Mobilanwendungen bereitstellt.

Ein wesentlicher Vorteil von Firebase ist die Integration zahlreicher Dienste in einer einzigen Plattform. Dazu zählen unter anderem eine Echtzeit-Datenbank (Realtime Database), Cloud Firestore, Authentifizierung, Cloud Functions, Cloud Storage und ein Hosting-Service. Besonders hervorzuheben ist die nahtlose Integration mit anderen Google-Diensten sowie die gute Unterstützung für mobile Frameworks wie React Native, was es zu einer idealen Backend-Lösung für viele App-Projekte macht.

Die große Stärke von Firebase liegt in der schnellen Umsetzbarkeit von Ideen. Entwickler können sich auf die Anwendungslogik konzentrieren, während Firebase zentrale Funktionen wie Benutzerverwaltung, Datenspeicherung, Benachrichtigungen oder Serverlogik übernimmt. Durch die umfangreiche Dokumentation und die tiefe Integration in bestehende Entwicklungsumgebungen (z. B. Android Studio oder Visual Studio Code) ist der Einstieg in Firebase relativ einfach.

Für dieses Projekt wurde Firebase als Backend-as-a-Service (BaaS) gewählt, da es eine stabile, cloudbasierte und gleichzeitig flexible Infrastruktur bietet – insbesondere im Hinblick auf die Echtzeit-Datenverarbeitung und die unkomplizierte Nutzerverwaltung.

6.1.2 Entwicklungssysteme

Expo Go

Abbildung 6.3: Expo Go Logo

Expo Go ist eine mobile Anwendung, die Entwicklern ermöglicht, React Native Projekte schnell und unkompliziert auf physischen Geräten zu testen, ohne die App jedes Mal neu kompilieren und installieren zu müssen. Die App unterstützt eine nahtlose Vorschau und Echtzeit-Updates während der Entwicklung, was den Entwicklungsprozess erheblich beschleunigt.

Im Rahmen dieser Diplomarbeit wurde Expo Go verwendet, um die entwickelte React Native Anwendung direkt auf mobilen Endgeräten zu testen. Dies ermöglichte eine realistische Überprüfung der Funktionalitäten und des Nutzererlebnisses unter realen Bedingungen, ohne auf den Emulator oder eine manuelle Installation angewiesen zu sein.

Dank der engen Integration mit dem Expo-Framework unterstützt Expo Go verschiedene Plattformen (iOS und Android) und erleichtert so die plattformübergreifende Entwicklung und Fehlerbehebung. Aufgrund dieser Vorteile wurde Expo Go als wichtiges Werkzeug für das Testing und die Qualitätssicherung der App eingesetzt.

6.2 Theoretische Grundlagen

Der Abschuss eines Wildtieres stellt einen zentralen Bestandteil der nachhaltigen Jagd- und Wildtierbewirtschaftung dar. Er dient nicht nur der Regulierung der Wildbestände, sondern auch dem Erhalt eines ökologischen Gleichgewichts in der jeweiligen Region. Der Prozess des Abschusses ist dabei klar strukturiert und erfolgt unter Beachtung gesetzlicher Vorgaben sowie fachlicher Standards.

Zunächst erfolgt der tatsächliche Abschuss durch den Jäger. Nach erfolgreicher Erlegung eines Tieres dokumentiert der Jäger sämtliche relevanten Informationen zum geschossenen Wild. Dazu zählen unter anderem Angaben zur Art, zum Geschlecht, zum Alter, zum Körpergewicht sowie der genaue Zeitpunkt und Ort des Abschusses. Diese Daten bilden die Grundlage für eine lückenlose Nachverfolgung und Auswertung des Wildbestandes.

Im Anschluss wird das Tier von einer kundigen Person, häufig einem Wildbiologen, Förster oder

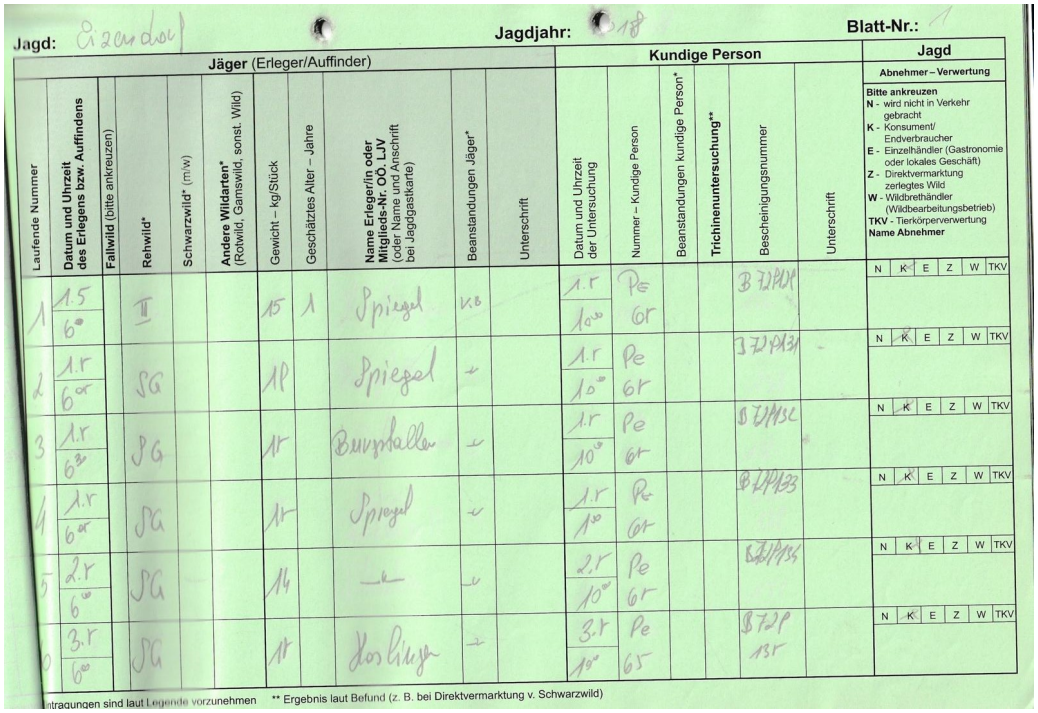


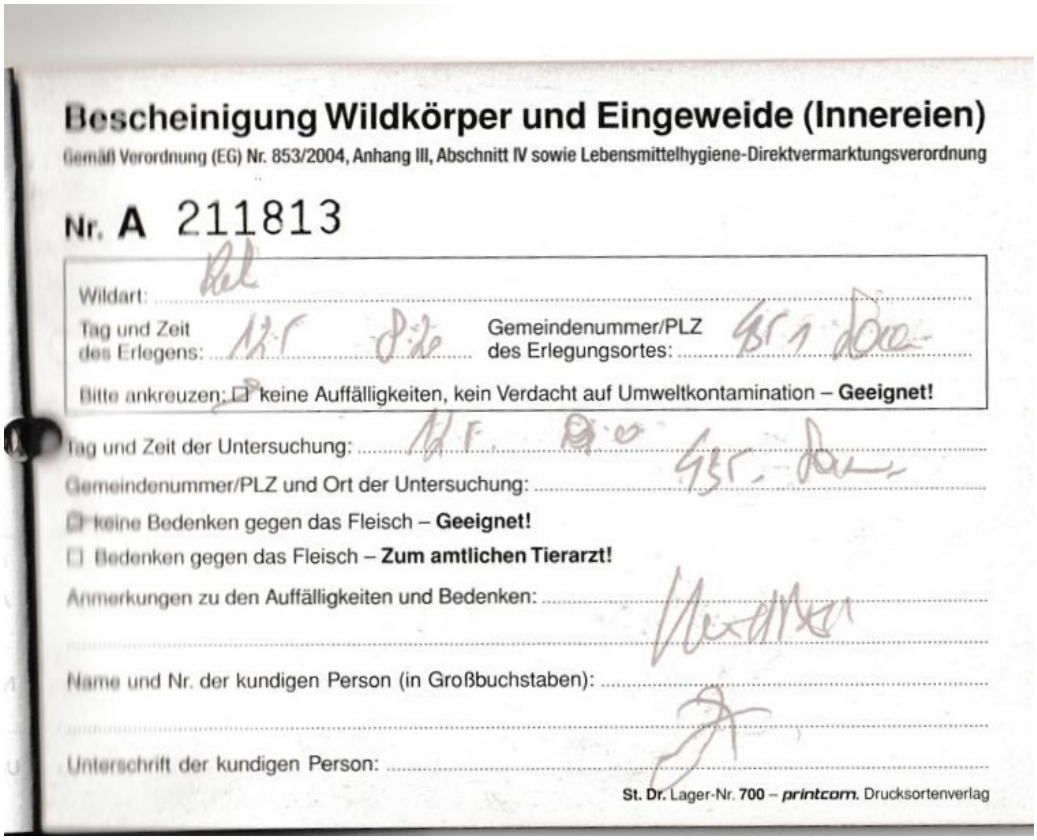
Abbildung 6.4: Beispielseite aus dem aktuell verwendeten Abschussbuch in Papierform

einem erfahrenen Jagdaufseher, begutachtet. Diese Person überprüft das Tier auf gesundheitliche Auffälligkeiten, Verletzungen oder Anzeichen von Krankheiten, welche Rückschlüsse auf den Zustand der Population und das Ökosystem zulassen. Diese fachliche Begutachtung ist von großer Bedeutung, um potenzielle Gesundheitsrisiken im Wildbestand frühzeitig zu erkennen und gegebenenfalls Maßnahmen zu ergreifen.

Zur eindeutigen Identifikation und Nachverfolgung wird dem erlegten Tier eine individuelle Nummer zugewiesen. Diese Nummer dient als eindeutiges Identifikationsmerkmal und ermöglicht eine effiziente Verwaltung und Dokumentation in Jagdbüchern oder digitalen Abschussverwaltungsprogrammen. Durch die Vergabe einer solchen Nummer kann das Tier bei späteren Auswertungen, Untersuchungen oder auch bei der Erstellung von Statistiken gezielt erfasst und nachvollzogen werden.

Ein weiterer wichtiger Schritt im Prozess ist die regelmäßige Meldung der geschossenen Tiere an die zuständige Bezirkshauptmannschaft. Diese Meldungen erfolgen in festgelegten Zeitintervallen und werden durch den Jagdleiter koordiniert. Die Übermittlung der Abschussdaten an die Behörden dient der Überwachung und Kontrolle der Wildbestände auf regionaler Ebene und gewährleistet die Einhaltung jagdrechtlicher Vorschriften.

Der gesamte Ablauf vom Abschuss über die Dokumentation bis hin zur Begutachtung, eindeutigen Kennzeichnung und Meldung des Wildtieres an die Behörden gewährleistet eine transparente und fachlich fundierte Wildtierbewirtschaftung. Dies trägt maßgeblich zur Erhaltung eines gesunden Wildbestandes und zur Sicherstellung einer nachhaltigen Jagdpraxis bei.



Bescheinigung Wildkörper und Eingeweide (Innereien)

Gemäß Verordnung (EG) Nr. 853/2004, Anhang III, Abschnitt IV sowie Lebensmittelhygiene-Direktvermarktungsverordnung

Nr. A 211813

Wildart: Reh

Tag und Zeit des Erlegens: 12.5. 8:20 Uhr

Gemeindenummer/PLZ des Erlegungsortes: 95100

Bitte ankreuzen: ☒ keine Auffälligkeiten, kein Verdacht auf Umweltkontamination – **Geeignet!**

Tag und Zeit der Untersuchung: 12.5. 9:00 Uhr

Gemeindenummer/PLZ und Ort der Untersuchung: 95... Hau...

☒ keine Bedenken gegen das Fleisch – **Geeignet!**

☐ Bedenken gegen das Fleisch – **Zum amtlichen Tierarzt!**

Anmerkungen zu den Auffälligkeiten und Bedenken: Metzler

Name und Nr. der kundigen Person (in Großbuchstaben):

Unterschrift der kundigen Person:

St. Dr. Lager-Nr. 700 – printcom. Drucksortenverlag

Abbildung 6.5: Beispielseite aus dem aktuell verwendeten Abschussbuch zur vergabe der laufenden Identifikationsnummer in Papierform

7 Planung

7.1 Meilensteine

Die folgenden Meilensteine wurden im Voraus definiert. Sie zeigen das Ergebnis, welches erreicht wurden als auch den geschätzten Soll-Termin und dem tatsächlichen Zeitpunkt der Erreichung.

Meilenstein	Ergebnis	Soll-Termin	Ist-Termin
1	Projektinitialisierungsphase abgeschlossen	01.04.2025	01.04.2025
2	Recherchearbeiten abgeschlossen	14.04.2025	10.04.2025
3	User Login möglich	10.05.2025	30.04.2025
4	Fall eines Tieres kann gemeldet werden	01.06.2025	10.05.2025
5	Statistik kann erstellt und versendet werden	10.06.2025	10.06.2025
6	JagdEizendorf fertig	15.06.2025	20.06.2025

7.2 Kommunikation

Im Rahmen der Diplomarbeit wurde das Projekt vollständig eigenverantwortlich durchgeführt. Diese selbstständige Arbeitsweise ermöglichte es mir, sämtliche Aufgaben – von der Planung über die Implementierung bis hin zur Dokumentation – eigenständig zu steuern und umzusetzen.

Die Kommunikation mit dem Auftraggeber fand persönlich statt. Diese direkte Interaktion erleichterte den Informationsaustausch und förderte ein gemeinsames Verständnis der Projektanforderungen sowie der gewünschten Ergebnisse. Durch den persönlichen Kontakt konnten Rückfragen schnell geklärt und notwendige Anpassungen frühzeitig besprochen werden.

Parallel dazu erfolgte die fachliche Betreuung durch den Betreuungslehrer, mit dem während der gesamten Projektlaufzeit regelmäßiger telefonischer Kontakt gepflegt wurde. Diese telefonischen Abstimmungen dienten dazu, den Fortschritt zu reflektieren, Feedback zu erhalten und offene Fragen zu klären.

Darüber hinaus wurden im Verlauf der Implementierung regelmäßig Statusmeetings abgehalten. Diese Meetings hatten das Ziel, den aktuellen Stand des Projekts zu dokumentieren, Herausforderungen zu identifizieren und Lösungsansätze zu diskutieren. Die strukturierte und kontinuierliche Kommunikation mit allen Beteiligten trug wesentlich dazu bei, den Projektverlauf transparent und zielgerichtet zu gestalten sowie eine qualitativ hochwertige Umsetzung sicherzustellen.

8 Realisierung

8.1 Grundaufbau des Programmes

8.1.1 Architektur des Programms

Die Architektur der React Native App zur Jagdabschussverwaltung basiert auf einem modularen und komponentenorientierten Aufbau, der eine klare Trennung der Verantwortlichkeiten gewährleistet. Ziel ist es, eine skalierbare, wartbare und erweiterbare Anwendung bereitzustellen.

Client-Server-Architektur

Die App folgt dem Prinzip einer Client-Server-Architektur: Die React Native App bildet den Client, der die Benutzeroberfläche bereitstellt und die Interaktion mit dem Nutzer steuert. Das Backend wird durch Firebase realisiert, welches als serverloser Cloud-Dienst sowohl die Datenhaltung (Firestore) als auch die Authentifizierung (Firebase Auth) übernimmt.
Diese Architektur ermöglicht eine schlanke App mit minimalem Wartungsaufwand, da der Großteil der Logik und der Datenverwaltung in der Cloud stattfindet.

Modulare Komponentenstruktur

Im Frontend ist die Anwendung in verschiedene Komponenten unterteilt, die jeweils spezifische Funktionen übernehmen, wie beispielsweise:

- Erfassung und Verwaltung von Abschussdaten
- Nutzeranmeldung und -verwaltung
- Anzeige von Statistiken und Berichten

Durch diese Modularisierung lassen sich einzelne Teile der App unabhängig voneinander entwickeln, testen und erweitern.

Datenfluss und Kommunikation

Die Kommunikation zwischen Frontend und Backend erfolgt über die Firebase SDKs. Die App liest und schreibt Daten in die Firestore-Datenbank in Echtzeit. Änderungen werden automatisch synchronisiert, sodass alle Nutzer stets aktuelle Daten sehen.
Die Nutzer authentifizieren sich über Firebase Auth, wodurch Zugriffsrechte zentral gesteuert und gesichert werden.

8.1.2 Datenmodell

Allgemeiner Aufbau einer Firebase-Datenbank

Firebase Firestore ist eine NoSQL-Datenbank, die Daten in hierarchischen Strukturen speichert. Statt klassischer Tabellen wie bei relationalen Datenbanken verwendet Firestore *Collections* und *Dokumente*:

- **Collection:** Eine Sammlung von Dokumenten, vergleichbar mit einer Tabelle, enthält mehrere Datensätze.
- **Dokument:** Ein einzelner Datensatz innerhalb einer Collection, bestehend aus Schlüssel-Wert-Paaren. Dokumente sind schemalos und können beliebige Felder sowie verschachtelte Objekte enthalten.
- **Unter-Collections:** Dokumente können weitere Collections als Unterstruktur enthalten, was komplexe Datenhierarchien erlaubt.

Diese Struktur ermöglicht eine flexible und skalierbare Speicherung von Daten mit variabler Struktur, ideal für moderne Anwendungen mit dynamischen Datenanforderungen.

Datenbank im System

Im vorliegenden System wird eine Collection namens `Abschüsse` verwendet, um erfasste Jagddaten zu speichern. Jedes Dokument repräsentiert einen einzelnen Abschuss mit folgenden Attributen:

- **ID (Integer):** Eine global eindeutige, fortlaufende Identifikationsnummer zur Referenzierung des Abschusses.
- **Datum (Timestamp):** Zeitpunkt des Abschusses.
- **Fallwild (Boolean):** Kennzeichnung, ob es sich um Fallwild handelt.
- **WildArt (String):** Art des Wildes, z. B. „Reh“, „Hirsch“.
- **Gewicht (Float):** Gewicht des erlegten Wildes in Kilogramm.
- **Alter (Integer):** Geschätztes Alter des Wildes in Jahren.
- **Beschreibung (String):** Optionales Feld für Beobachtungen oder Auffälligkeiten.
- **Schütze (String):** E-Mail-Adresse des Benutzers, der den Abschuss erfasst hat.
- **AbschussOrt (String):** Beschreibung des Fund- oder Abschussortes.
- **DatumUntersuchung (Timestamp):** Datum einer durchgeführten Untersuchung.
- **IstBeschaut (Boolean):** Kennzeichnung, ob der Abschuss bereits beschaut wurde.
- **IstJada (Boolean):** Kennzeichnung, ob der Abschuss bereits an die Bezirkshauptmanschaft gemeldet wurde

Weiters wird noch die Collecion `User` verwendet,in der alle Informationen zu den Benutzern gespeichert werden:

- **Name (String):** Name des Users.
- **Email (String):** Email des Users.
- **Role (String):** Rolle des Users.

Die ID wird über ein separates Dokument verwaltet, das als Zähler fungiert. Vor dem Anlegen eines neuen Abschusses wird der aktuelle Zählerstand ausgelesen, inkrementiert und dem neuen Datensatz als eindeutige ID zugewiesen. Anschließend wird der Zähler aktualisiert, um Duplikate zu vermeiden.

Diese Datenstruktur erlaubt eine effiziente Speicherung und Abfrage der Jagddaten, wobei Firestore die Echtzeit-Synchronisation und skalierbare Datenverwaltung unterstützt.

Verwendungsgrund
Für die Umsetzung des Systems fiel die Wahl auf Firebase Firestore aus mehreren technischen und praktischen Gründen:

- **Echtzeit-Daten-Synchronisation:** Firestore bietet eine native Unterstützung für Echtzeit-Updates, wodurch Änderungen an den Daten sofort an alle verbundenen Clients übertragen werden. Dies ist besonders vorteilhaft für Anwendungen mit mehreren Nutzern, die gleichzeitig Daten einsehen oder bearbeiten.
- **Serverloses Backend:** Firebase ermöglicht ein serverloses Datenmanagement, wodurch der Entwicklungsaufwand für Infrastruktur und Wartung erheblich reduziert wird. Es entfällt die Notwendigkeit, eigene Server zu betreiben oder zu skalieren.
- **Benutzer-Authentifizierung und Sicherheit:** Firebase Auth ermöglicht eine einfache und sichere Benutzerverwaltung, inklusive Authentifizierung per E-Mail, was im System zur Steuerung von Zugriffsrechten genutzt wird.

Im Vergleich zu relationalen Datenbanken oder anderen NoSQL-Systemen wie MongoDB oder Couchbase bietet Firebase eine Kombination aus Echtzeitfähigkeiten, einfacher Integration und Wartungsfreiheit, die für die Anforderungen dieses Systems besonders geeignet ist.

8.2 Backend Entwicklung

8.2.1 Implementierung des Datenbankzugriffs mit Firebase

Für die Anbindung der App an Firebase wird das offizielle Firebase JavaScript SDK verwendet. Zunächst erfolgt die Initialisierung der Firebase-App mit einer projektspezifischen Konfiguration, die alle notwendigen Zugangsdaten wie API-Schlüssel, Projekt-ID und weitere Parameter enthält, welche nach der Erstellung der Datenbank automatisch zur Verfügung gestellt wird. Dies geschieht durch den Aufruf von `initializeApp(firebaseConfig)`. Dieser Schritt stellt die Verbindung zwischen der React Native App und dem Firebase-Projekt her.
Anschließend werden die einzelnen Firebase-Dienste initialisiert, die im System genutzt werden:

- `getAuth(FIREBASE_APP)`: Initialisiert den Authentifizierungsdienst, der für die Nutzeranmeldung und Zugriffskontrolle zuständig ist.
- `getFirestore(FIREBASE_APP)`: Initialisiert die Firestore-Datenbank, welche die Speicherung und Abfrage der Jagddaten ermöglicht.

Durch diese Struktur können innerhalb der App sowohl Benutzer sicher authentifiziert als auch Daten in der Cloud-Datenbank effizient verwaltet werden. Der Zugriff auf die Daten erfolgt über die von Firestore bereitgestellten Funktionen, die es ermöglichen, Dokumente und Collections

zu lesen, zu schreiben oder zu aktualisieren.
Insgesamt bildet die Kombination aus `initializeApp()`, `getAuth()` und `getFirestore()` die technische Basis, auf der die App aufbaut, um sowohl Nutzerverwaltung als auch Datenhaltung sicher und skalierbar zu realisieren.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import{getAuth} from "firebase/auth";
import{getFirestore} from 'firebase/firestore'

import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries


// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyCSIfOe-KqDrmk0vqzGo9l-UkvzPmpey18",
  authDomain: "jagdeizendorf.firebaseapp.com",
  projectId: "jagdeizendorf",
  storageBucket: "jagdeizendorf.firebasestorage.app",
  messagingSenderId: "311241136839",
  appId: "1:311241136839:web:262a40340b234457243ce2",
  measurementId: "G-SQXT9VJR49"
};

// Initialize Firebase
export const FIREBASE_APP = initializeApp(firebaseConfig);
export const FIREBASE_AUTH = getAuth(FIREBASE_APP);
export const FIREBASE_DB = getFirestore(FIREBASE_APP);
```

Abbildung 8.1: Datenbankanbindung

8.3 Frontend Entwicklung

8.3.1 Login

Die Benutzeranmeldung erfolgt in der entwickelten App über Firebase. Hierbei wird den Nutzer:innen ermöglicht, sich mit einer zuvor registrierten E-Mail-Adresse und einem Passwort in die App einzuloggen.

Beim Start des Anmeldevorgangs werden die Eingaben der Benutzer:innen (E-Mail und Passwort) über Eingabefelder erfasst und an die Funktion signInWithEmailAndPassword() übergeben.

Diese Funktion ist Teil der Firebase-Bibliothek und prüft die angegebenen Daten gegen die in Firebase hinterlegten Benutzerkonten. Erfolgt die Anmeldung erfolgreich, wird der:die Benutzer:in automatisch zur Hauptansicht der App weitergeleitet. Sollte die Anmeldung fehlschlagen

```
const signIn = async () => {
  setLoading(true);
  try {
    const userCredential = await signInWithEmailAndPassword(auth, email, password);
    console.log('Benutzer angemeldet:', userCredential);

    // Nach erfolgreichem Login weiterleiten
    navigation.navigate('AppTabs');  //weiterleitung zum HomeScreen
  } catch (error: any) {
    console.error('Sign-in error: ', error);
    Alert.alert('Login fehlgeschlagen', error.message);
  } finally {
    setLoading(false);
  }
};
```

Abbildung 8.2: Loginfunktion

(z. B. durch ein falsches Passwort oder eine nicht vorhandene E-Mail-Adresse), wird eine entsprechende Fehlermeldung angezeigt.

Während des Anmeldevorgangs wird eine Ladeanimation (Activity Indicator) angezeigt, um visuelles Feedback über den laufenden Prozess zu geben. Dies verbessert die Benutzerfreundlichkeit und signalisiert, dass die Anwendung gerade arbeitet.

Insgesamt bietet Firebase eine einfache, sichere und skalierbare Möglichkeit zur Benutzerverwaltung, wodurch die Entwicklung eigener Authentifizierungsmechanismen vermieden werden kann.

8.3.2 Abschuss Übersicht

Rollenbestimmung des Benutzers

Beim Laden der Komponente wird mittels der Firebase-Auth-Funktion `onAuthStateChanged` der aktuell angemeldete Benutzer ermittelt. Anschließend wird in der Firestore-Collection `User` nach der hinterlegten Rolle gesucht:

```
const userQuery = query(
  collection(FIREBASE_DB, 'User'),
  where('email', '==', email)
);
```

Die ermittelte Rolle (*role*) wird im State gespeichert und steuert die nachfolgenden Datenabfragen.

Laden der Abschussdaten

Abhängig von der Benutzerrolle werden unterschiedliche Firestore-Abfragen ausgeführt:

- **Jagdleiter:** Abruf aller Abschussdatensätze, sortiert nach ID.
- **Kundig:** Kombination zweier Abfragen: Abschüsse mit `istJada = false` (Attribut, ob ein Abschuss bereits an die Bezirkshauptmanschaft gemeldet wurde) sowie Abschüsse, die der eigenen Benutzer-E-Mail zugeordnet sind.

- **Jäger:** Abruf aller eigenen Abschüsse, gefiltert nach `Schütze = Benutzer-E-Mail`.

Für den *Kundig*-Fall werden die Ergebnisse beider Abfragen mittels einer `Map`-Struktur zusammengeführt, um Duplikate zu vermeiden.

Zusätzliche Datenanerweiterung

Jeder Abschussdatensatz wird um den vollständigen Namen des Schützen ergänzt. Hierzu erfolgt eine weitere Firestore-Abfrage auf die `User`-Collection anhand der im Abschussdatensatz gespeicherten Schützen-E-Mail:

```
const userSnapshot = await getDocs(
  query(
    collection(FIREBASE_DB, 'User'),
    where('email', '==', abschuss.Schütze.toLowerCase())
  )
);
```

Darstellung der Daten

Die Daten werden in einer horizontal scrollbareren Tabelle angezeigt, bestehend aus Kopfzeile und Zeilen für jeden Datensatz. `FlatList` übernimmt die performante Darstellung großer Datenmengen, während `TouchableOpacity` für interaktive Zeilen sorgt, die je nach Rolle (Jagsleiter, Kundige Person) editierbar sind.

8.3.3 Neuen Abschuss melden

1. Datums- und Zeitverwaltung

Die Eingabe der Abschussdaten umfasst zwei Datumsfelder:

1. **Datum des Abschusses**
2. **Datum der Untersuchung**

Beide Felder werden mit dem `DateTimePicker` realisiert und im State gespeichert. Standardmäßig wird das aktuelle Datum vorbelegt.

2. Erfassung der Abschussdetails

Über verschiedene Formularelemente werden die relevanten Daten erfasst:

- **Fallwild** (Boolean über `Switch`).
- **Wildart** (Auswahl aus vordefinierten Optionen im `Picker`).
- **Abschussort** (Auswahl im `Picker`).
- **Gewicht** (Zahleneingabe in Kilogramm).
- **Alter** (Zahleneingabe in Jahren).
- **Beschreibung** (Freitextfeld für Auffälligkeiten).
- **Abnehmer**, **ID-Bescheinigung**, **ID-Kundig** (optionale Textfelder).

3. Generierung einer eindeutigen Abschuss-ID

Die Komponente greift auf ein spezielles Dokument `AbschussIDCounter/counter` in Firestore zu, um die aktuelle Zähler-ID zu lesen und um +1 zu erhöhen:

```
const idRef = doc(FIREBASE_DB, 'AbschussIDCounter', 'counter');
const idDoc = await getDoc(idRef);

let newId = 1;
if (idDoc.exists()) {
  const data = idDoc.data();
  newId = (data?.counter || 0) + 1;
}
```

Abbildung 8.3: Erhöhung des globalen Counters

Nach dem Speichern des neuen Datensatzes wird der Zähler mit `updateDoc` aktualisiert.

4. Speichern der Daten in Firestore

Die gesammelten Eingabedaten werden in der Collection `Abschüsse` gespeichert. Dazu wird `addDoc` verwendet, wobei Datumswerte als `Timestamp.fromDate` konvertiert werden. Das Feld `Schütze` wird automatisch mit der E-Mail-Adresse des aktuell angemeldeten Benutzers gefüllt.

8.3.4 Admin Ansicht

Laden der Daten

Beim Laden der Ansicht wird die Funktion `fetchData()` ausgeführt. Diese ruft alle Dokumente aus der Collection Abschüsse in Firestore ab und speichert sie im State der Komponente.

```
const snapshot = await getDocs(collection(FIREBASE_DB, 'Abschüsse'));
const data = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
setAbschuesse(data);
berechneStatistik(data);
```

Abbildung 8.4: Datenauslesung

Parallel wird direkt die statistische Auswertung erzeugt.

Statistische Auswertung

Die Funktion `berechneStatistik()` fasst die geladenen Daten nach **Jagdgebiet** und **Wildart** zusammen und zählt die jeweiligen Vorkommen. Das Ergebnis wird in einer verschachtelten Objektstruktur gespeichert:

$$\text{Ort} \rightarrow \text{Tierart} \rightarrow \text{Anzahl}$$

Diese Datenstruktur dient sowohl der **Visualisierung innerhalb der App** als auch der **Integration in den Excel-Export**.

Filtermöglichkeiten
Der Benutzer kann die angezeigten Datensätze über mehrere Parameter einschränken:

- **Jagdgebiet** (Auswahlliste)
- **Zeitraum** (Start- und Enddatum)
- **Status "istJada"** (Schalter für bereits in einem anderen System erfasste Abschüsse)

Die Filterlogik prüft nacheinander alle Kriterien und schließt nicht zutreffende Datensätze aus:

```
const filteredAbschuesse = abschuesse.filter(item => {
  if (filterOrt !== 'Alle' && item.AbschussOrt !== filterOrt) {
    return false;
  }
  if (istJadaFilter && item.istJada !== true) {
    return false;
  }
```

Abbildung 8.5: Filtermöglichkeiten der Admin Ansicht

Export in Excel
Über den Button *„Daten exportieren & senden"* werden alle aktuell gefilterten Datensätze in ein Excel-Dokument exportiert. Hierzu wird die Bibliothek `xlsx` eingesetzt, um aus JavaScript-Objekten Arbeitsblätter zu generieren. Das resultierende Dokument enthält zwei Tabellenblätter:

1. **Abschüsse** – die gefilterten Rohdaten
2. **Statistik** – aggregierte Werte pro Jagdgebiet und Tierart

```
const worksheet = XLSX.utils.json_to_sheet(filteredData);
const statistikSheet = XLSX.utils.json_to_sheet(statistikArray);

const workbook = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(workbook, worksheet, 'Abschüsse');
XLSX.utils.book_append_sheet(workbook, statistikSheet, 'Statistik');
```

Abbildung 8.6: Exportfunktion via Excel

Anschließend wird die Datei lokal gespeichert und automatisch per E-Mail mit Anhang versendet.

Teilen der Datei
Zusätzlich steht eine Funktion zum Teilen der erstellten Datei zur Verfügung. Diese nutzt die Expo-Schnittstelle `expo-sharing`, um das Dokument über andere installierte Anwendungen wie WhatsApp, AirDrop oder Cloud-Speicher zu verteilen.

8.4 Testing

Auf Grund der Verwendung von Expo Go konnte die Applikation direkt auf einem Smartphone getestet werden. Durch die echtzeitsynchronisation konnten änderungen sofort gesehen werden und somit einerseits auf Funktionalität als auch User-Freundlichkeit getestet werden.

Während der Entwicklung wurden in den regelmäßigen Statusmeetings mit dem Auftraggeber die Fortschritte direkt auf dem Smartphone getestet, dadurch konnten gewünschte Änderungen besser formuliert werden.

Die Applikation wurde in einer Jagdsitzung bereits vorgestellt und anschließend unter den Mitgliedern verteilt, dementsprechend wird derzeit eine Testphase durchgeführt und gewünschte Änderungen anschließend weitergegeben.

8.5 Deployment

Abbildung 8.7: QR Code zum App Download

Für das Deployment wurde auch das Framework Expo verwendet, welches eine einfache und effiziente Veröffentlichung mobiler Apps über die Cloud ermöglicht. Nach der lokalen Entwicklung und erfolgreichen Tests der App erfolgt das Deployment durch den Befehl expo publish, welcher die Anwendung auf den Expo-Servern veröffentlicht. Dabei wird ein öffentlicher Link generiert, der unkompliziert an Nutzer verteilt werden kann.

Um die Anwendung auch als eigenständige App lokal auf mobilen Geräten ausführen zu können, ist eine Installationsdatei notwendig. Für Android handelt es sich dabei um eine APK-Datei, für iOS um eine IPA-Datei. Diese werden mithilfe des Cloud-Dienstes Expo Application Services (EAS) Build erstellt, der den Build-Prozess übernimmt und somit eine lokale Entwicklungsumgebung überflüssig macht.
Der Build wird über die Kommandozeile mit dem Befehl `eas build --platform android` bzw. `eas build --platform ios` gestartet. Nach erfolgreichem Abschluss stellt Expo einen Download-Link zur Verfügung, über den die entsprechende Installationsdatei heruntergeladen, auf das Gerät übertragen und installiert werden kann. Dies ermöglicht die Ausführung der App als eigenständige Anwendung, unabhängig von der Expo Go App. Für die iOS-Installation ist jedoch ein kostenpflichtiger Apple Developer Account erforderlich, da Apple strenge Vorgaben für App-Signierungen auf realen Geräten vorschreibt.

Die entwickelte Applikation wurde grundsätzlich betriebssystemunabhängig programmiert und ist somit für Android- und iOS-Geräte ausgelegt. Aufgrund der mit der Veröffentlichung für iOS verbundenen hohen Kosten, insbesondere durch die erforderlichen Apple Developer Gebühren, wurde mit dem Auftraggeber gesprochen und gemeinsam beschlossen, dass das initiale Deployment ausschließlich für Android-Geräte erfolgt. Hierzu wird jedoch bereits nach möglichen Lösungswegen gesucht, welche keinen bis niedrigen Kostenaufwand verursachen.
graphicx float

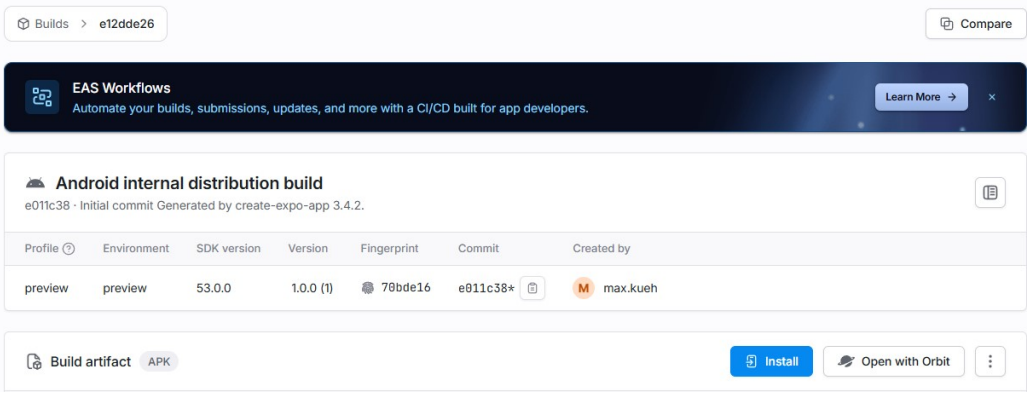


Abbildung 8.8: Ansicht dirkekt im Expo Account

9 Benutzung

9.1 Gedankenmodell

Der Jagdgenossenschaft war es wichtig, ein System zu entwickeln, welches einfach zu verwalten ist, jedoch gleichzeitig den zusätzlichen Aufwand von organisatorischen Aufgaben wie das händische Erstellen von Listen als auch Statistiken der Abschüsse reduziert.

Da jede Rolle der Genossenschaft (Jäger, Kundige Person, Jagdleiter) verschiedene Aufgaben als auch Pflichten hat, musste ein Benutzerrollensystem erstellt werden.

Mit dem Jagdleiter gemeinsam wurde für jede Rolle entschieden, welche Rechte vergeben werden, damit eine Datenkonsistenz herrscht und keine Verwirrung unter den Jagd-Mitgliedern aufkommt.

9.2 Programmablauf nach Benutzerrollen

Um eine klare Aufgabenverteilung, Datensicherheit und gezielte Zugriffskontrolle zu gewährleisten, wurde das System in drei Benutzerrollen unterteilt: Jäger, Kundige Person und Jagdleiter. Jede dieser Rollen verfügt über unterschiedliche Berechtigungen und Funktionalitäten innerhalb der Anwendung. Der konkrete Ablauf und die Nutzung des Programms unterscheiden sich daher je nach zugewiesener Rolle deutlich.

Jäger

Nach dem Login mit den persönlichen Zugangsdaten erhält der Jäger ausschließlich Zugriff auf seine eigenen Abschussdaten. In dieser Benutzeroberfläche kann er:

- neue Abschüsse erfassen,
- bestehende, von ihm erstellte Einträge einsehen,
- das eigene Passwort ändern.

Dadurch wird sichergestellt, dass nur der jeweilige Ersteller eines Abschusses über die Informationen verfügt und keine fremden Daten verändert werden können.

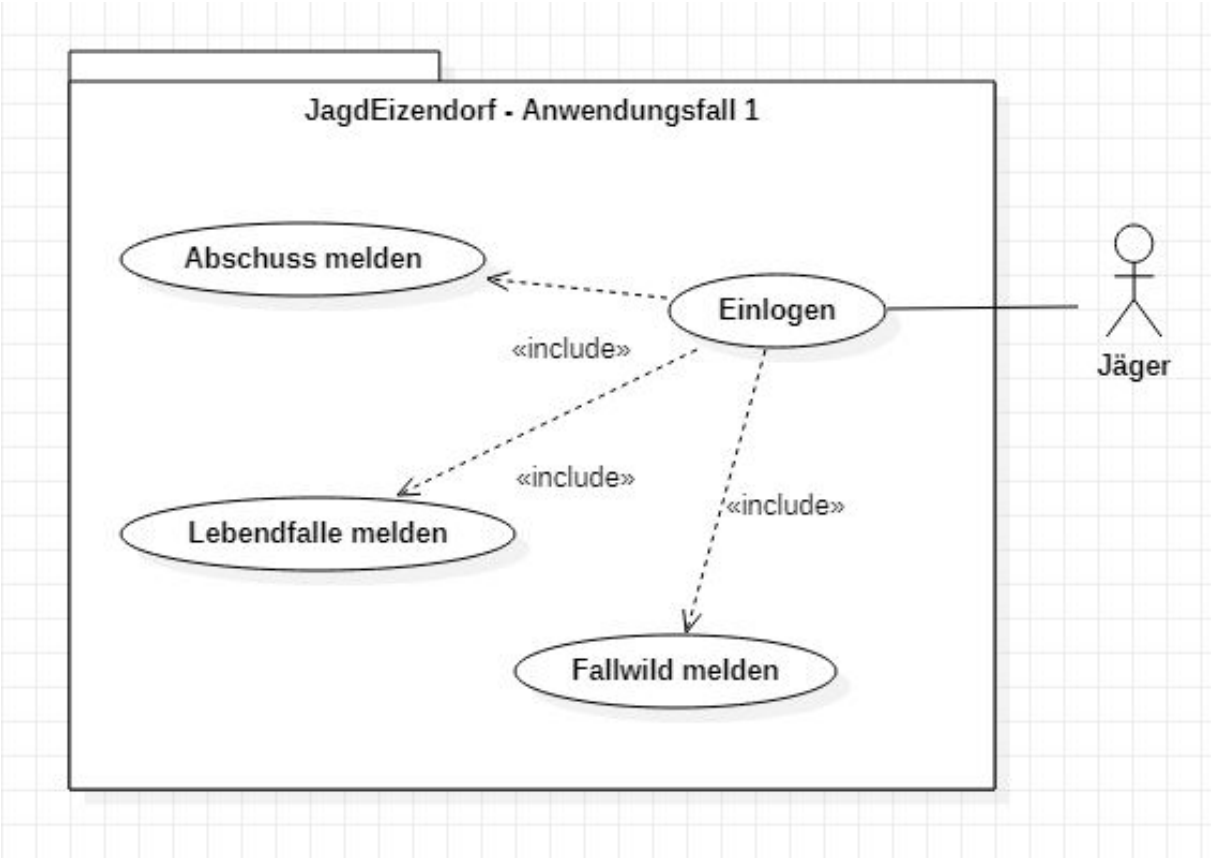


Abbildung 9.1: Use Case eines Jägers

Kundige Person

Die Kundige Person verfügt über erweiterte Rechte im System. Nach erfolgreicher Anmeldung kann sie zusätzlich zu den Rechten, die ein Jäger hat:

- alle im System erfassten Abschüsse einsehen,
- bestehende Datensätze bearbeiten (z. B. Ergänzung des Alters, Gewichts oder auffälliger Merkmale).

Diese Rolle unterstützt somit die Qualitätssicherung und stellt sicher, dass die Abschussdaten fachlich korrekt und vollständig dokumentiert sind.

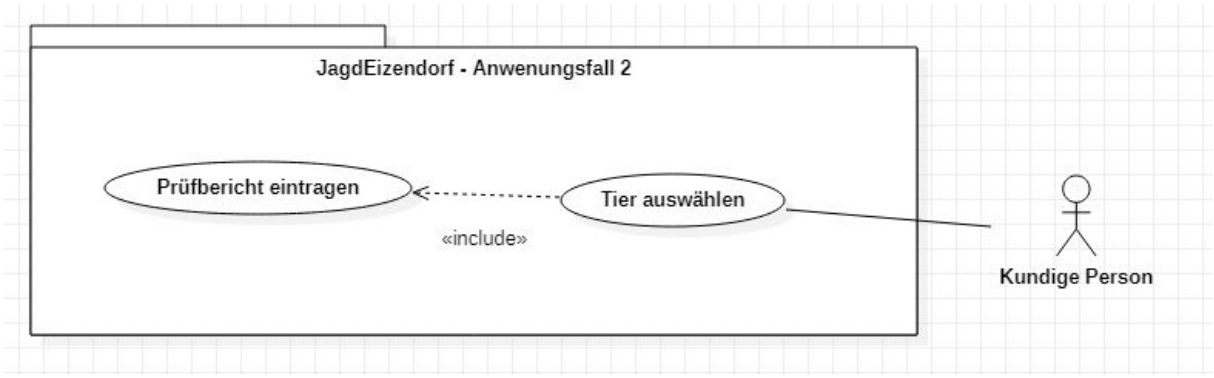


Abbildung 9.2: Use Case einer Kundigen Person

Jagdleiter

Der Jagdleiter hat Zugriff auf alle Funktionen des Systems. Zusätzlich zu den Rechten der Kundigen Person und des Jägers kann er:

- Abschusslisten als Excel exportieren,
- statistische Auswertungen zur Wildverteilung und Abschussverteilung aufrufen,
- die Benutzerverwaltung übernehmen – also neue Benutzer anlegen, bestehende Benutzer bearbeiten oder entfernen sowie Rollen zuweisen.

Durch diese erweiterten Funktionen übernimmt der Jagdleiter eine administrative Rolle und gewährleistet die Kontrolle und Pflege des Gesamtsystems.

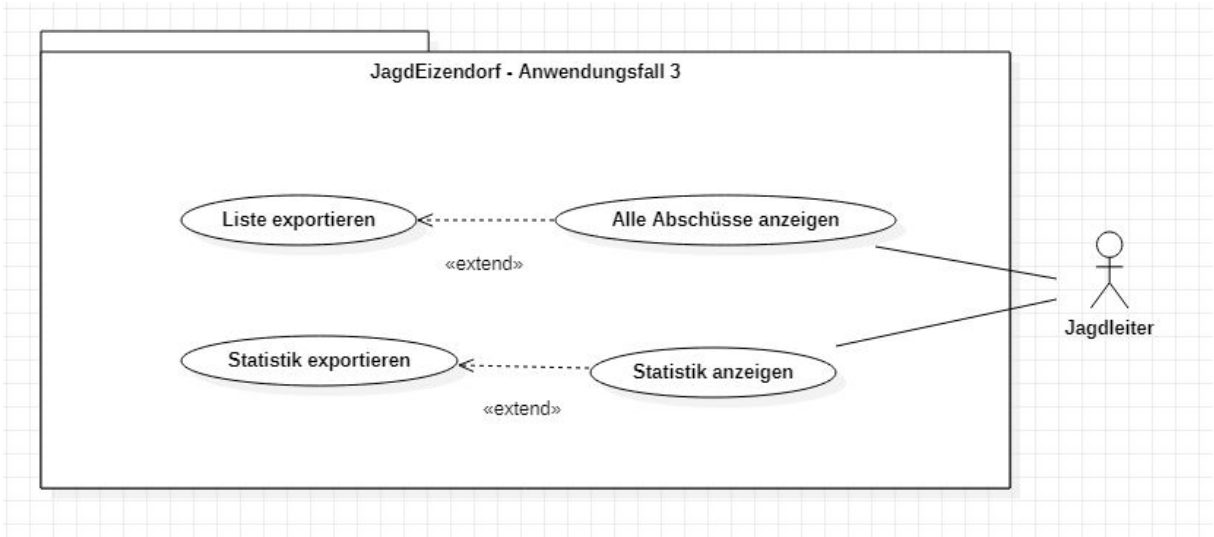


Abbildung 9.3: Use Case eines Jagdleiters

9.3 Ergebnisse

9.3.1 Login

Beim Starten der App öffnet sich als Erstes der Login-Screen. Hier meldet man sich mit E-Mail und Passwort an.

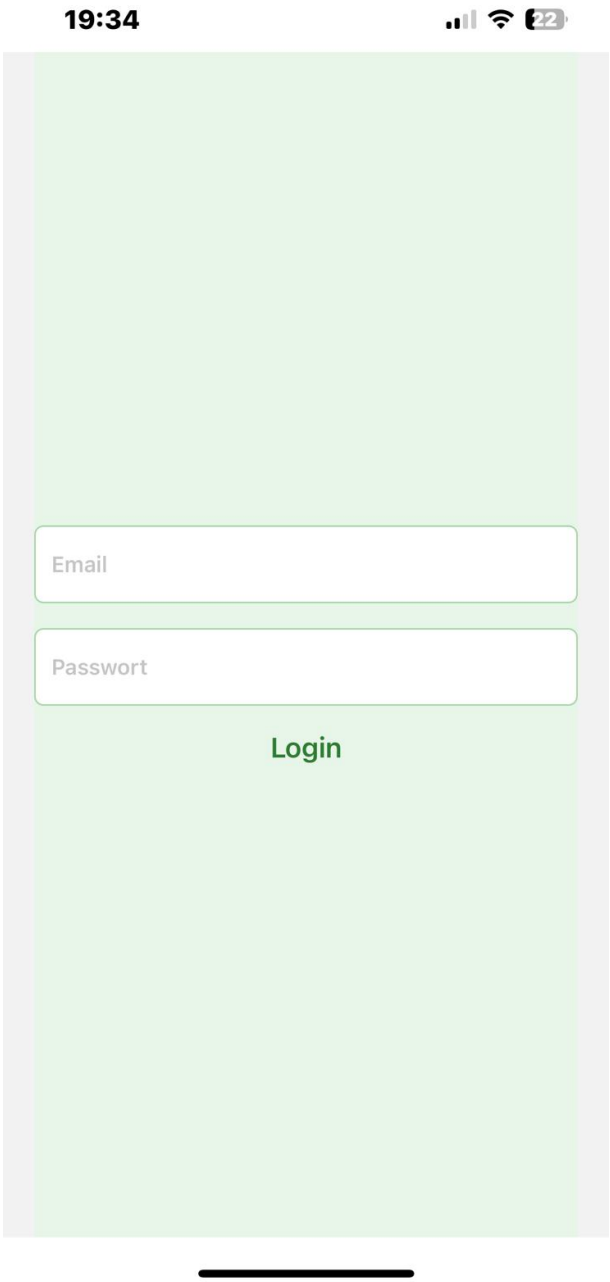


Abbildung 9.4: Login-Screen

9.3.2 Home-Screen

Im Home-Screen werden je nach Rolle entweder nur die eigenen Abschüsse (Jäger) oder alle Abschüsse (Kundige Person und Jagdleiter) angezeigt. Um einen Datensatz zu bearbeiten, muss dieser lediglich gedrückt werden. Dabei öffnet sich der Bearbeiten-Screen, wo fehlende Daten eingetragen oder bestehende korrigiert werden können.

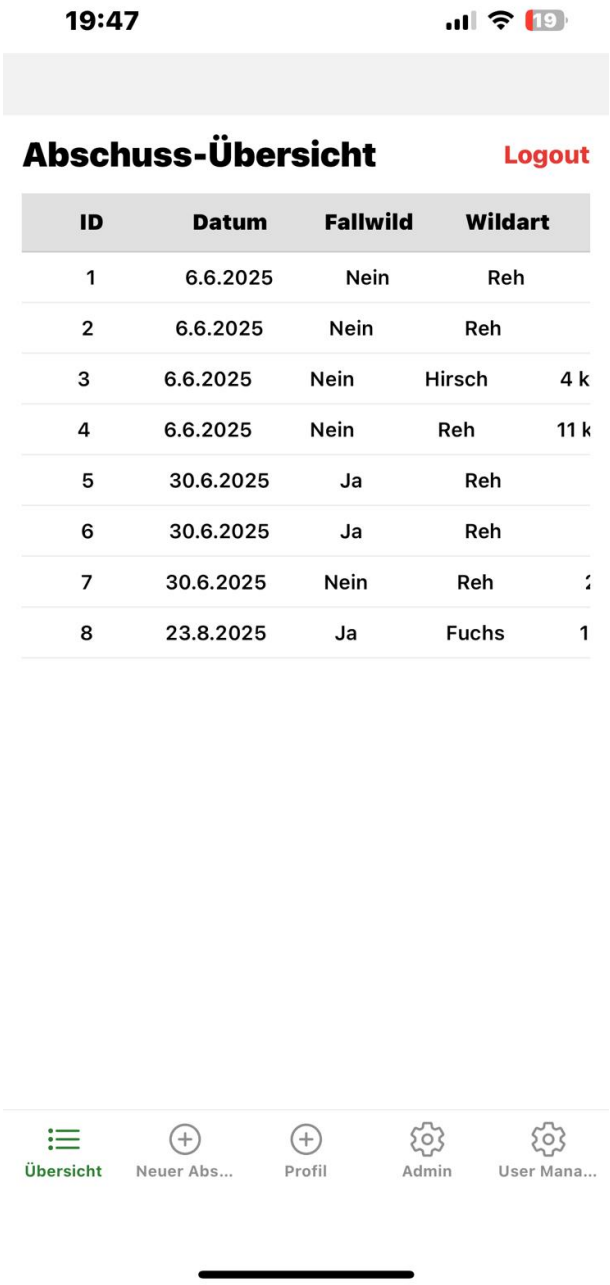


Abbildung 9.5: Startbildschirm

9.3.3 Neuer Abschuss

Um einen neuen Abschuss zu melden, muss in der Tab-Leiste „Neuer Abschuss“ ausgewählt werden. Dabei öffnet sich das Formular, in dem alle essentiellen Daten erfasst werden können. Mittels „Speichern“ wird der Datensatz in die Firebase-Datenbank geladen und ist somit bereits für Kundige Personen und Jagdleiter einsehbar.

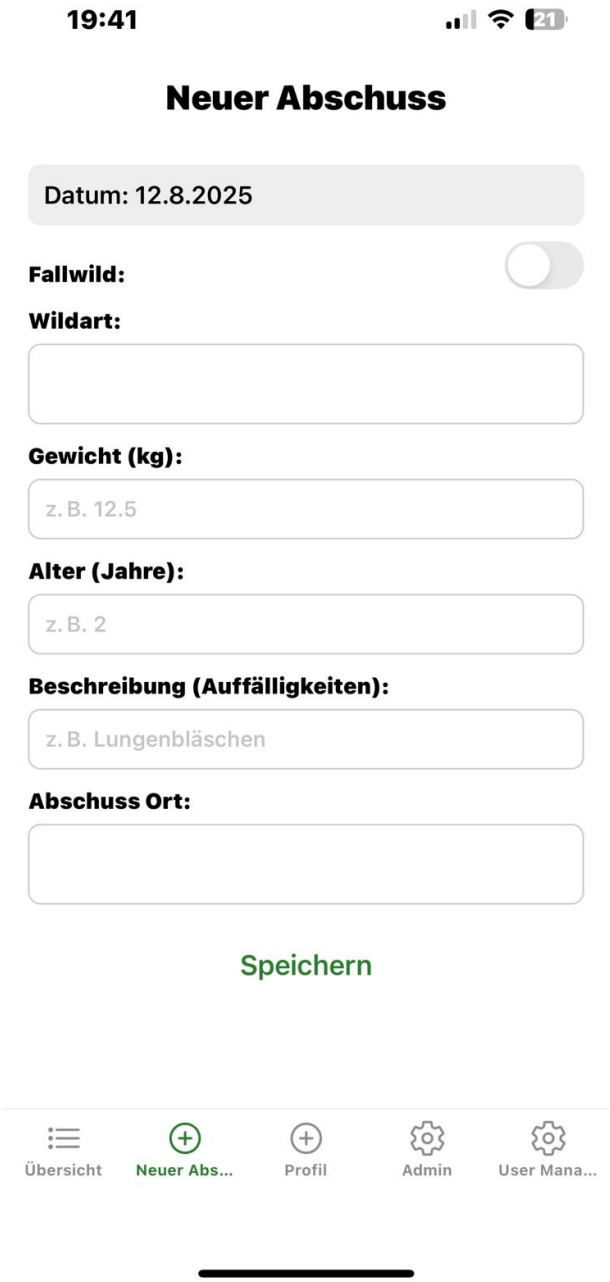


Abbildung 9.6: Erfassung eines neuen Abschusses

9.3.4 Admin-Screen

Der Jagdleiter verfügt zusätzlich über die Admin-Ansicht. Hier können einerseits Auswertungen zum jeweiligen Jagdgebiet angezeigt werden. Das Exportieren der Daten kann auf zwei Kriterien gefiltert werden: Durch Einschränkung des Zeitraums werden nur Datensätze, welche sich darin befinden, in eine Excel-Datei geladen. Dem Auftraggeber war es auch wichtig, dass die Daten auch danach gefiltert werden können, ob diese bereits an die Bezirkshauptmannschaft gemeldet wurden (hier handelt es sich um den Toggle „Bereits in JADA eingegeben“).

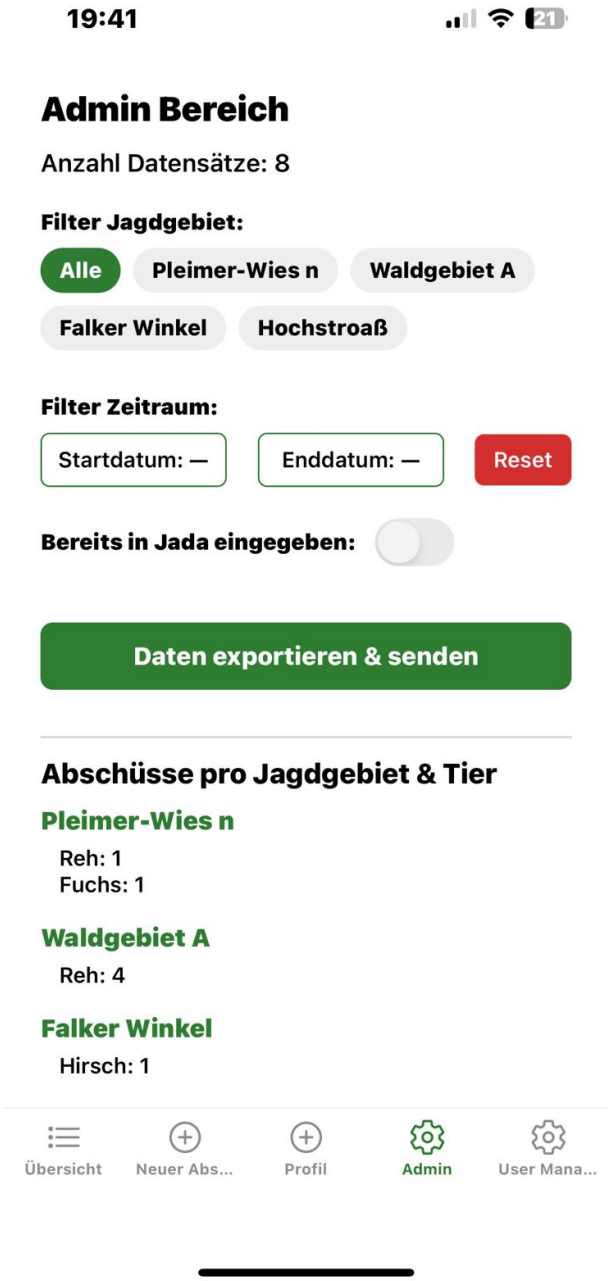


Abbildung 9.7: Admin-Übersicht

9.3.5 Benutzerverwaltung

Die Benutzerverwaltung wird ebenfalls vom Jagdleiter übernommen. Hier können Nutzer gelöscht, bearbeitet oder hinzugefügt werden. Beim Hinzufügen werden die Rolle, E-Mail sowie ein einmaliges Passwort vergeben. Das Passwort muss anschließend vom Benutzer selbst geändert werden.

Abbildung 9.8: Benutzerverwaltung

10 Projektumfeld

10.1 Max Kühberger

Derzeit bin ich als HR-Specialist im Bereich SAP HCM tätig, wo ich mich auf die Optimierung von Personalprozessen und die Implementierung von HR-Systemlösungen spezialisiere.

Auf das Thema der nachhaltigen Jagdabschussverwaltung bin ich durch meinen Bekanntenkreis aufmerksam geworden, der im jagdlichen Umfeld aktiv ist. Dies hat mein Interesse an der Verbindung von jagdlicher Praxis und moderner Softwareentwicklung geweckt.

Im Rahmen dieser Arbeit habe ich die Konzeption und Umsetzung einer plattformübergreifenden React Native App übernommen, die eine effiziente und transparente Verwaltung von Abschussdaten ermöglicht. Dabei konnte ich meine Kenntnisse in mobiler Entwicklung sowie in der Nutzung von Cloud-Diensten wie Firebase vertiefen.

Für Rückfragen oder weiterführende Informationen stehe ich gerne zur Verfügung:

E-Mail: max.kuehberger@gmail.com
Telefon: +43 6802320187

11 Ausblick in die Zukunft

Obwohl die App derzeit im Rahmen der Jagdgenossenschaft Eizendorf eingesetzt wird, ist sie technisch so konzipiert, dass sie nicht an diese spezifische Jagd gebunden ist. Die Struktur und Funktionalität der Anwendung wurden bewusst allgemein gehalten, um eine flexible Anpassung an unterschiedliche jagdliche Gegebenheiten zu ermöglichen.

Daher kann die App grundsätzlich auch in anderen Jagdgebieten eingesetzt werden. Durch vielfältige Konfigurationsmöglichkeiten – wie die Anpassung von Benutzerrechten, Formularinhalten oder organisatorischen Strukturen – lässt sich die Anwendung ohne großen Entwicklungsaufwand an die jeweiligen Anforderungen vor Ort anpassen. So bietet sie eine solide Grundlage für den Einsatz in weiteren Jagdgenossenschaften oder vergleichbaren jagdlichen Organisationen.

Die Verwendung von Firebase als Backend-Technologie ermöglicht zudem eine einfache und zentrale Datenverwaltung. Da alle Daten cloudbasiert gespeichert werden, reicht ein einziges Google-Konto aus, um die gesamte Datenbank zu betreiben. Dadurch entfällt die Notwendigkeit lokaler Serverinfrastruktur, was die Einrichtung und Wartung erheblich erleichtert.

Die Weiterentwicklung und Betreuung der App werde ich auch künftig übernehmen, um neue Funktionen bedarfsgerecht umzusetzen und den langfristigen Betrieb sicherzustellen.Verstärkt wird noch die Installation auf IOS-Geräten analysiert, damit JagdEizendorf auf allen Geräten vollständig genutzt werden kann. Zusammengefasst kann die App somit zu einer überregional einsetzbaren Lösung für die digitale Abschussplanung und Wildtierdokumentation werden.

12 Resümee

Im Zuge meiner Arbeit konnte ich wertvolle Erfahrungen im gesamten Entwicklungsprozess sammeln – von der ersten Idee über die Konzeption bis hin zur technischen Umsetzung und Dokumentation. Dabei stand für mich im Vordergrund, vorhandenes Wissen praktisch einzusetzen, kontinuierlich zu erweitern und den Umgang mit neuen Technologien zu vertiefen.

Besonders lehrreich war der Umgang mit Veränderungen und Anpassungen, die während der Entwicklung notwendig wurden. Diese habe ich nicht als Hindernis, sondern als Chance gesehen, flexibel auf neue Anforderungen zu reagieren und dadurch sowohl meine technischen als auch meine organisatorischen Fähigkeiten weiter auszubauen.

Neben der technischen Arbeit habe ich gelernt, meine Zeit eigenverantwortlich zu strukturieren und die einzelnen Arbeitsschritte sinnvoll zu planen. Auch die Bedeutung klarer Kommunikation und die Bereitschaft, Arbeitsweisen zu hinterfragen und anzupassen, wurden mir im Verlauf der Arbeit bewusst.
Darüber hinaus fand ich es besonders spannend, das Thema Jagd kennenzulernen und einen Einblick in die damit verbundenen Abläufe gewinnen zu können.

Literatur

[1] Meta Platforms, Inc. (o. J.). *React Native – Create native apps for Android and iOS using React*. Abgerufen von `https://reactnative.dev/`

[2] Expo. (o. J.). *Expo Documentation*. Abgerufen von `https://docs.expo.dev/`

[3] Sharma, A. (2022). *React Native performance optimization techniques*. Abgerufen von `https://www.simform.com/blog/react-native-performance/`

[4] Google LLC. (o. J.). *Firebase – Build apps fast, without managing infrastructure*. Abgerufen von `https://firebase.google.com/`

[5] Firebase Documentation. (o. J.). *Firebase Developer Documentation*. Abgerufen von `https://firebase.google.com/docs`

[6] Google Developers. (o. J.). *Chrome DevTools*. Abgerufen von `https://developer.chrome.com/docs/devtools/`

[7] Expo. (o. J.). *Expo Go – Run your React Native projects instantly*. Abgerufen von `https://expo.dev/client`

13 Bildquellen

Google Firebase Logo: https://firebase.google.com/static/images/brand-guidelines/logo-monochrome.png

Expo Go Logo: https://images.seeklogo.com/logo-png/45/2/expo-go-app-logo-png$_seeklogo-457073.png$

$React-NativeLogo : https : //1000logos.net/wp-content/uploads/2023/10/React-Logo.png$

Abbildungsverzeichnis