

# **Diplomarbeit**

Inventurscanner der Freiwilligen Feuerwehr Hohenkogl

**Lukas Adelmann, Simon Flicker**

Betreuer:  
BEd Siegfried Schöberl  
Prof. DI Christoph Wurzinger

---

## **Abgabevermerk:**

Datum:

Unterschrift:



---

## 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.

Weiz, am 1. April 2022

Lukas Adelmann: .....

Simon Flicker: .....

---

## Kurzbeschreibung

### Vor dieser Diplomarbeit

Die Freiwillige Feuerwehr Hohenkogl verwendete zur Inventarisierung eine Access-Datenbank, welche das gesamte Inventar der Feuerwehr beinhaltet. Die Gegenstände wurden mit Etiketten versehen und per Hand mit der Inventarnummer beschriftet.

### Anforderungen

Mit dieser Diplomarbeit soll das bestehende Inventursystem der Freiwilligen Feuerwehr Hohenkogl überarbeitet werden. Ziel ist die Verbesserung der vorhandene Access Datenbank, die Daten sollen nach jeder Änderung im Cloudspeicher aktualisiert werden. Ein mobiles Endgerät soll es ermöglichen, die Daten aus der Cloud mit einem eingescannnten Strich- oder 2D-Code zu verknüpfen und gegenstandsspezifische Informationen anzeigen. Die Etiketten mit den Codes sollen mit einem eigens dafür vorgesehenen Drucker selbst erstellt werden. Erweitert wird auf dem mobilen Endgerät eine Funktion zur Inventur gefordert.

### Ziel

Diese Diplomarbeit dient dazu, die Arbeit mit dem Inventursystem für die Mitglieder der FF Hohenkogl zu erleichtern und zu beschleunigen. Die Funktion, gegenstandsspezifische Informationen anzuzeigen, soll für die einfache Identifizierung von Gegenständen sowie der Bestimmung des zugehörigen Raums und des Besitzers dienen. Die Inventur wird durch die Verwendung der Anwendung vereinfacht und beschleunigt. Weiters wird durch eine Funktion in der Datenbank die Durchführung wiederkehrender Überprüfungen erleichtert, indem durch Access Abfragen die anstehenden Prüfungen angezeigt werden.

---

## Abstract

### Current state

The starting point is an inventory system based on an existing Access database which contains the entire inventory. The items are getting labeled by hand with the inventory number.

### Requirements

The thesis is aimed to revise the current inventory system of the voluntary fire brigade Hohenkogl. When changes are made, all the data should automatically be stored in the cloud. A mobile device should allow a barcode or 2D code to be linked to existing data in the cloud. Information should be shown based on the scanned item. The label should be self produced with a printer designed for this purpose. As an extended function it should be possible to do a stocktake.

### Goals

The goal of this thesis is to simplify and advance the work with the inventory system for the members of the voluntary fire brigade Hohenkogl. The possibility to display item-specific information is intended for easy identification of items and determination of the associated room and owner. Furthermore, a function in the database facilitates the execution of recurring checks by displaying the pending checks through Access queries.

---

## Vorwort

Aufgrund von beidseitigem Interesse an der Feuerwehr und den Tätigkeiten von Simon Flicker in diesem Bereich, entschieden wir uns dafür, das Inventursystem der Freiwilligen Feuerwehr Hohenkogl, als Diplomarbeit zu überarbeiten und zu verbessern. Während sich Simon Flicker mit dem Drucken der Etiketten sowie der Verbesserung der Access-Datenbank auseinandersetzte, war Lukas Adelmann für die Entwicklung der Webanwendung zuständig.

An dieser Stelle möchten wir uns bei unseren Betreuern, BEd Siegfried Schöberl und Prof. DI Christoph Wurzinger bedanken. Unser Dank geht auch an Carmen Hirtenfellner und Hödl Manfred, welche uns von Seiten der FF Hohenkogl unterstützten.

## Inhaltsverzeichnis

<b>Eidesstattliche Erklärung</b>	i
<b>Kurzbeschreibung</b>	ii
<b>Abstract</b>	iii
<b>Vorwort</b>	iv
<b>1 Datenbank</b>	1
1.1 Grundlegender Aufbau . . . . .	1
1.2 Tabellen . . . . .	1
1.2.1 Gruppe . . . . .	1
1.2.2 Inventar . . . . .	1
1.2.3 Mitglieder . . . . .	3
1.2.4 Raum . . . . .	4
1.3 Abfragen . . . . .	4
1.3.1 Daten für Formulare und Berichte . . . . .	4
1.3.2 qryExport . . . . .	5
1.3.3 Anstehende Prüfungen und Ablaufdaten . . . . .	5
1.4 Formulare . . . . .	6
1.5 Berichte . . . . .	7
1.6 Export zu .js . . . . .	7
<b>2 Drucken der Etiketten</b>	10
2.1 Thermotransferdruck . . . . .	10
2.2 Drucker und Druckvorgang . . . . .	11
2.3 Design der Etiketten . . . . .	13
2.3.1 Codes . . . . .	15
2.4 Etiketten . . . . .	15
2.4.1 Etiketten Standard . . . . .	15
2.4.2 Patch-Etiketten . . . . .	16
2.5 Farbfolien . . . . .	17
<b>3 Mobiles Gerät</b>	18
<b>4 Datenübertragung zwischen Datenbank und Mobilgerät</b>	18
4.1 Erzeugung des Dropbox-Links . . . . .	18
<b>5 Anwendung</b>	19
5.1 Scanner . . . . .	19
5.1.1 Code-Scanner . . . . .	19
5.2 Datenverarbeitung . . . . .	21

5.3	Initialisieren . . . . .	22
5.3.1	Überprüfen . . . . .	22
5.4	Ausgabe . . . . .	24
5.5	Inventur . . . . .	26
5.5.1	Inventur Knopf . . . . .	26
5.5.2	Verarbeitung der gescannten Nummern . . . . .	27
5.5.3	Anzeige der Inventurdaten . . . . .	28
5.5.4	Speichern der laufenden Inventur . . . . .	29
5.5.5	Ausgabe als PDF . . . . .	30
5.6	Verwendung ohne Codes . . . . .	33
5.7	GitHub Pages . . . . .	33
5.8	Progressive Web App . . . . .	34
5.8.1	Progressive Web App erstellen . . . . .	34
<b>Zusammenfassung</b>		<b>37</b>
<b>Abbildungsverzeichnis</b>		<b>39</b>
<b>Literaturverzeichnis</b>		<b>41</b>
<b>Anhang</b>		<b>42</b>
<b>6</b>	<b>Leitfaden Datenbank</b>	<b>42</b>
6.1	Einleitung . . . . .	42
6.2	Eingabe von Daten . . . . .	42
6.2.1	Tabelle Gruppe und Mitglieder . . . . .	42
6.2.2	Tabelle Inventar . . . . .	42
<b>7</b>	<b>Anleitung Drucker</b>	<b>43</b>
7.1	Drucken von Etiketten . . . . .	43
7.2	Wechseln der Etiketten . . . . .	43
7.3	Wechseln der Farbrolle . . . . .	43
<b>8</b>	<b>Anleitung zum Inventurscanner der FF-Hohenkogl</b>	<b>44</b>
8.1	Erstes Starten der Anwendung . . . . .	44
8.2	Startbildschirm . . . . .	44
8.3	Manuelle Eingabe . . . . .	45
8.4	Scannen . . . . .	45
8.5	Inventur . . . . .	45
8.5.1	Ausgabe der Inventur . . . . .	46
8.5.2	Inventur Speichern und Laden . . . . .	46
8.5.3	Inventur als PDF ausgeben . . . . .	47
8.6	Ergebnis . . . . .	47
<b>Projektzeitennachweis Lukas Adelmann</b>		<b>48</b>



# 1 Datenbank

## 1.1 Grundlegender Aufbau

Die Inventardatenbank besteht bereits und enthält vier Tabellen (siehe Kapitel 1.2), zwölf Abfragen (siehe Kapitel 1.3), acht Formulare (siehe Kapitel 1.4) und elf Berichte (siehe Kapitel 1.5).

## 1.2 Tabellen

Die Daten sind in den vier Tabellen gespeichert: „tblGruppe“, „tblInventar“, „tblmitglieder“ und „tblRaum“.

Tabellen	
	tblGruppe
	tblInventar
	tblmitglieder
	tblRaum

Abbildung 1: Tabellen

### 1.2.1 Gruppe

Die Tabelle „tblGruppe“ beinhaltet die Gruppe/Kategorie, in welche die Gegenstände eingeordnet werden, und die dazugehörige grId. Insgesamt sind 15 Gruppen hinterlegt.

tblGruppe	
grId	grName
1	Alarm-, Fernmelde-, Brandmeldegeräte
2	Löschergeräte
3	Leitern u. Rettungs-, Sanitätsausrüstung
4	Feuerwehrbekleidung
5	Schutzausrüstung u. -bekleidung
6	Beleuchtungs- und Stromversorgung
7	Handwerkzeuge, Gurte, Seile, Ketten
8	Techische Geräte u. Zubehör
9	Materialien für Hochwasser und Straßensperren
10	Öleinsatzgeräte

Abbildung 2: Tabelle Gruppe

### 1.2.2 Inventar

Die Tabelle „tblInventar“ beinhaltet alle inventarisierten Gegenstände und sämtliche Informationen zu diesen, darunter auch Raumnummer, Mitgliedsnummer, Gruppennummer und Daten zu Prüfungen. Aktuell beinhaltet diese Tabelle 4294 Datensätze.

## Änderungen

An der Tabelle tblinventar wurden diverse Verbesserungen vorgenommen. Es wurden Redundanzen bei der Inventarnummer und der Prüfpflicht entfernt, indem diese berechnet wurden. Um zwei verschiedene Prüfintervalle darstellen zu können, wurden noch drei weitere Spalten hinzugefügt. Weiters wurde auch das Datum der nächsten Prüfung auf Basis der bereits vorhandenen Felder „invPrüfZeitraum“ und „invLetztePrüfung“ berechnet. Aus der „invVerwDauer“ und dem „invBaujahr“ wurde das Verweildatum berechnet.

invId	invName	invInvNummer	invRaum	invMitgliederId	invGruppe	invNurNummer	invBaujahr	invPrüPflichtig	invPrüfZeitraum
1305	Abschleppseil	8/13	1	0	8	13	1994 -		
1307	Arbeitshandschuhe Paar	4/10	5	0	4	10	1994 -		
1308	Arbeitshandschuhe Paar	4/11	5	0	4	11	2006 -		
1309	Arbeitshandschuhe Paar	4/12	5	0	4	12	2006 -		
1310	Arbeitshandschuhe Paar	4/13	5	0	4	13	2006 -		
1311	Arbeitshandschuhe Paar	4/14	5	0	4	14	1995 -		
1312	Arbeitshandschuhe Paar	4/15	5	0	4	15	1995 -		
1317	Ladegerät zu Tragkraftspritze	2/5	2	0	2	5	2008 -		
1325	Bolzenschneider	7/5	1	0	7	5	1992 -		
1326	Brecheisen	7/4	1	0	7	4	1992 -		

Abbildung 3: Tabelle Inventar Teil 1

invLetztePrüfung	invNächstePrüfung	inv2PrüfZeitraum	inv2LetztePrüfung	invNächstePrüfung	invVerwDauer	invAblaufDatum	invPreis
							27,6
							9,0
							9,0
							9,0
							9,0
							9,0
							92,9
							61,5

Abbildung 4: Tabelle Inventar Teil 2

Feldname	Felddatatype
invId	AutoWert
invName	Kurzer Text
invRaum	Zahl
invGruppe	Zahl
invPreis	Zahl
invBaujahr	Zahl
invVerwDauer	Zahl
invPrüfZeitraum	Zahl
invLetztePrüfung	Datum/Uhrzeit
invNurNummer	Zahl
invMitgliederId	Zahl
invInvNummer	Berechnet
inv2PrüfZeitraum	Zahl
inv2LetztePrüfung	Datum/Uhrzeit
invNächstePrüfung	Berechnet
inv2NächstePrüfung	Berechnet
invPrüfpflichtig	Berechnet
invAblaufDatum	Berechnet

Abbildung 5: Entwurfsansicht der Tabelle Inventar

### Berechnung Inventarnummer

$$[invGruppe] & "/" & [invNurNummer]$$

### Berechnung Nächste Prüfung

$$[invLetztePrüfung]+365*[invPrüfZeitraum]$$

### Berechnung Prüfpflichtig

$$\text{Wenn}(\text{IstNull}([invPrüfZeitraum]); "-"; "JA")$$

### Berechnung Ablaufdatum

$$[invBaujahr]+[invVerweilDauer]$$

### 1.2.3 Mitglieder

Die Tabelle „tblmitglieder“ beinhaltet Namen, Dienstgrad und Mitgliedsnummer aller Mitglieder der FF Hohenkogl. Aktuell sind 110 Mitglieder hinterlegt.

mitNr	mitVorname	mitNachname	mitDienstgrad
0 ---	---	---	---
1 Jürgen	Kubera	OLM	
2 Johann	Harrer	HFM	
3 Franz	Mußbacher	EHBI	
4 Herbert	Walcher	EOBI	
5 Tobias	Schlemmer	JFM	
6 Celina	Derler	JFM	
7 Florian	Unger	FM	
8 Erich	Mußbacher	HLM	
9 Gerald	Wiedenhofer	ABI-V	
10 Alois	Wagner	EBM	

Abbildung 6: Tabelle Mitglieder

### 1.2.4 Raum

Die Tabelle „tblRaum“ beinhaltet die Räume, in denen das Inventar gelagert ist, und die dazugehörige „RaumId“. Insgesamt sind 27 Räume hinterlegt.

raumId	raumName
1	KLFA Land Rover
2	LFBA Mercedes 814
3	MTFA Ford - Transit
4	TLFA 3000
5	Garage
6	Schulungsraum
7	Großlager
8	Kleinalager
9	Getränkekeller
10	Stiegenhaus

Abbildung 7: Tabelle Raum

### 1.3 Abfragen

Abfragen	
	qryAblaufDatum
	qryAblaufDatumErweitert
	qryAnstehende2Prüfungen
	qryAnstehende2PrüfungenErweitert
	qryAnstehendePrüfungen
	qryAnstehendePrüfungenErweitert
	qryBerichtnachGruppen
	qryBerichtnachGruppenMitglieder
	qryBerichtnachMitglieder
	qryBerichtnachRäume
	qryExport
	qryFormularDatenHerkunft

Abbildung 8: Abfragen

#### 1.3.1 Daten für Formulare und Berichte

Die fünf bereits bestehenden Abfragen „qryBerichtnachGruppen“, „qryBerichtnachGruppenMitglieder“, „qryBerichtnachMitglieder“, „qryBerichtnachRäume“ und „qryFormularDatenHerkunft“ stellen die Daten für die Formulare (siehe Kapitel 1.4) und Berichte (siehe Kapitel 1.5) bereit.

### 1.3.2 qryExport

In der neuen Abfrage „qryExport“ werden die Daten, welche später in der Web-Anwendung angezeigt werden, aus den Tabellen (siehe Kapitel 1.2) „tblGruppe“, „tblinventar“, „tblmitglieder“ und „tblmitglieder“ zusammengefügt.

invInvNummer	invName	raumName	mitNr	mitVorname	mitNachname	invPrüfpflichtig	invBaujahr
4/100	Einsatzstiefel	Persönlich	3	Franz	Mußbacher	-	2002
4/1002	Einsatzjacke grün	Persönlich	72	Elisabeth	Karner	-	2006
4/1004	Einsatzjacke grün	Persönlich	37	Johann	Schwarz	-	2006
4/1005	Einsatzjacke grün	Persönlich	18	Nikolina	Kriendlhofer	-	2006
4/1009	Einsatzmütze grün	Persönlich	95	Peter	Stevens	-	2006
4/101	Einsatzstiefel	Kleinlager	4	Herbert	Walcher	-	2002
4/1010	Einsatzmütze grün	Persönlich	87	Roman	Mußbacher	-	2006
4/1014	Einsatzstiefel	Kleinlager	93	Sebastian	Zöhrer	-	2011
4/1015	Einsatzstiefel	Persönlich	80	Franz	Unger	-	2006
4/1017	Dienstgürtel schwarz	Persönlich	73	Philipp	Mußbacher	-	2007

Abbildung 9: Abfrage Export

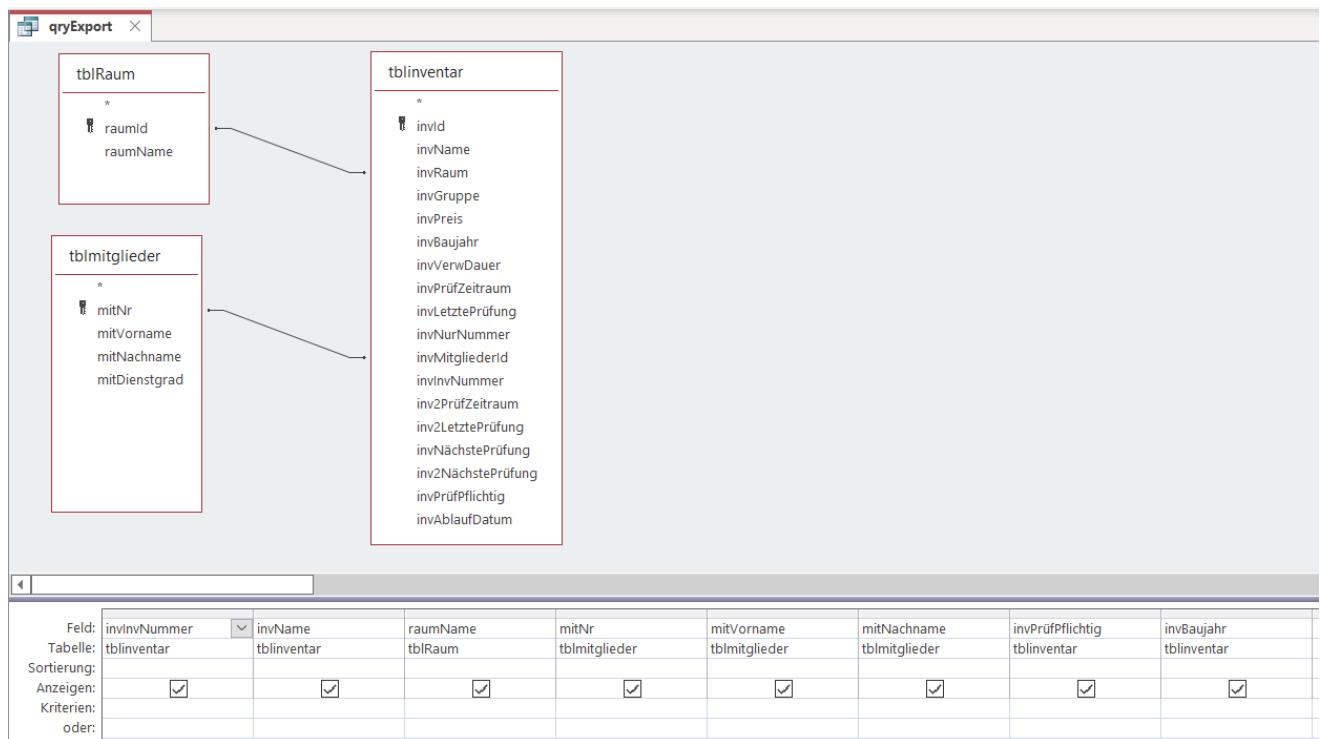


Abbildung 10: Entwurfsansicht der Abfrage Export

### 1.3.3 Anstehende Prüfungen und Ablaufdaten

Die Abfragen „qryAnstehendePrüfungen“, „qryAnstehende2Prüfungen“ und „qryVerweilDatum“ listen alle Datensätze auf, welche in einem halben Jahr bzw. einem

Jahr geprüft bzw. ausgetauscht werden müssen. Die Abfragen „qryAnstehendePrüfungenErweitert“, „qryAnstehende2PrüfungenErweitert“, „qryAblaufDatumErweitert“ beinhalten weitere Informationen über die zu prüfenden bzw. auszutauschenden Gegenstände.

Anmerkung: Da manche der bereits durchgeführte Prüfungen nicht in die Datenbank eingetragen wurden, werden mehr Gegenstände aufgelistet, als tatsächlich geprüft werden müssen.

invInvNummer	invName	raumName	invNächstePrüfung
8/46	Hydraulikstempel	Großlager	31.05.2013
8/47	Hydraulikstempel Verlängerung	Großlager	05.01.2016
11/2	Fahrzeug LFB-A Mercedes 814	LFBA Mercede	29.02.2016
11/14	Fahrzeug MTF Ford - Transit	MTFA Ford - Ti	30.04.2016
11/11	Fahrzeug KLF-A Landrover	KLFA Land Rov	30.06.2016
8/71	Anbauseilwinde WARN 5,4t	TLFA 3000	11.01.2020
2/8	Handfeuerlöscher S9	TLFA 3000	03.05.2021
2/191	Handfeuerlöscher CO2 5kg	TLFA 3000	03.05.2021
2/185	Handfeuerlöscher Trocken G2	TLFA 3000	03.05.2021
2/38	Handfeuerlöscher 6kg ABC Pulv	MTFA Ford - Ti	03.05.2021

Abbildung 11: Abfrage Anstehende Prüfungen

Feld:	invInvNummer	invName	raumName	invNächstePrüfung
Tabelle:	tblInventar	tblInventar	tblRaum	tblInventar
Sortierung:				Aufsteigend
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kriterien:				<(Datum() + 180)
oder:				

Abbildung 12: Abfrage Anstehende Prüfungen in der Entwurfsansicht

## 1.4 Formulare

Es bestehen bereits diverse Formulare, mit welchen die Datensätze nach diversen Kriterien gefiltert werden können. An diesen wurden nur Änderungen am Feld „invPrüfPflichtig“ durchgenommen. Da dieses in der Tabelle „tblInventar“ verändert wurde, mussten die Formulare ebenfalls angepasst werden. Das Standardformular „frmMain“ beinhaltet neben Links zu anderen Formularen auch die Schaltfläche, mit welcher die Daten der Abfrage „qryExport“ als js-Datei exportiert werden (siehe Kapitel 1.6). Zudem wurden Unterformulare eingefügt, welche die anstehenden Prüfungen zur besseren Übersicht direkt beim Starten der Datenbank anzeigen.

### Formulare

-  frmGruppen
-  frmMain
-  frmMitglieder
-  frmMitglieder\_Administration
-  frmRäume
-  frmUfInventarGruppen
-  frmUfInventarMitglieder
-  frmUfInventarRäume

Abbildung 13: Formulare

## 1.5 Berichte

Die bereits bestehenden Berichte stellen den Inhalt der Formulare grafisch aufbereitet dar. An diesen wurden keine Änderungen vorgenommen.

### Berichte

-  berAblaufDatum
-  berAlleGruppen
-  berAnstehende2Prüfungen
-  berAnstehendePrüfungen
-  berEinzelGruppe
-  berEinzelGruppeMitglieder
-  berEinzelMitglieder
-  berEinzelRaum
-  berMitglieder
-  berMitgliederGesamt
-  berMitgliederGesamt\_1Seite\_pro\_Mitglied

Abbildung 14: Berichte

## 1.6 Export zu .js

Um die Daten aus der Access-Abfrage als JSON-String exportieren zu können, wird eine VBA-Funktion angewandt. Diese wird sowohl beim Klicken auf die Schaltfläche "Export" als auch beim Schließen des Formulars in der Access-Datenbank ausgeführt. Die Funktion schreibt jeden einzelnen Datensatz in eine .js-Datei (siehe Kapitel 5.2), welche wiederum im Cloudspeicher Dropbox gespeichert wird. Nach jedem Exportvorgang wird die vorherige Version überschrieben.

Die Export-Funktion baut auf die stackoverflow-Antwort von Norm Simoneau [1] auf. Diese wurde um die UTF-8 Darstellung erweitert. Zudem wurde der Aufbau des

JSON-Strings an die Web-Anwendung angepasst Außerdem werden Anführungszeichen durch ' \\ '' ersetzt, da die normalen Anführungszeichen bereits in der Syntax des JSON-Strings verwendet werden, da die darauffolgenden Datensätze ansonsten nicht mehr berücksichtigt werden würden.

Da die Daten zunächst im ANSI-Code gespeichert wurden, konnten keine Sonderzeichen oder Umlaute dargestellt werden. Dies wurde durch die Verwendung eines Datenstroms (Data Stream) umgangen. Stream-Objekte können Bytedatenströme lesen, schreiben und verwalten [2]. In diesem Fall werden die Datensätze in den Datenstrom geschrieben, welcher in UTF-8 codiert ist, und später wieder in die Datei gespeichert.

```

'Funktion beim Klicken Ausführen
Private Sub Befehl19_Click()
    toJSON ("qryExport")
End Sub

'Funktion beim Schließen des Formulars ausführen
Private Sub Befehl19_Exit(Cancel As Integer)
    toJSON ("qryExport")
End Sub

'Funktion toJSON
Function toJSON(PassTblQry)

    Dim mydb As DAO.Database, rs As Recordset           'Variablen deklarieren
    Dim VarField(255), VarFieldType(255)
    Dim fld As DAO.Field, VarDat As String, VarNummer As String

    'Pfad und Name der js-Datei festlegen
    Set db = CurrentDb
    fn = CurrentProject.Path + "\\" + "Abfrage" + ".js"

    Dim fsT As Object          'Stream-Objekt erstellen
    Set fsT = CreateObject("ADODB.STREAM")
    fsT.Type = 2
    fsT.Charset = "utf-8"
    fsT.Open
    Recs = DCount("*", PassTblQry)           ' Datensätze zählen
    Set rs = db.OpenRecordset("Select * from [" & PassTblQry & "]")

    'Nonulls: entfernt alle leeren Felder in der js-Datei
    Nonulls = True

    'Feldnummer, Feldername und Feldtyp in Array speichern
    fieldcount = 0
    For Each fld In rs.Fields
        fieldcount = fieldcount + 1
        VarFieldType(fieldcount) = "TEXT"
        Select Case fld.Type

```

```

Case 4, 5, 6, 7
    'Feldarten: 4=long, 5=Currency, 6=Single, 7=Double
    VarFieldType(fieldcount) = "NUMBER"
End Select
Next

Set fld = Nothing

fsT.WriteLine "data=[ "           ' "data='[" in js-Datei schreiben

Do While Not rs.EOF            'ausführen wenn (nicht letztes Feld)

    fsT.WriteLine "{"           'Klammer vor Datensatz öffnen

    'Beschreiben der Datei
    For looper = 1 To fieldcount
        VarFT = VarFieldType(looper)
        If VarFT = "NUMBER" Then QuoteID = ""
        QuoteID = Chr(34)
        If IsNull(rs(VarField(looper)).Value) Then
            VarDat = "Null": QuoteID = ""
            If Nonulls = True Then VarDat = "": QuoteID = Chr(34)
            If Nonulls = True And VarFT = "NUMBER"
                Then VarDat = "0": QuoteID = ""
            Else
                VarDat = Trim(rs(VarField(looper)).Value)
            End If
            VarDat = Replace(VarDat, Chr(34), "\\" + Chr(34))      'ersetzt " durch \\
        jsonRow = Chr(34) & VarField(looper) & Chr(34) & ":" & QuoteID & VarDat & QuoteID

        If looper < fieldcount Then jsonRow = jsonRow & ","
        fsT.WriteLine jsonRow
    Next looper
    fsT.WriteLine "}"           'Klammer nach Datensatz schließen
    rs.MoveNext

    'Beistrich einfügen, wenn Datensatz nicht der letzte ist
    If Not rs.EOF Then
        fsT.WriteLine ","
    Else
        fsT.WriteLine ""
    End If

    Loop
    fsT.WriteLine "]'"           'Klammern schließen

Close           'speichern und schließen
fsT.SaveToFile fn, 2
End Function

```

## 2 Drucken der Etiketten

### 2.1 Thermotransferdruck

Die Etiketten werden alle im Thermotransferverfahren gedruckt. Dieses ist für Barcodedruck am weitesten verbreitet. Dabei wird eine mit temperaturempfindlicher Farbe beschichtete Folie zwischen dem Trägermaterial und dem Thermo-Druckkopf durchgeführt. Beim Druckvorgang schmelzen die Heizelemente punktuell Teile des Farbbandes, welche auf das Trägermaterial übertragen werden. Die Farbe kühlt auf dem Trägermaterial ab und haftet darauf. Mit diesem Verfahren können verschiedenste Arten von Medien bedruckt werden, jedoch sind materialabhängig unterschiedliche Farbfolien notwendig. Die Drucke sind besonders lange haltbar und je nach Trägermaterial auch beständig gegen Hitze, Kälte, Licht, Nässe und Abrieb. Aufgrund der hohen Auflösung ist das Thermotransferverfahren besonders gut für das Drucken von Barcodes geeignet, da ungenaue Drucke zu fehlerhaften Barcodes führen können. Weiters erfolgt der Druck im Vergleich zu anderen Verfahren mit hoher Geschwindigkeit. Thermotransferdrucker sind grundsätzlich robust und wartungsarm. Mit diesem Verfahren kann in der Regel nur einfarbig gedruckt werden, in seltenen Fällen sind zwei Farben möglich. Da zwei Verbrauchsmaterialien, nämlich das Trägermaterial und das Farbband, benötigt werden, ist das Thermotransferverfahren teuer und abfallintensiv. Detailliertere Informationen zu diesem Verfahren sind den Quellen [3] [4] [5] zu entnehmen.

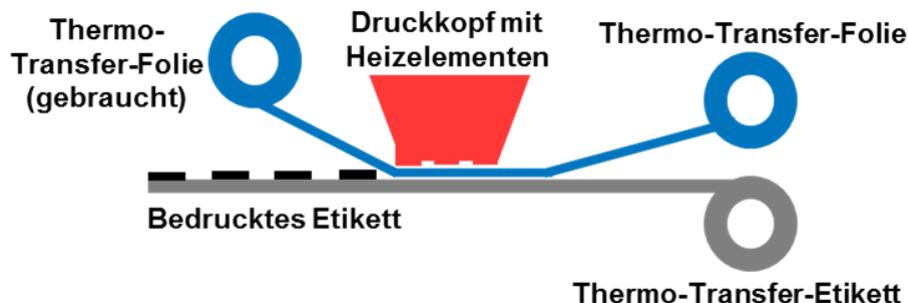


Abbildung 15: Schema des Thermotransferdrucks [6]

## 2.2 Drucker und Druckvorgang

Die Etiketten werden mit dem Thermotransferdrucker TSC TTP-247 [7] gedruckt.



Abbildung 16: Drucker TSC TTP-247

Die Rolle mit dem Trägermaterial wird mithilfe der türkisen Spindel befestigt und mithilfe der Führungen und der Antriebswalze unter dem Druckkopf hindurchgezogen.

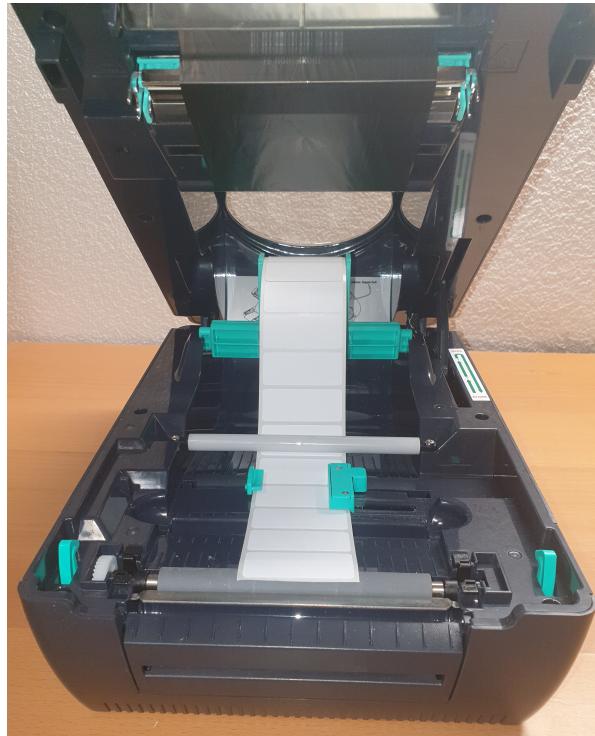


Abbildung 17: Aufgeklappter Drucker mit Sicht auf das Trägermaterial

Das Farbband wird über zwei angetriebene Spindeln zwischen dem Druckkopf und dem Trägermaterial durchgeführt. Dabei wird es von der hinteren Spindel abgespult und von der vorderen wieder aufgerollt. Da die Farbbänder nur einseitig Farbe abgeben, ist es wichtig, dass dem Trägermaterial die matte Seite zugewandt ist.



Abbildung 18: Drucker mit Sicht auf Farbband

Wenn die Verbrauchsmaterialien (siehe Kapitel 2.4) korrekt eingelegt sind, muss der Drucker noch mit Strom versorgt sowie über die USB-Schnittstelle mit einem Computer verbunden werden. Anschließend können die Etiketten, wie in Kapitel 2.3 genauer beschrieben, ausgedruckt werden. Für die korrekte Positionierung des gedruckten Objekts sorgt ein Sensor unter dem Trägermaterial, welcher die Abstände zwischen den einzelnen Etiketten erkennt. Während des Druckvorgangs werden das Trägermaterial und das Farbband mit der gleichen Geschwindigkeit unter dem Druckkopf durchgeführt. Dieser schmilzt das Farbband, welches die Farbe wiederum auf das Trägermaterial abgibt.



Abbildung 19: Druckvorgang

Nach dem Drucken bleibt das gedruckte Objekt auf der Farbfolie erkennbar, wie in Abbildung 18 und Abbildung 20 zu sehen. Dies geschieht, da an diesen Stellen die komplette Farbe abgegeben wird und die Folie dadurch transparent wird, aber die nicht gedruckten Stellen unverändert bleiben.



Abbildung 20: Benutztes Farbband

## 2.3 Design der Etiketten

Die Etiketten werden im Programm “BarTender UltraLite“ [8] entworfen. Dazu wurden zunächst drei Vorlagen für die jeweiligen Etiketten-Größen (siehe Kapitel 2.4)

erstellt. Dazu müssen im Programm zunächst die exakten Dimensionen der Etiketten sowie deren Position und die Breite der Ränder eingegeben werden. Danach kann bereits mit dem Entwurf der Oberfläche begonnen werden. Zur Verfügung stehen alle gängigen Barcodes. Weiters können Textfelder, Formen und Grafiken verwendet werden. Die eingefügten Objekte können einfach angepasst und beliebig ausgerichtet werden. Danach kann direkt mit dem Drucken begonnen werden. Dieses geschieht, gleich wie mit einem einfachen Drucker, über ein Druckmenü. Dort können noch weitere Druckeinstellungen vorgenommen werden. Nach Bestätigung wird das Etikett in der gewünschten Menge vom verbundenen Drucker ausgedruckt (siehe Kapitel 2.2).



Abbildung 21: Etiketten Standard klein in der BarTender-Ansicht



Abbildung 22: Etiketten Standard mittel in der BarTender-Ansicht



Abbildung 23: Patch-Etiketten mittel in der BarTender-Ansicht

### 2.3.1 Codes

Zur Auswahl an Codes standen zunächst QR-Code, Aztec-Code und Code128. Ein wichtiges Kriterium ist die Fehlerkorrektur, da die Barcodes im Einsatz teilweise verschmutzt oder zerkratzt werden können. Der QR-Code bietet zwar eine ausreichende Fehlerkorrektur, jedoch müssen dazu alle Eckmarkierungen erkennbar sein.

Der Atztec-Code hat diesen Nachteil aufgrund seines Aufbaus trotz einer höheren Fehlerkorrektur nicht, wodurch sich zunächst für diesen entschieden wurde. Jedoch wurde bei den ersten Druckversuchen festgestellt, dass dieser Code mit dem Drucker (siehe Kapitel 2) in der benötigten Größe nicht genau genug gedruckt werden konnte. Dadurch waren die Codes auf den gedruckten Etiketten nicht oder nur schwer lesbar. Dadurch wurde der Code128 gewählt. Durch den Aufbau des Codes ist die Fehlerkorrektur zwar niedriger, jedoch muss nur ein kleiner Teil eines jeden Balken vorhanden sein, um lesbar zu bleiben.

Genauere Informationen zu diesen Codes sind auf der Website von Seagull Scientific [9] [10] [11] ersichtlich.

### RFID

RFID-Transponder eignen sich nicht für diese Anwendung.

Die Transponder können nur auf flachen, nicht stark beanspruchten Oberflächen angebracht werden. Da die Feuerwehr Hohenkogl sich ein einheitliches System wünscht, wurden Barcodes gewählt. Weiters sind RFID-Transponder wesentlich teurer als bedruckte Etiketten.

## 2.4 Etiketten

### 2.4.1 Etiketten Standard

Die Standardetiketten sind auf der Oberfläche silberfarbig und bestehen aus Polyester. Sie werden einfach auf das Objekt aufgeklebt, welches zur besseren Haftung zuvor entfettet werden sollte. Bei der Anbringung ist es wichtig, dass das Etikett nicht stark konvex gewölbt ist, da ansonsten der Barcode nicht gelesen werden kann. Durch die starke Klebekraft haften sie auf nahezu allen flachen Oberflächen und sind nur schwer wieder abzulösen. Diese Etiketten wurden in zwei verschiedene Größen angeschafft.

#### **Etiketten Standard klein [12]**

Maße: 30x15mm

Menge: 1 Rolle zu je 1000 Stück



Abbildung 24: Etiketten Standard klein auf der Rolle

### **Etiketten Standard mittel [13]**

Maße: 50x25mm

Menge: 1 Rolle zu je 1000 Stück



Abbildung 25: Etiketten Standard mittel auf der Rolle

#### **2.4.2 Patch-Etiketten**

Die Patchetiketten sind auf der Oberfläche weiß und eignen sich ausschließlich für die Anbringung auf Textilien. Die Etiketten bestehen aus Polyester und sind waschbeständig. Angebracht werden sie mithilfe eines Bügeleisens. Dazu wird das Etikett zunächst mit der Hand auf das Objekt gedrückt, worauf es bereits leicht haftet. Anschließen wird mit dem Bügeleisen bei einer Temperatur von etwa 200°C für etwa 15 Sekunden das Etikett fest angedrückt. Von diesen Etiketten wird lediglich eine Größe benötigt.

### **Patch-Etiketten mittel [14]**

Maße: 46x19mm

Menge: 2 Rollen zu je 1000 Stück



Abbildung 26: Patch-Etiketten mittel auf der Rolle

## 2.5 Farbfolien

Für das Bedrucken der Etiketten sind Farbbänder nötig (siehe Kapitel 2). Diese bestehen aus Kunstharz und sind schwarz. Aufgrund der unterschiedlichen Arten von Trägermaterialien sind auch zwei Arten von Farbbändern nötig.

### **Farbband Standard [15]**

Maße: 70mmx300m

Menge: 1 Rolle

### **Farbband Bügel/Patch [16]**

Maße: 70mmx100m

Menge: 1 Rolle



Abbildung 27: Farbband auf Spindeln

### 3 Mobiles Gerät

Für die Hardware des Scanners wurde ein Android-Smartphone verwendet, da es die benötigte Kamera besitzt und die Möglichkeit bietet Eingaben zu tätigen. Geplant war es einen Raspberry PI zu verwenden, da dieser keine Kamera und Tastatur besitzt, müssten diese als externe Geräte mit dem Raspberry in ein Gehäuse verbaut werden. Die Wahl des Geräts fiel auf das Smartphone, da dieses für denselben Funktionsumfang weniger Platz benötigt und somit handlicher ist. Ein weiterer Punkt der für das Smartphone spricht ist die bekannte Bedienung, mit dem Raspberry wäre es nötig allen Benutzern die Bedienung erneut zu lernen.

### 4 Datenübertragung zwischen Datenbank und Mobilgerät

Die Anwendung benötigt die Daten um durch eine Inventarnummer bestimmte Informationen über einen Gegenstand auszugeben. Die Verbindung zur Datenbank soll verlässlich und möglichst aktuell sein, diese Anforderungen bieten Cloud-Lösungen. Als Cloud Anbieter wurde Dropbox [17] gewählt da die FF Hohenkogl dieses System bereits in verwendung hat und es möglich ist eine Datei über einen Link zu teilen.

#### 4.1 Erzeugung des Dropbox-Links

Bei der ersten Benützung muss ein neuer Link zur Datei auf Dropbox erstellt werden, dies muss nur einmal geschehen da der Link gleich bleibt wenn die Daten überschrieben werden. In Abbildung 28 ist zu sehen dass hierzu nur der Knopf Link kopieren gedrückt werden muss. Es ist zu beachten dass dieser Link zur Datei auf Dropbox führt und nicht die Rohdaten ausgibt. Um an die Rohdaten zu gelangen muss im Link dass "www" durch "dl" ersetzt werden.

Link zu der Dropbox Datei:

<https://www.dropbox.com/s/qio02ic7s56so1x/Datensatz.js?dl=0>

Link zu den Rohdaten:

<https://dl.dropbox.com/s/qio02ic7s56so1x/Datensatz.js?dl=0>

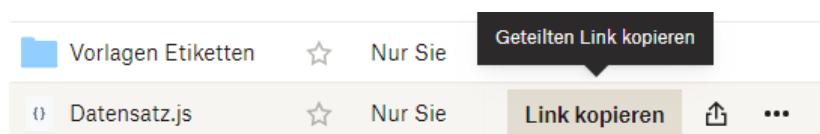


Abbildung 28: Dropbox-Link erzeugen

## 5 Anwendung

Zur Realisierung der Webanwendung wurden JavaScript, HTML und CSS verwendet. Um die Website als Smartphone Anwendung zu verwenden, wurde eine progressive Web App daraus erstellt. Die Website wird auf GitHub Pages [18] gehostet, da es kostenlos ist und eine sehr einfache Möglichkeit zum Bearbeiten des Programmcodes bietet, auf GitHub Pages wird in Kapitel 5.7 genauer eingegangen.

### 5.1 Scanner

Als Scanner wird die vorgefertigte Bibliothek "html5-qrcode" [19] genutzt, sie bietet den Zugriff auf die Kamera sowie eine Visualisierung bei erfolgreichem Scan. Diese Bibliothek unterliegt der Apache License 2.0 und ist daher frei zur Nutzung, sie unterliegt dem Urherberrecht von mebjas.

#### 5.1.1 Code-Scanner

Um die Bibliothek zu verwenden, muss sie in dem HTML-Dokument als Script geladen werden.

```
<script src="https://unpkg.com/html5-qrcode"></script>
```

Für den Scanner muss im HTML-Dokument ein Container erstellt werden, auf den über JavaScript zugegriffen werden kann.

```
<div id="reader"></div>
```

Nun kann der Scanner im JavaScript Programm initialisiert werden. Von der Klasse wird ein Objekt erstellt und eine config Variable wird erzeugt. Die config Variable "conf" bestimmt die Bildwiederholrate des Scanners, das Seitenverhältnis und die Größe des Kamerafelds.

```
const html5QrCode = new Html5Qrcode("reader");
const conf = { fps: 10, aspectRatio: 1.0, qrbox: 200 };
```

Für den Scanner wurde eine eigene Klasse erstellt, da sie zur erleichterten Bedienung beiträgt. Im Konstruktor werden alle für den Scanner nötigen Variablen konfiguriert. Zum Initialisieren der Klasse muss der Container für den Scanner und die config Variable übergeben werden.

```
class Camera {
  constructor(scanner, config) {
    this.scanner = scanner;
    this.config = config;
    this.ready = false;
  }
}
```

Mit der Methode "film(success)" kann man den Scanner starten, mit "success" wird eine Funktion übergeben, welche der Scanner bei Erfolg aufruft. Mit der Einstellung facingMode: "environment" wird angegeben, dass wenn vorhanden, die äußere Kamera des Geräts verwendet wird.

```
film(success) {
    this.scanner.start({ facingMode: "environment" }, this.config, success);
}
```

Die Methode stopfilm() ermöglicht es, den Scanner zu stoppen und mögliche Fehlermeldungen zu verarbeiten.

```
stopfilm() {
    this.scanner
        .stop()
        .then((ignore) => {
        })
        .catch((err) => {
    });
    this.scanner.clear();
}
```

War ein Scan erfolgreich, so wird mit der Methode "dectxt(decodedText)" mit "decodedText" der eingescannte Wert übergeben, dann wird der Wert gespeichert sodass die ganze Klasse darauf Zugriff hat. Um den Wert weiter zu verarbeiten, wird er in die Gruppe und Nummer des Gegenstands unterteilt, dass dient dazu die ganze Inventarnummer in einem speziellen Format (Kapitel 5.2) mit der Methode "getid()" auszugeben.

```
dectxt(decodedText) {
    this.text = decodedText;

    let idfirst = decodedText.charAt(0) + decodedText.charAt(1);
    this.group = parseInt(idfirst);

    let idlast = decodedText.charAt(2) + decodedText.charAt(3) +
        decodedText.charAt(4) + decodedText.charAt(5);
    this.number = parseInt(idlast);}

getid() {
    return this.group + "/" + this.number;
}
```

Um die Klasse zu verwenden, muss ein Objekt von ihr erstellt werden.

```
const cam = new Camera(html5QrCode, conf);
```

## 5.2 Datenverarbeitung

Zum Speichern der Daten wird ein JSON-String in einer separaten JavaScript Datei als Variable gespeichert (Kapitel 1.6), da in diesem Format auch der Zugriff von der Anwendung erlaubt ist. Wenn JSON als Dateiformat verwendet wird, ist es der Anwendung nicht möglich auf die Datei zuzugreifen, da durch Cross-Origin Resource Sharing (CORS) nur der Zugriff auf HTTP-Resourcen der gleichen Herkunft gestattet ist. Die Daten werden in folgendem Format gespeichert.

```
data= '[JSON-Daten]'
```

Die JSON Daten sind in folgendem Format in der Variable enthalten.

```
{
  "Inventarnummer": "8/13",
  "Name": "Abschleppseil",
  "Raum": "Garage",
  "Mitarbeiter Nummer": "0",
  "Vorname": "Max",
  "Nachname": "Mustermann",
  "Baujahr": "2000",
  "Prüfpflichtig": "Ja"
},
```

Wenn Bedarf besteht, mehr oder andere Informationen anzuzeigen, muss die Access-Abfrage geändert werden und die Ausgabe der HTML-Datei (Kapitel 5.4) verändert werden. Im Barcode ist die Inventarnummer wie in Abbildung 29 zu sehen mit führenden Nullen gespeichert, jedoch wird sie im Programm in diesem Format verwendet: 5/84. Dadurch wird die Access-Abfrage erleichtert und das Programm beschleunigt.



Abbildung 29: Code 128

## 5.3 Initialisieren

Um die Anwendung verwenden zu können, muss zuerst ein Link zu den Daten initialisiert werden. Dieser wird mit einem QR-Code eingescannt. Diese Initialisierung dient als Sicherheitsmaßnahme, sodass nur Personen mit Zugriff auf die Datenbank, die Anwendung verwenden können. Zur Verwaltung des Links, wurde eine Klasse erstellt. Um die Klasse zu verwenden muss beim Erstellen des Objekts der Name des Local storage Objekts übergeben werden, mehr ist für die Klasse nicht nötig.

```
class Link {
    constructor(name) {
        this.name = name;
    }
}
```

Der erste Teil dient zum Erstellen und Auslesen des Local storage [20]. Um das Objekt zu erstellen, muss man die Methode ”localStorage.setItem(Name, Inhalt)” aufrufen. Um die Daten aus dem Local storage auszulesen, wird die Funktion ”localStorage.getItem(Name)” verwendet. Da die Klasse schon mit dem Namen für das Objekt initialisiert wurde, wird dieser nicht mehr benötigt, der Name ist in der Variable ”this.name” gespeichert und wird der Methode übergeben.

```
setdata(value) {
    localStorage.setItem(this.name, value);
}
getdata() {
    return localStorage.getItem(this.name);
}
```

### 5.3.1 Überprüfen

Mit der Methode ”checklink()” ist es möglich, den aktuellen Link zu überprüfen, weiters prüft diese Methode, ob in der URL das Schlüsselwort /init enthalten ist, damit diesem ein neuer Link initialisiert werden kann. Wenn nichts im Speicherobjekt vorhanden ist, wird die Methode ”initlink()” aufgerufen, um einen neuen Link zu initialisieren.

```
checklink() {
    let dataurl = this.getdata();
    let geturl = new URLSearchParams(window.location.search);
    let link = geturl.get("k");
    if (link == "/init") {
        this.initlink();
    } else {
        if (dataurl != null && dataurl.charAt(0) == "h" && dataurl.charAt(1) == "t") {
            showdiv.style.height = 0;
            this.loadlink();
        } else {
            this.initlink();
        }
    }
}
```

Wenn ein Speicherobjekt vorhanden ist, werden die Daten aus ihm extrahiert und der Link wird in der Methode "loadlink()" als Script in die HTML-Datei geladen. Hier wird auch der in der Datei enthaltene String in einen JSON String umgewandelt, so dass er im gesamten Programm verwendet werden kann.

```
loadlink() {
    let dataurl = this.getdata();
    if (dataurl != null && dataurl.charAt(0) == "h" && dataurl.charAt(1) == "t") {
        var script = document.createElement("script");
        script.onload = function () {
            obj = JSON.parse(data);
        };
        script.src = dataurl;
        document.getElementsByTagName("head")[0].appendChild(script);
        showdiv.style.height = "fit-content";
        showtext.innerHTML = "Datenbank geladen!";
    }
}
```

Wenn kein Link gespeichert war, wird die Methode "initlink()" aufgerufen. Diese startet die Kamera und speichert bei erfolgreichem Scan einen neuen in dem dafür vorgesehenen Speicherobjekt. Zur manuellen Eingabe wird die Funktion des "Los" Knopfs geändert.

```
initlink() {
    showtext.innerHTML = "Bitte Initialisierung durchführen!";
    searchbar.placeholder = "Bitte Link einfügen!";
    gobutton.onclick = function () {
        initlink();
    };

    const onsuccess = (decodedText, decodedResult) => {
        this.setdata(decodedText);
        cam.stopfilm();
        this.loadlink();
        gobutton.onclick = function () {
            ManualID();
        };
        searchbar.placeholder = "ID-Eingeben..";
    };
    cam.film(onsuccess);
}
```

Wenn der Link erneuert werden muss, erscheint beim Starten der Anwendung der Scanner und ein Textfeld informiert darüber, dass der Link zur Datenbank initialisiert werden muss. In Abbildung 30 ist der Initialisierungsbildschirm zu sehen. In Abbildung 31 ist der Bildschirm zu sehen, wenn der Link erfolgreich initialisiert wurde.

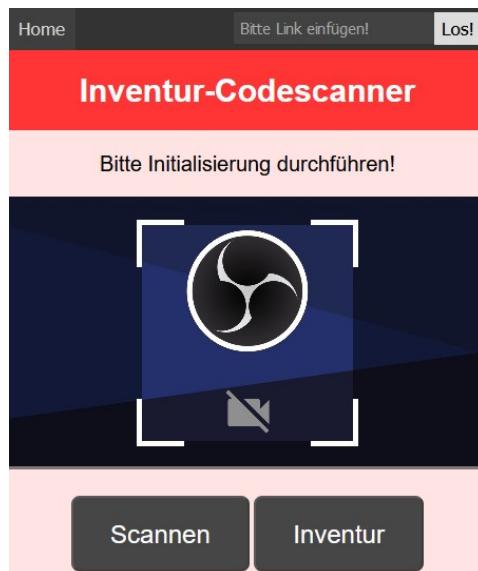


Abbildung 30: Initialisierung



Abbildung 31: Datenbank geladen

## 5.4 Ausgabe

Wenn die Datenbank erfolgreich initialisiert wurde, ist es möglich, Informationen über einen bestimmten Gegenstand zu bekommen. Wenn der Knopf "Scannen" gedrückt wird, wird die Funktion "Scan()" aufgerufen. Diese startet die Kamera und leitet bei erfolgreichem Scan auf die Ergebnis Seite (Abbildung 32) weiter. Um die gescannte Nummer zur nächsten Seite weiter zu geben, wird diese am Ende der URL gespeichert. Die URL sieht dann wie folgt aus: "<https://ff-hk.github.io/showdata.html?k=4/1670>". Am Ende kann man die Inventarnummer sehen.

```

function Scan() {
    try {
        cam.stopfilm();
    } catch {}
    showdiv.style.height = "fit-content";
    showtext.innerHTML = "Bitte einen Barcode Scannen";

    const onsuccess = (decodedText, decodedResult) => {
        cam.dectxt(decodedText);
        resulte = cam.getid();
        cam.stopfilm();
        document.close();
        window.location.replace("showdata.html?k=" + resulte);
    };
    cam.film(onsuccess);
}

```

In der Ergebnis Seite wird die Inventarnummer von der URL extrahiert.

```

const geturl = new URLSearchParams(window.location.search);
const scannedid = geturl.get("k");

```

Diese Seite muss auch den Link zur Datenbank laden. Sie nutzt, zum Extrahieren der Daten, die Methode ”getdata()” der Klasse ”Link” (Kapitel 5.3). Um den richtigen Gegenstand anzuzeigen, wird die Methode ”filter()” von JavaScript verwendet, sie gibt alle Werte in einem Array zurück, welche einer bestimmten Bedingung entsprechen, in diesem Fall wird nach der Inventarnummer gefiltert.

```
var data = object.filter(object=> object.invInvNummer === scannedid)
```

Die Daten werden, wie in Abbildung 32 zu sehen ausgegeben, diese Ausgabe kann je nach Bedarf bestimmte Informationen anzeigen. Die Informationen werden mit folgendem Code in das HTML-Feld gespeichert.

```
document.getElementById("showname").innerHTML = data[0].invName + "<br>";
```

Das HTML Feld besteht aus einem allgemeinen Container und einem Label für jedes Feld. Der Text in dem Feld kann durch die vorhandenen Id für das Feld verändert werden.

```

<label for="showname">Name:</label> <br />
<div class="output">
    <p id="showname">Placeholder</p>
</div>

```

Zur Hauptseite gelangt man durch drücken des ”Home” Knopfs in der Navigationsleiste.



Abbildung 32: Ergebniss des Scans

## 5.5 Inventur

Um eine Inventur zu starten, muss zunächst der Knopf "Inventur" auf der Startseite, wie in Abbildung 31 zu sehen gedrückt werden. Die Elemente, welche auf der Startseite zu sehen sind, werden entfernt und es werden die neuen für die Inventur geladen, der Scanner jedoch wird nicht entfernt, denn er kann durch die vorher definierte Klasse sehr einfach weiterverwendet werden.

### 5.5.1 Inventur Knopf

Die Funktion "Inv()" wird beim Drücken des "Inventur" Knopfs aufgerufen. Als Erstes wird die Funktion des "Los" Knopfs geändert, damit eine manuelle Inventur möglich ist. Das Programm versucht dann die Kamera zu stoppen, falls sie noch aktiv sein sollte. Um die richtige Seite anzuzeigen, wird die Funktion "showinv()" aufgerufen.

```
function Inv() {
    gobutton.onclick = function () {
        manualinv();
    };
    try {
        cam.stopfilm();
    } catch {}
    showinv();
    showdiv.style.height = "fit-content";
```

In dieser nachstehenden Funktion werden die sichtbaren Knöpfe versteckt und die neuen Knöpfe angezeigt.

```

function showinv() {
    scanbutton.style.visibility = "hidden";
    invreadybutton.style.visibility = "visible";
    saveinventory.style.visibility = "visible";
    getinventory.style.visibility = "visible";
    readydiv.style.height = "fit-content";
    loaddiv.style.height = "fit-content";
    invbutton.style.visibility = "hidden";
}

```

Um einen Code einzuscannen, wird die Kamera gestartet und der Hilfe Text geändert. Wenn der Scan erfolgreich war, wird die Funktion ”id()” (Kapitel 5.5.2) aufgerufen, diese verwertet die Inventarnummern.

```

const onsuccess = (decodedText, decodedResult) => {
    cam.dectxt(decodedText);
    id();
};

cam.film(onsuccess);
if (init == true) {
    showtext.innerHTML = "Bitte mit einem Barcode Initialisieren!";
}
}

```

### 5.5.2 Verarbeitung der gescannten Nummern

Nach Aufruf der Funktion, wird zunächst geprüft ob schon eine Nummer initialisiert wurde. Wenn dass nicht der Fall ist, wird aus der ersten Nummer der Raum ausgelesen und es werden dann alle Gegenstände aus diesem Raum in ein Array gespeichert. Die Funktion ”obj.filter()” [21] von JavaScript ermöglicht dies sehr schnell und einfach umzusetzen. Es werden alle Gegenstände in ein zweites Array gespeichert, welches als Vergleich dient.

```

function {
    if (init == true) {
        init = false;
        scanneddata.push(cam.getid());
        var dat = obj.filter((obj) => obj.invInvNummer === cam.getid());
        room = dat[0].raumName;

        realdata = obj.filter((obj) => obj.raumName === room);

        realdata.forEach((element) => {
            realdata[realdata.indexOf(element)] = element.invInvNummer;
        });

        realdata.forEach((element) => {
            comparedata.push(element);
        });
        showtext.innerHTML = "Raum: " + room;
    }
}

```

Wenn keine Initialisierung mehr erforderlich ist, wird die gescannte Nummer einfach in ein Array, welches alle bereits gescannten Inventarnummern beinhaltet gespeichert.

```

} else if (init == false) {
    if (!scanneddata.includes(cam.getid())) {
        scanneddata.push(cam.getid());
    }
}
Inventoryresult();

```

Wenn die neue Nummer gespeichert oder die Initialisierung durchgeführt wurde, wird die Funktion "Inventoryresult()" aufgerufen. Diese ist dafür zuständig, die Gegenstände anzuzeigen.

### 5.5.3 Anzeige der Inventurdaten

Zur Ausgabe der Gegenstände wird eine HTML Liste verwendet.

```

function Inventoryresult() {
    let list = document.getElementById("myList");
    list.innerHTML = "";

```

Um zu prüfen, ob ein Gegenstand in diesen Raum gehört oder nicht, werden die gescannten Nummern mit dem Vergleichsarray verglichen. Wenn sie im Array vorhanden sind, werden sie daraus entfernt, um nur die nicht gescannten Nummern anzuzeigen. Falls die Nummer nicht im Vergleichs-Array vorhanden ist, wird sie in das "notrightdata" Array gespeichert. Ist der Gegenstand nicht in der Datenbank vorhanden, wird der Nutzer darüber informiert.



Abbildung 33: Inventur mit falschem und richtigem Gegenstand

```

scanneddata.forEach((item) => {
    try{
        var resultel = obj.filter((obj) => obj.invInvNummer === item);
        if (comparedata.includes(resultel[0].invInvNummer)) {
            comparedata.splice(comparedata.indexOf(resultel[0].invInvNummer), 1);
        } else if (!realdata.includes(resultel[0].invInvNummer)
        && !notrightdata.includes(resultel[0].invInvNummer)) {
            notrightdata.push(resultel[0].invInvNummer);
        }
    }
    catch(err)
    {
        alert("Gegenstand nicht in der Datenbank enthalten!");
    }
});

```

Zum Anzeigen der gefilterten Daten wird ein neues Element in der Liste erstellt und es werden die relevanten Daten als Text darin gespeichert. Zuletzt wird dieses Element der Liste hinzugefügt. Dieser Teil gibt die Gegenstände aus, welche gescannt wurden, sich aber nicht in diesem Raum befinden.

```

notrightdata.forEach((item) => {
    var resultel = obj.filter((obj) => obj.invInvNummer === item);
    let li = document.createElement("li");
    li.classList.add("notinventory");
    li.innerText = "Nummer: " + resultel[0].invInvNummer +
    "\n" + "Name: " + resultel[0].invName;
    list.appendChild(li);});

```

Der Teil des Codes funktioniert beinahe identisch zu dem vorher angeführten, es werden jedoch nur die Gegenstände ausgegeben, welche noch nicht gescannt wurden, aber in diesem Raum vorhanden sein sollten.

```

comparedata.forEach((element) => {
    var resultel = obj.filter((obj) => obj.invInvNummer === element);
    let li = document.createElement("li");
    li.classList.add("inventory");
    li.innerText = "Nummer: " + resultel[0].invInvNummer +
    "\n" + "Name: " + resultel[0].invName;
    list.appendChild(li);
}

```

#### 5.5.4 Speichern der laufenden Inventur

Es ist möglich, eine bereits begonnene Inventur zu speichern und zu einem späteren Zeitpunkt fortzusetzen. Um dies zu erreichen, werden die Arrays, welche für die Inventur benötigt werden, im Local storage gespeichert, dieser ist auch nach Beenden des Browsers oder Neustarten des Geräts noch verfügbar. Geplant war es, die Daten in einem Cookie zu speichern, da dieser jedoch zu wenig Speicherplatz besitzt musste auf

Local storage umgestellt werden. Wenn in der Anwendung die Inventur gespeichert werden soll, muss der "Speichern" Knopf wie in Abbildung 33 zu sehen gedrückt werden.

```
function SaveInventory() {
    showtext.innerHTML = "Daten gespeichert!";

    localStorage.setItem("scanned", scanneddata);
    localStorage.setItem("compare", comparedata);
    localStorage.setItem("real", realdata);
}
```

Wenn die Daten geladen werden sollen, muss der "Laden" Knopf gedrückt werden. Im Code wird dann die Gegenoperation zu "setItem" durchgeführt, nämlich "localStorage.getItem("name")". Beim Speichern in das Array werden die Daten an den Beistrichen getrennt, da sie so im Local storage gespeichert werden.

```
function GetInventory() {
    scanneddata = localStorage.getItem("scanned").split(",");
    comparedata = localStorage.getItem("compare").split(",");
    realdata = localStorage.getItem("real").split(",");

    var dat = obj.filter((obj) => obj.invInvNummer === scanneddata[0]);
    room = dat[0].raumName;
```

Um wieder den richtigen Raum zu erkennen, wird eine Inventarnummer aus dem Array mit den bereits gescannten Codes auf ihren Raum überprüft, dieser wird dann in eine globale Variable gespeichert.

### 5.5.5 Ausgabe als PDF

Wenn die Inventur beendet ist, ist es möglich, diese als PDF Datei auszugeben, dazu wurde eine Klasse erstellt, um die Befehle und die Abstände für die PDF leichter zu verwalten. Basis für diese Klasse ist die Bibliothek "jsPDF" [22] von James Hall, sie unterliegt der MIT Lizenz, welche es erlaubt, die Bibliothek weiter zu verwenden, sie unterliegt auch einem Urheberrecht von James Hall und der yWorks GmbH. Als Erstes werden im Konstruktor alle nötigen Informationen für das Erstellen der PDF initialisiert, die Klasse benötigt einen x Abstand, dieser gibt an, wie weit der Text vom Rand des Blattes entfernt sein soll. Weiters werden der Zeilenabstand zwischen den Gegenständen und die Organisation für die Kopfzeile benötigt.

```

class PDF {
    constructor(x, columnspace, organisation) {
        this.pdf = new jsPDF("p", "mm", "a4");
        this.pdf.setFont("Arial");
        this.pdf.setFontSize(12);
        this.pageheight = this.pdf.internal.pageSize.height;
        this.pagewidth = this.pdf.internal.pageSize.width;
        this.space = columnspace;
        this.xc = x;
        this.pdfnum = 1;
        this.headerspace = 21;
        this.yc = columnspace + this.headerspace;
        this.org = organisation;
        this.date = date;
        this.Header();
    }
}

```

Die Kopfzeile wurde in eine eigene Methode geschrieben, da sie bei jeder Seite neu erstellt werden muss. In dieser Methode werden drei Texte in die gleiche Zeile am Seitenanfang geschrieben, die Organisation, "Inventur" und das aktuelle Datum.

```

Header() {
    this.pdf.text(this.org, 10, this.headerspace / 2 + 5);
    this.pdf.text("Inventur", this.pagewidth / 2 - 10, this.headerspace / 2 + 5);
    this.pdf.text(this.date, this.pagewidth - this.date.length * 3,
    this.headerspace / 2 + 5);
}

```

Die Methode "checkforheight()" prüft, ob das Seitenende schon erreicht wurde und erstellt, wenn dass der Fall ist, eine neue Seite und die Kopfzeile für diese. Um eine neue Seite zu erstellen, wird die Methode "pdf.addPage()", welche von der Bibliothek vorgefertigt ist, verwendet.

```

checkforheight() {
    if (this.pageheight - 14 < this.yc) {
        this.pdf.addPage();
        this.Header();
        this.yc = 14 + this.headerspace;
    }
}

```

Der Text wird mit der Methode "Write()" geschrieben, sie nutzt die von der Bibliothek vorgefertigte Methode "pdf.text()", welche es erlaubt, Text in die PDF zu schreiben, es muss dazu der Text, der Abstand zum Seitenrand und der Abstand zum Seitenanfang übergeben werden. Die Klasse verwaltet die Abstände durch eine beständige Variable, welche die Höhe der Zeile in Abhängigkeit vom Seitenanfang angibt.

```

Write(text) {
    this.checkforheight();
    this.pdf.text(text, this.xc, this.yc);
    this.yc = this.yc + this.space;
}

```

Es wird in der PDF auch eine Linie gezeichnet, um die Lesbarkeit zu erhöhen, diese wird mit einer vorgefertigten Methode "pdf.line()" von der Bibliothek gezeichnet.

```
Line() {
    this.pdf.line(5, this.yc - 2, 200, this.yc - 2, "F");
    this.yc = this.yc + this.space;
}
```

Um ein Fenster zum Speichern der PDF zu öffnen, wird die Methode "pdf.save" aus der Bibliothek aufgerufen, es werden der Name und die Nummer der PDF übergeben.

```
Save(name) {
    this.pdf.save(name + this.pdfnum + ".pdf");
    this.pdfnum += 1;
}
```

Als erstes wird der Raum in das PDF geschrieben und eine neue Linie erstellt.

```
function InventoryReady() {
    pdf.Write("Inventur in Raum: " + room);
    pdf.Line();
```

Um die Inventar-Daten auszugeben, wird zuerst geprüft, ob sie angezeigt werden müssen, das Array wird auf vorhandenen Inhalt geprüft. Wenn nichts im Array gespeichert ist, wird es auch nicht in der PDF ausgegeben. Es wird für jedes Element im Array eine Zeile erstellt, in diese werden alle benötigten Informationen zu dem jeweiligen Gegenstand geschrieben. Am Ende wird die PDF Datei noch gespeichert.

```
if (notrightdata.length != 0) {
    pdf.Write("Dinge die hier nicht sein sollten:");
    notrightdata.forEach((element) => {
        var comp = obj.filter((obj) => obj.invInvNummer === element);
        pdf.Write("Nummer: " + comp[0].invInvNummer + " Name: " + comp[0].invName);
    });
    pdf.Line();
}

if (comparedata.length != 0) {
    pdf.Write("Nicht eingescannt:");
    comparedata.forEach((element) => {
        var comp = obj.filter((obj) => obj.invInvNummer === element);
        pdf.Write("Nummer: " + comp[0].invInvNummer + " Name: " + comp[0].invName);
    });
}
pdf.Save("Inventur-" + room);
```

Im PDF werden zuerst falls vorhanden falsch gescannte Gegenstände und dann nicht eingescannte Gegenstände angezeigt. In Abbildung 34 ist die fertig erstellte PDF zu sehen.

FF Hohenkogl Inventur 24.3.2022

Inventur in Raum: Persönlich

Nicht eingescannt:

Nummer: 4/163 Name: Einsatzgürtel grün  
Nummer: 4/164 Name: Einsatzgürtel grün  
Nummer: 4/167 Name: Einsatzgürtel grün

Abbildung 34: PDF Ausgabe

## 5.6 Verwendung ohne Codes

Die Anwendung soll auch ohne Kamera oder mit nicht lesbaren Barcodes verwendbar sein, deshalb ist sie so aufgebaut, dass bei allen Eingaben, die durch die Kamera getätigten werden, auch die Eingabeleiste am rechten oberen Rand der Anwendung verwendet werden kann. Um dies zu erreichen, muss lediglich die Funktion, welche vom "Los" Knopf ausgeführt wird, geändert werden. Es wird zur leichteren Orientierung auch der Text im Eingabefeld geändert.

```
searchbar.placeholder = "Bitte Link einfügen!";  
gobutton.onclick = function () {  
    initlink();  
};
```

## 5.7 GitHub Pages

Um GitHub Pages zu nutzen wurde die Dokumentation von der offiziellen Seite befolgt [18][23]. Dazu muss zuerst ein Repository, bei dem der Name dem Nutzernamen entspricht, erstellt werden. Danach kann in den Einstellungen die Seite unter dem Punkt Pages erstellt werden, dazu muss ein Zweig und ein Verzeichnis des Repositorys gewählt werden, drückt man nun auf speichern wird die Seite unter der URL "username.github.io" veröffentlicht. Die "index.html" Datei muss in das angegebene Verzeichniss gelegt werden.

## 5.8 Progressive Web App

Die progressive Web App dient dazu, die Website aus dem Browser zu installieren und als normale Anwendung zu verwenden. Dazu müssen einige Schritte durchgeführt werden, Lighthouse [24], ein Tool von Google Chrome hilft dabei. Es gibt aus, welche Teile für eine funktionierende PWA fehlen. Wenn alle Anforderungen für die PWA erfüllt sind, ist das Fenster wie in Abbildung 35 zu sehen.

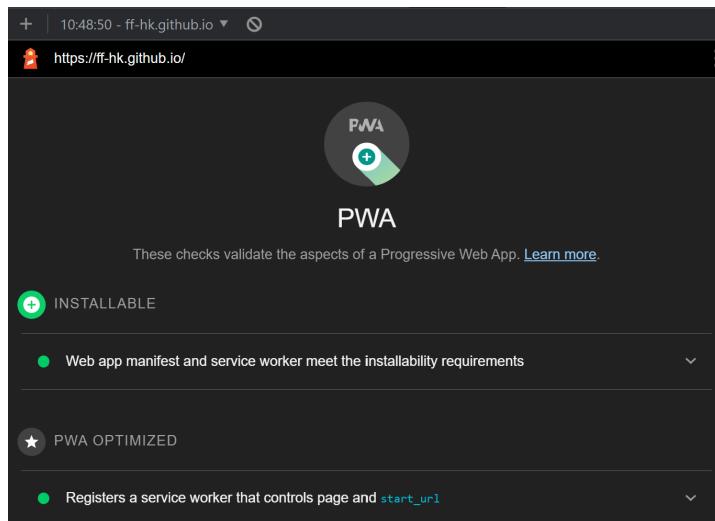


Abbildung 35: Lighthouse Erfolgreich

In der Abbildung 36 ist das Icon zu sehen, welches zum Installieren dient, Abbildung 37 zeigt die installierte Anwendung.

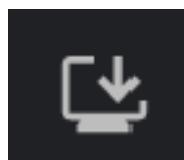


Abbildung 36: Download Icon

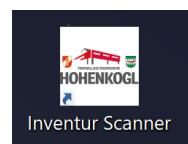


Abbildung 37: Installierte Anwendung

### 5.8.1 Progressive Web App erstellen

Um eine Web-Anwendung zu installieren, werden drei Dinge benötigt, eine Manifest Datei, ein Serviceworker und ein Icon für die App [25].

## Manifest

Die Manifest Datei ist im JSON Format, in ihr stehen wichtige Informationen. Hier werden unter anderem der Name, der Hintergrund, das Displayformat und das Icon gesetzt. Es sind zwei Icons vorhanden, da eines ein Maskable Icon ist, es wird verwendet wenn Smartphones kein standardisiertes Icon Format benutzen.

```
{
  "name": "Inventur Scanner",
  "short_name": "Inv Scanner",
  "theme_color": "#404040",
  "background_color": "#ffe4e4",
  "display": "standalone",
  "scope": "/",
  "start_url": "https://ff-hk.github.io/",
  "description": "Inventory Scanner",
  "orientation": "portrait",
  "icons": [
    {
      "src": "ffhk.png",
      "sizes": "1024x1024",
      "purpose": "maskable"
    }
  ],
  "icons": [
    {
      "src": "ffhk.png",
      "sizes": "1024x1024"
    }
  ]
}
```

## Service Worker

Der Serviceworker ist eine JavaScript Datei welche separat zum Hauptthread vom Browser läuft. Wenn eine Anfrage auf eine Ressource gestellt wird, entscheidet der Serviceworker, ob diese aus dem Zwischenspeicher, aus dem Netzwerk oder von einem lokalen Algorithmus erstellt werden soll.

```

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open('v1').then(function(cache) {
      return cache.addAll([
        'index.html',
        'showdata.html',
        'scanner.js',
        'navbar.js',
        'style.css',
        '/pwa/ffhk.png',
        '/pwa/manifest.json',
        '/classes/camera.js',
        '/classes/init.js',
        '/classes/pdf.js',
      ]);
    })
  );
});

self.addEventListener('fetch', event => {
  event.respondWith(async function() {
    const cachedResponse = await caches.match(event.request);
    if (cachedResponse) return cachedResponse;
    return fetch(event.request);
  })();
});

```

In dieser Anwendung speichert der Service worker bestimmte Ressourcen im Zwischenspeicher und versucht bei einer Netzwerkanfrage die Ressource mit dem Zwischengespeicherten Stand, zu ersetzen. Um die Ressource vom Zwischenspeicher aufzurufen prüft der Service Worker ob die angefragte Ressource gespeichert ist. Wenn dies der Fall ist ersetzt er sie mit dem lokal gespeicherten Stand, anderenfalls wird die Netzwerkanfrage normal durchgeführt [26].

## Zusammenfassung

Im Zuge dieser Diplomarbeit wurde das Inventarsystem der Feuerwehr Hohenkogl verbessert. Dazu wurde die Access-Datenbank überarbeitet. Dabei wurden Redundanzen entfernt sowie neue Abfragen hinzugefügt. Weiters wurde diese um die Berechnung verschiedener Prüfdaten sowie eines Ablaufdatums ergänzt. Um die Daten aus der Datenbank im richtigen Format in den Cloudspeicher zu exportieren, wird eine VBA-Export-Funktion angewandt. Um das Inventar zu kennzeichnen, werden Etiketten mit Barcodes verwendet. Die Etiketten werden mit einem eigenem Thermotransferdrucker gedruckt. Je nach Art der Etiketten werden diese entweder auf die Gegenständen aufgeklebt oder aufgebügelt. Des Weiteren wurde eine Webanwendung erstellt, welche es ermöglicht, Informationen über einen Gegenstand nach dem Scannen eines Barcodes auszugeben. Mit dieser Anwendung kann auch eine Inventur durchgeführt werden. Sollte die Inventur nicht an einem Tag abgeschlossen werden, besteht die Möglichkeit, diese zu speichern und zu einem späteren Zeitpunkt fortzuführen. Wenn die Inventur beendet wird, kann eine PDF-Datei ausgegeben werden. In dieser werden jenen Gegenstände, welche entweder fehlen oder nicht in diesem Raum sein sollten, aufgelistet.

## Abbildungsverzeichnis

1	Tabellen . . . . .	1
2	Tabelle Gruppe . . . . .	1
3	Tabelle Inventar Teil 1 . . . . .	2
4	Tabelle Inventar Teil 2 . . . . .	2
5	Entwurfsansicht der Tabelle Inventar . . . . .	3
6	Tabelle Mitglieder . . . . .	3
7	Tabelle Raum . . . . .	4
8	Abfragen . . . . .	4
9	Abfrage Export . . . . .	5
10	Entwurfsansicht der Abfrage Export . . . . .	5
11	Abfrage Anstehende Prüfungen . . . . .	6
12	Abfrage Anstehende Prüfungen in der Entwurfsansicht . . . . .	6
13	Formulare . . . . .	7
14	Berichte . . . . .	7
15	Schema des Thermotransferdrucks [6] . . . . .	10
16	Drucker TSC TTP-247 . . . . .	11
17	Aufgeklappter Drucker mit Sicht auf das Trägermaterial . . . . .	11
18	Drucker mit Sicht auf Farbband . . . . .	12
19	Druckvorgang . . . . .	13
20	Benutztes Farbband . . . . .	13
21	Etiketten Standard klein in der BarTender-Ansicht . . . . .	14
22	Etiketten Standard mittel in der BarTender-Ansicht . . . . .	14
23	Patch-Etiketten mittel in der BarTender-Ansicht . . . . .	14
24	Etiketten Standard klein auf der Rolle . . . . .	16
25	Etiketten Standard mittel auf der Rolle . . . . .	16
26	Patch-Etiketten mittel auf der Rolle . . . . .	17
27	Farbband auf Spindeln . . . . .	17
28	Dropbox-Link erzeugen . . . . .	18
29	Code 128 . . . . .	21
30	Initialisierung . . . . .	24
31	Datenbank geladen . . . . .	24
32	Ergebniss des Scans . . . . .	26
33	Inventur mit falschem und richtigem Gegenstand . . . . .	28
34	PDF Ausgabe . . . . .	33
35	Lighthouse Erfolgreich . . . . .	34
36	Download Icon . . . . .	34
37	Installierte Anwendung . . . . .	34
38	Initialisierungsbildschirm . . . . .	44
39	Startbildschirm . . . . .	44
40	Scannen . . . . .	45
41	Ergebnis . . . . .	46
42	Nicht gescannt . . . . .	46

43	Falsche Gegenstände . . . . .	46
44	Ergebnis . . . . .	47

## Literatur

- [1] N. Simoneau, *MDB to JSON converter [closed]*, <https://stackoverflow.com/questions/38541672/mdb-to-json-converter>, [Letzter Zugriff: 28. März 2022].
- [2] Microsoft, *Kapitel 10: Datensätze und Datenströme*, <https://docs.microsoft.com/de-de/office/client-developer/access/desktop-database-reference/chapter-10-records-and-streams>, [Letzter Zugriff: 28. März 2022].
- [3] KenPro, *Thermotransferdruck - wie funktioniert es ?* <https://www.kenpro.de/thermotransferdruck-wie-funktioniert-es>, [Letzter Zugriff: 28. März 2022].
- [4] HellermannTyton, *Thermotransfertechnologie*, <https://www.hellermanntyton.at/kompetenzen/thermotransferdrucker>, [Letzter Zugriff: 28. März 2022].
- [5] B. Systeme, *Thermotransferdruck vs. Thermodirektdruck*, <https://www.bluhmsysteme.com/blog/thermodirektdruck-oder-thermotransferdruck/>, [Letzter Zugriff: 28. März 2022].
- [6] RAJAPACK, *Etiketten drucken mit ThermoDrucker*, <https://verpackungsnews.rajapack.at/etikettendruck-mit-thermodrucker/>, [Letzter Zugriff: 28. März 2022].
- [7] E. Identtechnik, *TSC TTP-247 Etikettendrucker (Desktop) 203dpi*, <https://www.tsc-drucker.com/Desktopdrucker/TTP-247-Serie/TSC-TTP-247-Etikettendrucker-Desktop-203dpi-oxid-1.html>, [Letzter Zugriff: 28. März 2022].
- [8] S. SCIENTIFIC, *SEAGULL SCIENTIFIC*, <https://www.seagullscientific.com/>, [Letzter Zugriff: 28. März 2022].
- [9] S. SCIENTIFIC, *Code 128*, [https://barcodeguide.seagullscientific.com/Content/Symbologies/Code\\_128.htm](https://barcodeguide.seagullscientific.com/Content/Symbologies/Code_128.htm), [Letzter Zugriff: 28. März 2022].
- [10] S. SCIENTIFIC, *Aztec Code*, [https://barcodeguide.seagullscientific.com/content/Symbologies/Aztec\\_Code.htm](https://barcodeguide.seagullscientific.com/content/Symbologies/Aztec_Code.htm), [Letzter Zugriff: 28. März 2022].
- [11] S. SCIENTIFIC, *QR Code*, [https://barcodeguide.seagullscientific.com/content/Symbologies/QR\\_Code.htm](https://barcodeguide.seagullscientific.com/content/Symbologies/QR_Code.htm), [Letzter Zugriff: 28. März 2022].
- [12] MP-FEUER, *Barcode-Etiketten Standard/ klein leer*, <https://mp-feuer.de/produkt/barcode-etiketten-standard-klein-leer/>, [Letzter Zugriff: 28. März 2022].
- [13] MP-FEUER, *Barcode-Etiketten Standard / mittel leer*, <https://mp-feuer.de/produkt/barcode-etiketten-standard-mittel-leer/>, [Letzter Zugriff: 28. März 2022].

- [14] MP-FEUER, *Barcode-Etiketten Textil / mittel leer*,  
<https://mp-feuer.de/produkt/barcode-etiketten-textil-mittel-leer/>, [Letzter Zugriff: 28. März 2022].
- [15] MP-FEUER, *Farbband Standard*,  
<https://mp-feuer.de/produkt/farbband-standard/>, [Letzter Zugriff: 28. März 2022].
- [16] MP-FEUER, *Farbband Bügel/Patch*,  
<https://mp-feuer.de/produkt/farbband-buegel-patch/>, [Letzter Zugriff: 28. März 2022].
- [17] Dropbox, <https://www.dropbox.com/de/>.
- [18] GitHub, *GitHub Pages — Websites for you and your projects*,  
<https://pages.github.com/>, [Letzter Zugriff: 25. November 2021].
- [19] Minhaz, *html5-qrcode*, <https://github.com/mebjas/html5-qrcode>, [Letzter Zugriff: 09. November 2021].
- [20] Mozilla, *Window.localStorage*, <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>, [Letzter Zugriff: 19. Februar 2022].
- [21] Mozilla, *Array.prototype.filter()*, [https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global\\_Objects/Array/filter](https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Array/filter), [Letzter Zugriff: 18. Februar 2022].
- [22] Parallax, *jsPDF*, <https://github.com/parallax/jsPDF>, [Letzter Zugriff: 12. Januar 2022].
- [23] GitHub, *GitHub Pages Documentation*, <https://docs.github.com/en/pages>, [Letzter Zugriff: 25. November 2021].
- [24] Google, *Lighthouse*,  
<https://developers.google.com/web/tools/lighthouse/>, [Letzter Zugriff: 17. Februar 2022].
- [25] Mozilla, *Progressive web apps (PWAs)*,  
[https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps), [Letzter Zugriff: 17. Februar 2022].
- [26] Mozilla, *Using Service Workers*, [https://developer.mozilla.org/en-US/docs/Web/API/Service\\_Worker\\_API/Using\\_Service\\_Workers](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers), [Letzter Zugriff: 28. März 2022].

## Anhang

# 6 Leitfaden Datenbank

## 6.1 Einleitung

Da die Datenbank die eingegebenen Daten verarbeitet und umformatiert, ist es wichtig, dass die Daten im richtigen Format eingegeben werden. Ansonsten kommt es zu Fehlern bei der Weiterverarbeitung sowie in der Handyapp.

## 6.2 Eingabe von Daten

### 6.2.1 Tabelle Gruppe und Mitglieder

Es dürfen Einträge dieser Tabellen nur gelöscht werden, wenn sie nicht in tblInventar verwendet werden. Wird ein Eintrag aus tblMitglieder/tblGruppe gelöscht, die noch in tblInventar eingetragen sind, können die betroffenen Daten nicht ausgegeben werden.

### 6.2.2 Tabelle Inventar

- invId: wird automatisch generiert
- invName: alle Zeichen erlaubt
- invRaum: Zahl; muss unbedingt in tblRaum eingetragen sein
- invGruppe: Zahl; muss unbedingt in tblGruppe eingetragen sein
- invBaujar: Zahlen
- invVerwDauer: Anzahl der Jahre
- invPrüfZeitraum: Anzahl der Jahre
- invLetzePrüfung: Datum
- invNurNummer: Zahl; maximal vierstellig
- invMitgliederId: Zahl, muss unbedingt in tblMitglieder eingetragen sein
- inv2Prüfungen: gleich wie bei invPrüfungen; nur ausfüllen, wenn invPrüfungen bereits ausgefüllt sind
- invPrüfPflichtig: wird berechnet
- invAblaufDatum: wird berechnet
- invNächstePrüfung: wird berechnet
- invInvNummer: wird berechnet

## 7 Anleitung Drucker

### 7.1 Drucken von Etiketten

1. Drucker an Strom anschließen und mittels USB mit PC Verbinden
2. Je nach Etikette die Vorlage "klein", "mittel" oder "Patch" in BarTender UltraLite öffnen
3. Doppelklick auf den Barcode → Datenquellen → Eingebettete Daten → Inventarnummer im gleichen Format eintragen und Fenster schließen
4. Drucken → Drucker: TSC TTP-247 → Drucken
5. Schritt 3 und 4 für alle Nummern wiederholen
6. Vorlage nicht speichern

### 7.2 Wechseln der Etiketten

1. Drucker vollständig aufklappen
2. Rolle mit den Etiketten entnehmen und aufrollen
3. Neue Rolle etwa 20cm abrollen
4. Etikettenrolle unter der weißen Führungsrolle durchfädeln und türkise Führung anpasssen (etwa 1 mm zwischen Band und Führung frei lassen)
5. Das Band vorne bündig mit dem Drucker abschließen, den Rest wieder aufrollen

### 7.3 Wechseln der Farbrolle

1. Kleine Abdeckung öffnen
2. Vordere Rolle (mit gebrauchtem Farbband) vorsichtig aushaken
3. Drucker ganz aufklappen
4. Hintere Farbrolle aushaken
5. Beide verbundenen Rollen entnehmen
6. Rolle mit neuem Material hinten einhaken (Matte Fläche zeigt auf Etiketten/Folie nach hinten gerichtet)
7. Farbband unter dem Heizelement durchführen
8. Zweite Rolle einhaken

## 8 Anleitung zum Inventurscanner der FF-Hohenkogl

### 8.1 Erstes Starten der Anwendung

Bei dem ersten Start der Anwendung muss die Datenbank über einen Link initialisiert werden.



Abbildung 38: Initialisierungsbildschirm

Die Abbildung 38 zeigt den Bildschirm zur Initialisierung, hier muss der beigelegte QR-Code eingescannt werden. Dieser Schritt muss nur einmal durchgeführt werden da der Link in der Anwendung gespeichert bleibt.

### 8.2 Startbildschirm

Wenn die App den Link schon geladen hat, kommt der folgende Bildschirm wie in Abbildung 39 zu sehen. Am oberen Rand der Anwendung ist eine Statusanzeige welche die Bedienung erleichtern soll.

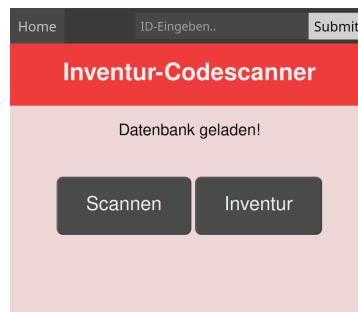


Abbildung 39: Startbildschirm

Wenn die Anzeige “Datenbank geladen!“ ausgibt war die Initialisierung erfolgreich. Es ist nun möglich einen Barcode zu scannen, einen Barcode manuell einzugeben oder eine Inventur durchzuführen.

### 8.3 Manuelle Eingabe

Um Informationen über einen Gegenstand zu bekommen, bei dem der Barcode nicht für die Anwendung lesbar ist, gibt es die Möglichkeit, die unter dem Barcode stehende Nummer im Feld ”ID-Eingeben...“ einzugeben und auf den Knopf ”Los“ zu drücken. Das Eingabefeld und der Knopf sind in Abbildung 39 in der oberen rechten Ecke zu sehen. Nun werden die Informationen des Gegenstands wie in Kapitel 8.6 erklärt angezeigt.

### 8.4 Scannen

Um Informationen über einen Gegenstand mittels Barcode zu bekommen, muss der Knopf ”Scannen“ gedrückt werden um bei dem in Abbildung 40 angezeigten Kamerafeld einen Code einzuscannen. Nun werden die Informationen des Gegenstands wie in Kapitel 8.6 erklärt angezeigt.



Abbildung 40: Scannen

### 8.5 Inventur

Eine Inventur kann durchgeführt werden, indem man den ”Inventur“ Knopf wie in Abbildung 40 zu sehen drückt. Als Erstes muss ein Barcode des Raums den Sie inventarisieren wollen eingescannt werden. Nach der Initialisierung lädt unter dem

Kamerafeld eine Liste mit allen Gegenständen, welche in dem Raum vorhanden sein sollten.



Abbildung 41: Ergebnis

### 8.5.1 Ausgabe der Inventur

Während der Inventur aktualisiert sich die Liste in Abbildung 42 und 43 fortlaufend. Wenn ein Gegenstand eingescannt wird, der in diesen Raum gehört verschwindet er von der Liste in Abbildung 42. Wenn ein Gegenstand eingescannt wird, der nicht in diesen Raum gehört, erscheint ein rotes Feld, wie in Abbildung 43 zu sehen.



Abbildung 43: Falsche Gegenstände

Abbildung 42: Nicht gescannt

### 8.5.2 Inventur Speichern und Laden

Wenn die Inventur noch nicht beendet ist, jedoch später weiter gemacht werden soll, ist es möglich den Knopf "Speichern" zu drücken, welcher die Inventurdaten für 90 Tage lang auf ihrem Gerät speichert. Mit dem Knopf "Laden" werden die Daten aus dem

Speicher geladen, wichtig zu beachten ist, dass vor dem Laden kein Barcode initialisiert werden sollte.

### 8.5.3 Inventur als PDF ausgeben

Wenn die Inventur abgeschlossen ist, kann die App auch die in Kapitel 8.5.1 erklärte Liste als PDF ausgeben indem der "PDF" Knopf gedrückt wird.

## 8.6 Ergebnis

Wenn erfolgreich eine ID in die Anwendung eingegeben wird, entweder durch einen Barcode oder die manuelle Eingabe dann erscheint der in Abbildung 44 zu sehende Bildschirm. Wenn ein Fehler aufgetreten ist, steht in den Feldern nur "Placeholder", dann sollte man versuchen, den Barcode erneut zu scannen oder die ID erneut eingeben. Um wieder auf den Startbildschirm zu gelangen, können sie den "Scannen" oder den "Home" Knopf drücken.

The screenshot shows a mobile application interface titled "Inventur-Codescanner". At the top, there are three buttons: "Home", "ID-Eingeben..", and "Submit". Below the title, the text "Scanned ID:" is followed by a text input field containing "4/117". Underneath, the text "Name:" is followed by a text input field containing "Einsatzstiefel". Then, the text "Ort:" is followed by a text input field containing "Persönlich". Finally, the text "Kommentar:" is followed by a text input field containing "Placeholder". At the bottom center is a large button labeled "Scannen!".

Abbildung 44: Ergebnis

## Projektzeitennachweis Lukas Adelmann

Datum	Tätigkeit	Stunden
13.03.2021	Besprechung-Feuerwehr	1:00
24.09.2021	Pflichtenheft	1:40
29.09.2021	Diplomarbeits-Datenbank	2:40
13.10.2021	Besprechung-Wurzinger	2:30
16.10.2021	Pflichtenheft	2:30
17.10.2021	GUI	1:40
17.10.2021	Struktogramm-Flussdiagramm	1:15
09.11.2021	Barcode-Scanner	1:40
10.11.2021	Barcode-Scanner	0:55
10.11.2021	Handy-kompatibel	1:45
10.11.2021	Einarbeiten und verbessern mit css	1:50
13.11.2021	Codescanner verbessern und Testen	01:50
13.11.2021	Android Kompatibel machen	00:30
13.11.2021	Testblatt erstellen, Codes testen	00:30
13.11.2021	in JS Datenbank einlesen	00:45
15.11.2021	Codes recherchieren und testen	01:00
17.11.2021	Besprechung	00:45
20.11.2021	Besprechung-Feuerwehr	01:40
21.11.2021	Besprechung-Feuerwehr	00:30
22.11.2021	JSON Daten einlesen und als ergebniss anzeigen	00:30
22.11.2021	Json Daten einlesen	01:30
22.11.2021	Json Daten Lesen und an neues Document übergeben	02:00
22.11.2021	Daten von Html zu Html übergeben	01:00
23.11.2021	JSON Datei einlesen und Daten übergeben	01:30
24.11.2021	Besprechung Json einlesen Single Page App	00:55
25.11.2021	Jsonp	00:30
25.11.2021	Github Pages + Testfeuerwehr codes erstellen	02:00
25.11.2021	Angular Setup	01:00
28.11.2021	verschlüsselung der Inventarnummer	01:20
02.12.2021	Daten durch Dropbox beziehen	03:00
03.12.2021	Angular einarbeiten (Heroes)	01:30
03.12.2021	Angular einarbeiten (Heroes)	01:55
03.12.2021	Einfügen des Scanners in mein Angular Projekt	02:30
03.12.2021	Funktioniert über sichere seite sonst keine kamera	01:30
05.12.2021	Framework Recherche, Egebniss: NodeJS mit Express reicht völlig aus ist nicht so OP wie angular	03:15
05.12.2021	Daten anzeigen (nodejs), fehlgeschlagen, design an grundtemplate	02:40
06.12.2021	Inventurscanner erstellen	05:30
17.12.2021	Methode für einen Export von Json überlegen	02:35

18.12.2021	Präsentation vorbereiten	01:00
19.12.2021	Vorbereiten der Zwischenpräsentation	02:00
20.12.2021	Navbar update mit suchfunktion für manuelle id	01:50
21.12.2021	Inventur Liveaktualisierung	01:10
22.12.2021	Inventur mit falsche gegenstände	00:35
22.12.2021	Inventur mit falsche gegenstände	00:50
22.12.2021	Inventur mit falsche gegenstände / funktioniert jetzt	00:40
12.01.2022	Einarbeiten in jspdf	01:00
12.01.2022	PDF von Website erstellen	00:45
13.01.2022	PDF Formatieren und Buttons zentrieren	00:35
14.01.2022	Adding cookie for database link	01:45
14.01.2022	Code Formatieren	00:20
16.01.2022	Design verbessern und Benutzerfreundlicher machen	00:45
16.01.2022	Home button und leichtere inventur	00:30
17.01.2022	erste versuche Acess Daten Formatiert extrahieren	01:10
18.01.2022	Neues ID Format und neuer Probe Datensatz, Access Datenbank Daten in richtigem format ausgeben	02:25
19.01.2022	VBA output als utf-8, man muss den ouput mit einem Object Stream machen da dann das charset eingestellt werden kann	01:00
19.01.2022	Objektorientiert Programmieren, Funktion noch nicht ganz gegeben	03:00
19.01.2022	Objektorientiert, noch nicht gelungen	01:40
19.01.2022	Klassen sind jetzt funktionsfähig, Code wurde noch Formatiert	01:25
20.01.2022	Iventurergebnisse können jetzt in einem Cookie gespeichert, und wieder aufgerufen werden	00:45
20.01.2022	Bugs entfernen und bei PDF neuen Seitenumbruch machen	00:40
20.01.2022	PDF Formatieren, Kopfzeile einfügen, Klasse für PDF erstellen	01:20
24.01.2022	Design änderungen, Button zentrieren, Tests an Benutzern, Daten wurden nach Laden nicht mehr gespeichert (behoben)	01:20
02.02.2022	Bugs fixen Drucken zu PDF umbenennen Button design und bei manueller id 0	01:20
02.02.2022	Anleitung für die Anwendung in LaTex schreiben + Bug fixes	02:10
04.02.2022	Anleitung schreiben und Nutzertests	01:35
07.02.2022	Latex Vorlage erstellen	01:20
09.02.2022	LAtex Vorlage erweitern	01:30
14.02.2022	VBA Daten als UTF-8 ausgeben, Lösung durch Datastream erreicht da dort die COdierung gewählt werden kann	00:20
14.02.2022	Latex Dokumentations Vorlage, Dokumentation Beginn Kapitel bestimmen	02:10
16.02.2022	Json im Datensatz auch Anführungszeichen ausgeben, Datenbank reparieren	01:30
17.02.2022	PWA machen mit Lighthouse privates repository	02:35

17.02.2022	PWA auf main app übertragen	01:40
18.02.2022	Performance verbessern da diese bei der PW Anwendung nicht sehr gut ist	01:40
18.02.2022	Performance wurde verbessert indem mit js Json.filter gearbeitet wurde jedoch ist die methode zu speicherung der Inventur nicht so möglich wie sie bisher umgesetzt wurde da der cookie zu klein ist.	02:25
19.02.2022	Inventur wird nun im LocalStorage gespeichert... funktioniert super	01:10
24.02.2022	Alles kann nun auch nur durch Eingabe gemacht werden, alles ins Hauptrepository übertragen	02:20
24.02.2022	Dokumentieren beginnen	01:00
28.02.2022	Dokumentation Code-Scanner	00:50
28.02.2022	Dokumentation Code-Scanner, Bug fixen bei manueller link eingabe	02:00
28.02.2022	Dokumentation Scann-Ergebniss	00:30
04.03.2022	Dokumentation Datenverarbeitung	01:00
11.03.2022	Dokumentation Initialisierung	01:00
13.03.2022	Dokumentation, neue Gliederung der Kapitel	01:30
14.03.2022	Dokumentation, Scanner/Ausgabe von Daten	01:20
17.03.2022	Dokumentation Design	00:45
18.03.2022	Dokumentation, Inventur - einfügen von neuen Subkapiteln	01:00
24.03.2022	Dokumentation, PDF-Speichern	01:00
25.03.2022	Service Worker reparieren Dokumentation, Serviceworker	03:30
25.03.2022	Dokumentation und korrektur	01:45
27.03.2022	Dokumentation korrigieren	01:00
28.03.2022	Reparieren des Service Worker, Dokumentation	01:30
28.03.2022	Dokumentation, Kurzbeschreibung Abstract Vorwort, Zitieren mit Biber	03:40
28.03.2022	Dokumentation, Github Pages kaputt	04:00
28.03.2022	Dokumentation, Github pages	01:20
29.03.2022	Dokumentation, Stundentabelle Anhang	03:00
31.03.2022	Dokumentation	03:20
31.03.2022	Dokumentation	01:00
31.03.2022	Dokumentation	01:00
31.03.2022	Dokumentation	01:00
<b>Gesamtstunden:</b>		153:50

## Projektzeitennachweis Simon Flicker

Datum	Tätigkeit	Stunden
13.03.2021	Besprechung-Feuerwehr Bekleidung	01:00
13.03.2021	Besprechung-Feuerwehr Werkzeug	01:00
24.09.2021	Pflichtenheft	01:40
29.09.2021	Diplomarbeits-Datenbank	02:40
13.10.2021	Besprechung-Wurzinger	02:30
16.10.2021	Pflichtenheft	02:30
16.10.2021	Blockschaltbild/Detaillschaltbild/Komponenten Besprechen	03:30
17.10.2021	Bestellliste	02:00
21.10.2021	Recherche Drucker/Druckverfahren	01:00
06.11.2021	Recherche Thermotransferdruck/Etiketten	02:00
06.11.2021	Besprechung Schöberl aktueller Stand, Datenbank	01:00
12.11.2021	Organisation Probeetiketten MP Feuer	00:30
19.11.2021	Test Etiketten normal	01:00
20.11.2021	Test Textiletiketten mit Carmen und Schöberl	01:30
20.11.2021	Kostenrechnung/Druckersuche	00:30
21.11.2021	Besprechung Etiketten normal Hödl	00:30
24.11.2021	Besprechung WU, HBI	00:55
24.11.2021	Bestellungen bei MP-Feuer	01:00
28.11.2021	Recherche/Vergleich Handy	01:00
29.11.2021	Bestellung Handy+Hülle	01:00
02.12.2021	Einrichten Handy	01:30
02.12.2021	Recherche Ms Access zu JSON exportieren	02:00
03.12.2021	Recherche Ms Access zu JSON exportieren	02:30
03.12.2021	1. Inbetriebnahme Drucker&Tests	00:30
05.12.2021	Recherche Ms Access zu JSON exportieren	00:30
06.12.2021	Einarbeitung in Access-Datenbank	01:00
16.12.2021	Präsentation/A3 Seite	01:00
16.12.2021	Versuch Acces to JSON mit „AccessToFile“	00:30
17.12.2021	Einarbeitung Datenbank, Erstellen Abfrage, + Versuch mit AccessToFile (erfolglos)	01:30
17.12.2021	Versuch Acces to JSON	01:15
18.12.2021	Einarbeitung VBA	02:30
19.12.2021	Vorbereitung Präsentation/BarTender	03:30
29.12.2021	Recherche Druckersoftware	01:30
30.12.2021	Erstellen von Etikettenvorlagen für alle Größen	00:30
03.01.2022	Versuche Export	01:00
05.01.2022	Recherche Export	02:15
08.01.2022	Vorbereitung für Textildruck	01:45
10.01.2022	1. Anwendung der Textil-Barcodes (neues MRAS-Gewand) mit Carmen	01:30

12.01.2022	Besprechung mit WU: Erklärung Exportvorgang	01:00
12.02.2022	Export in neuem Projekt (Test)	01:00
15.01.2022	Formatieren JSON-String	01:00
16.01.2022	Abfrage bearbeiten	00:30
16.01.2022	führende Null/Zusammensetzen von Spalten (Null: Entwurfsansicht-Format)	01:00
16.01.2022	formatieren JSON	00:30
17.01.2022	Formatieren der Inventarnummer	01:10
17.01.2022	Versuch Export mit führender Null/Überlegung Struktur von JSON	00:30
18.01.2022	VBA-Code angepasst (Formatierung des String)	02:30
19.01.2022	Entfernen der Leerzeilen in der JSON; in DB: Abfrage unvollständig	00:50
19.01.2022	vba filestream (für Sonderzeichen/Umlaute)	01:30
01.02.2022	Besprechung mit Schöberl bezüglich Access-Datenbank, Prüfintervalle, leere Nummern, Problem mit Synchronisierung mit FF-Laptop	00:45
02.02.2022	Versuch Ausgabe Sonderzeichen filestream	00:50
02.02.2022	Erweiterung Export-Code: Ersetzen von Sonderzeichen durch ae	00:40
03.02.2022	Import von Stammdatenbank mit allen Formularen, ... & Code in neue Datenbank	01:10
05.02.2022	Formatieren tblInventar in Prototyp-Datenbank	01:30
08.02.2022	Ausdrucken von Barcodes	00:30
10.02.2022	Datumsfunktionen Tabelle Access	01:55
11.02.2022	Abfrage mit Datumsfunktionen (anstehende Prüfungen)	00:30
11.02.2022	DB: beheben von falschen invNummern	01:30
12.02.2022	Prototyp-Datenbank: Berechnen invNummer, alle Prüfdaten, Verweildauer	02:00
14.02.2022	Ausgabe-Abfrage bearbeitet; Github-Konto FF einrichten	02:25
17.02.2022	Access: Schaltfläche beim Schließen, Präsentation für Wehrversammlung	03:00
18.02.2022	Einarbeitung LaTex; Anleitung Datenbank	01:00
18.02.2022	Anleitung Access und Drucker; Formatierung Access-Datenbank	02:15
18.02.2022	toJson bearbeiten (für bessere Performance am Handy)	00:35
19.02.2022	zusätzliche Rolle für Drucker	00:45
19.02.2022	Übertragen auf Haupt-Datenbank in Dropbox	02:10
23.02.2022	Erstellen Abfrage mit anstehenden Prüfungen; Formulare	01:15
24.02.2022	Formular Main; anstehende Prüfungen eingebunden (Prototyp-DB)	01:05
24.02.2022	Formulare/Berichte bearbeitet	00:30
25.02.2022	Anleitung Drucker	01:40

26.02.2022	Überarbeitung von Etikettenvorlagen & Testdrucken	01:55
28.02.2022	Dokumentation, Gespräch Schöberl über Datenbank	02:00
01.03.2022	Vorbereitung Präsentation Wehrversammlung	00:45
03.03.2022	Vorbereitung Präsentation Wehrversammlung	00:45
05.03.2022	Fertigstellung Access Formulare, Berichte	01:10
06.03.2022	Dokumentation Datenbank	01:35
07.03.2022	Dokumentation Druck	00:30
07.03.2022	Dokumentation Thermotransferdruck; Besprechnung Schöberl Datenbank(nicht mehr benötige Abfragen, Formulare entfernen)	01:50
07.03.2022	Dokumentation Etikettendesign	00:30
08.03.2022	Access frmMain berbeiten	01:10
08.03.2022	Abfragen für Bericht erstellen	00:30
08.03.2022	Berichte für Prüfungen/Ablaufdatum erstellen&verlinken	00:45
12.03.2022	Dokumentation Etiketten	01:00
12.03.2022	Dokumentation Etiketten, Datenbank , neues Inhaltsverzeichnis	01:25
15.03.2022	Dokumentation DB, Screenshots	01:15
15.03.2022	Dokumentation in Latex übertragen, Bilder bearbeiten	01:10
16.03.2022	Dokumentation allgemein, Bilder	01:30
18.03.2022	Dokumentation Bilder&Absätze formatieren, Design der Etiketten, Codes, Farbfolien	03:50
21.03.2022	Dokumentation Design der Etiketten	01:00
21.03.2022	Dokumentation Einleitung, Vorwort, ...	01:50
25.03.2022	Dokumentation allgemein, Bilder anpassen	03:30
25.03.2022	Dokumentatio Drucker, Etiketten, Druckvorgang	01:30
26.03.2022	Dokumentation Drucker und Druckvorgang	01:50
26.03.2022	Dokumentation VBA-Code	00:40
27.03.2022	Dokumentation VBA, Bilder, Druckschema	01:50
28.03.2022	Dokumentation allgemein, VBA	00:50
28.03.2022	Dokumentation, Einleitung, Quellen	02:30
28.03.2022	Dokumentation Literaturverzeichnis	03:45
28.03.2022	Dokumentation Literaturverzeichnis, allgemein	01:00
29.03.2022	Dokumentation Anhänge	01:00
30.03.2022	Dokumentation Anhänge	01:50
31.03.2022	Dokumentation Allgemein	02:45
Gesamtstunden:		144:25

