

ENTWICKLUNG EINER FORMULARANWENDUNG MIT KOMPATIBILITÄTSVALIDIERUNG DER EINFACH- UND MEHRFACHAUSWAHL-EINGABEFELDER

Vorgelegt von:

Alexander Johr

Meine Adresse

Erstprüfer: Prof. Jürgen Singer Ph.D.
Zweitprüfer: Prof. Daniel Ackermann
Datum: 02.11.2020

THEMA UND AUFGABENSTELLUNG DER MASTERARBEIT
MA AI 29/2021

FÜR HERRN ALEXANDER JOHR

ENTWICKLUNG EINER FORMULARANWENDUNG MIT
KOMPATIBILITÄTSVALIDIERUNG DER EINFACH- UND
MEHRFACHAUSWAHL-EINGABEFELDER

Das Thünen-Institut für Ländliche Räume wertet Daten zu Maßnahmen auf landwirtschaftlich genutzten Flächen aus. Dafür müssen entsprechende Maßnahmen bundesweit mit Zeitbezug auswertbar sein und mit Attributen versehen werden. Um die Eingabe für die Wissenschaftler des Instituts zu beschleunigen und um fehlerhafte Eingaben zu minimieren, soll eine spezielle Formularanwendung entwickelt werden. Neben herkömmlichen Freitextfeldern beinhaltet das gewünschte Formular zum Großteil Eingabefelder für Einfach- und Mehrfachauswahl. Je nach Feld kann die Anzahl der Auswahloptionen mitunter zahlreich sein. Dem Nutzer sollen daher nur solche Auswahloptionen angeboten werden, die zusammen mit der zuvor getroffenen Auswahl sinnvoll sind.

Im Wesentlichen ergibt sich die Kompatibilität der Auswahloptionen aus der Bedingung, dass für dasselbe oder ein anderes Eingabefeld eine Auswahlmöglichkeit gewählt bzw. nicht gewählt wurde. Diese Bedingungen müssen durch Konjunktion und Disjunktion verknüpft werden können. In Sonderfällen muss ein Formularfeld jedoch auch die Konfiguration einer vom Standard abweichenden Bedingung ermöglichen. Wird dennoch versucht, eine deaktivierte Option zu selektieren, wäre eine Anzeige der inkompatiblen sowie der stattdessen notwendigen Auswahl ideal.

Die primäre Zielplattform der Anwendung ist das Desktop-Betriebssystem Microsoft Windows 10. Idealerweise ist die Formularanwendung auch auf weiteren Desktop-Plattformen sowie mobilen Endgeräten wie Android- und iOS-Smartphones und -Tablets lauffähig. Die Serialisierung der eingegebenen Daten genügt dem Institut zunächst als Ablage einer lokalen Datei im JSON-Format.

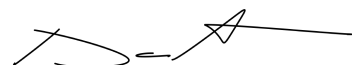
Die Masterarbeit umfasst folgende Teilaufgaben:

- Analyse der Anforderungen an die Formularanwendung
- Evaluation der angemessenen Technologie für die Implementierung
- Entwurf und Umsetzung der Übersichts- und Eingabeoberfläche
- Konzeption und Implementierung der Validierung der Eingabefelder
- Entwicklung von automatisierten Testfällen zur Qualitätskontrolle
- Bewertung der Implementierung und Vergleich mit den Wunschkriterien

Digital unterschrieben von
Juergen K. Singer
o= Hochschule Harz,
Hochschule fuer
angewandte
Wissenschaften, l=
Wernigerode
Datum: 2021.03.23 12:30:
26 MEZ



Prof. Jürgen Singer Ph.D.
1. Prüfer



Prof. Daniel Ackermann
2. Prüfer

Inhaltsverzeichnis

Listingsverzeichnis	9
1 Anforderungen	12
I Einleitung	13
2 Problemstellung	13
3 Grundlagen	13
3.1 Flutter	13
3.2 Dart	14
3.2.1 AOT und JIT	15
3.2.2 Set und Map Literale	15
3.2.3 Typen ohne NULL-Zulässigkeit	17
3.2.4 Typen mit Null-Zulässigkeit	17
3.2.5 Asynchrone Programmierung	19
4 Technologie Auswahl	22
4.1 Trendanalyse	22
4.1.1 Frameworks mit geringer Relevanz	23
4.1.2 Frameworks mit sinkender Relevanz	24
4.1.3 Frameworks mit steigender Relevanz	25
4.2 Vergleich React Native und Flutter	26
4.2.1 Vergleich zweier minimaler Beispiele für Formulare und Validierung .	26
4.2.2 Automatisiertes Testen	27
II Implementierung	30
5 Schritt 1 - Formular in Grundstruktur erstellen	30
5.1 Auswahloptionen hinzufügen	31
5.2 Serialisierung einer Maßnahme	35
5.3 Test der Serialisierung einer Maßnahme	38
5.4 Serialisierung der Maßnahmenliste	40

5.5	Test der Serialisierung der Maßnahmenliste	41
5.6	Der Haupteinstiegspunkt	42
5.7	Der Service für den Applikations übergreifenden Zustand	44
5.8	Speichern der Maßnahmen in eine Json-Datei	45
5.9	Abhängigkeit zum Verwalten der Maßnahmen	46
5.10	Übersichtsbildschirm der Maßnahmen	48
5.10.1	Auflistung der Maßnahmen im Übersichtsbildschirm	50
5.11	Widget MassnahmenTable	52
5.12	Das View Model	54
5.13	Eingabeformular	56
5.14	Widget SelectionCard	59
5.15	Integration Test zum Test der Oberfläche	63
5.16	Schritt 2	66
6	HIER EINFÜGEN Status Choice	66
6.1	Schritt 3	70
6.2	Schritt 4	74
6.3	Schritt 5	81
6.4	Schritt 6	85
6.5	Schritt 7	88
III	Anhang	99
A	Technologiewahl Anhang	100
A.1	Stimmen verwendeter Frameworks	100
A.2	Stimmen gewünschter Frameworks	100
B	Vergleich React Native und Flutter Anhang	100
C	Schritt 1 Anhang	108
D	Schritt 2 Anhang	109
Literatur		113

Abbildungsverzeichnis

1	Stimmen der Stack Overflow Umfrage von 2013 bis 2020	23
2	Suchinteresse der Frameworks mit geringer Relevanz	24
3	Stimmen für Cordova und PhoneGap 2013 bis 2020	24
4	Stimmen für Xamarin und Cordova	25
5	Suchinteresse sinkende und steigende Relevanz	25
6	Stimmen für React Native und Flutter	26
7	Schritt 1 Übersicht	30
8	Schritt 1 Eingabemaske	30
9	Schritt 1 Selektions-Bildschirm für Status	31
10	Stimmen verwendeter Frameworks	100
11	Stimmen gewünschter Frameworks	101

Listingsverzeichnis

1	Ein Set	15
2	Collection-for in einer Menge	16
3	Collection-for in einer Hashtabelle	16
4	Collection-if in einer Liste	17
5	Collection-if in einer Liste	17
6	Collection-if in einer Liste	18
7	Collection-if in einer Liste	18
8	Collection-if in einer Liste	19
9	Collection-if in einer Liste	19
10	Collection-if in einer Liste	20
11	Collection-if in einer Liste	20
12	Collection-if in einer Liste	21
13	Collection-if in einer Liste	21
14	Collection-if in einer Liste	21
15	Schritt 1 Die Klasse LetzterStatus	31
16	Schritt 1 Die Klasse Choice	32
17	Schritt 1 Die Menge letzterStatusChoices	32
18	Schritt 1 Die Klasse Choices	33
19	built_value Live Template	35
20	Schritt 1 Der Werte-Typ Massnahme	36
21	Schritt 1 Der Werte-Typ Identifikatoren	37
22	Schritt 1 Der Werte-Typ LetzteBearbeitung	37
23	Schritt 1 Der Serialisierer für Massnahme und Storage	38
24	Schritt 1 Serialisierung einer Maßnahme Unittest	38
25	Schritt 1 Deserialisierung einer Maßnahme Unittest	39
26	Schritt 1 Instanzvariable <code>letzteBearbeitung</code> gibt einen <code>LetzteBearbeitungBuilder</code> zurück	40
27	Schritt 1 Der Werte-Typ Storage	41
28	Schritt 1 Ein automatisierter Testfall überprüft	41
29	Schritt 1 Instanzvariable <code>massnahmen</code> gibt einen <code>SetBuilder</code> zurück	41
30	Schritt 1 Der Haupteinstiegspunkt	43
31	Schritt 1 Der Service AppState	44
32	Schritt 1 Die Klasse Massnahmen.JsonFile	46
33	Schritt 1 Die Klasse MassnahmenModel	47
34	Schritt 1 Die Struktur der Klasse MassnahmenMasterScreen	49
35	Schritt 1 Die Ausgabe der Maßnahmen	51
36	Schritt 1 Die Klasse MassnahmenTable	53
37	Schritt 1 Die Klasse MassnahmenFormViewModel	55
38	Schritt 1 Klasse MassnahmenDetailScreen Struktur	57
39	Schritt 1 Die Ausgabe der Formularfelder	58
40	Schritt 1 Die Funktion createMassnahmenTitelTextFormField	58
41	Schritt 1 Die Funktion saveRecordAndGoBackToOverviewScreen	58
42	Schritt 1 Die Klasse SelectionCard	59
43	Schritt 1 Die Build Methode der SelectionCard	60
44	Schritt 1 Die Funktion createMultipleChoiceSelectionScreen	61
45	Schritt 1 Der Integration Test Driver	63
46	Schritt 1 Initialisierung des Integrations Tests	63
47	Schritt 1 Initialisierung des Widgets für den Integrations Tests	63
48	Schritt 1 Die Hilfsmethode tabSelectionCard	64
49	Schritt 1 Der Button zum Kreieren einer Maßnahme wird ausgelöst	65
50	Schritt 1 Der letzte Status wird ausgewählt	65
51	Schritt 1 Der Maßnahmentitel wird eingegeben	65

52	Schritt 1 Der Button zum Speichern wird ausgelöst	65
53	Schritt 1 Der Button zum Speichern wird ausgelöst	65
54	Schritt 2 Der Integrationstest klickt 5 weitere Karten	66
55	Schritt 2 Die Klasse FoerderklasseChoice	66
56	Schritt 2 massnahmenCharakteristika wird Massnahme hinzugefügt	67
57	Schritt 2 Der Werte-Typ Massnahmencharakteristika	67
58	Schritt 2 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	67
59	Schritt 2 Maßnahmencharakteristika werden dem ViewModel hinzugefügt .	68
60	Schritt 2 Maßnahmencharakteristika werden dem Tabellenkopf hinzugefügt .	68
61	Schritt 2 Maßnahmencharakteristika werden dem Tabellenkörper hinzugefügt	69
62	Schritt 3 errorText wird der SelectionCard hinzugefügt	70
63	Schritt 3 errorText wird ausgegeben	70
64	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	71
65	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	71
66	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	71
67	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	72
68	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	72
69	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	72
70	Schritt 3 Die Maßnahmencharakteristika Selektionskarten werden ergänzt .	73
71	Schritt 4 XXXX	74
72	Schritt 4 XXXX	74
73	Schritt 4 XXXX	75
74	Schritt 4 XXXX	75
75	Schritt 4 XXXX	75
76	Schritt 4 XXXX	75
77	Schritt 4 XXXX	76
78	Schritt 4 Die Ausgabe der Formularfelder	76
79	Schritt 4 Die Ausgabe der Formularfelder	76
80	Schritt 4 Die Ausgabe der Formularfelder	76
81	Schritt 4 XXXXX	77
82	Schritt 4 XXXXX	77
83	Schritt 4 XXXXX	77
84	Schritt 4 XXXXX	78
85	Schritt 4 XXXXX	79
86	Schritt 4 XXXXX	80
87	Schritt 5 XXXX	81
88	Schritt 5 XXXX	81
89	Schritt 5 XXXX	81
90	Schritt 5 XXXX	82
91	Schritt 5 XXXX	82
92	Schritt 5 XXXX	83
93	Schritt 5 XXXX	83
94	Schritt 5 XXXX	83
95	Schritt 5 XXXX	83
96	Schritt 5 XXXX	83
97	Schritt 5 XXXX	83
98	Schritt 5 XXXXX	83
99	Schritt 5 XXXXX	84
100	Schritt 5 XXXX	84
101	Schritt 5 XXXX	84
102	Schritt 6 XXXX	85
103	Schritt 6 XXXX	85
104	Schritt 6 XXXX	85

105	Schritt 6 XXXX	86
106	Schritt 6 XXXX	86
107	Schritt 6 XXXX	86
108	Schritt 6 XXXX	86
109	Schritt 6 XXXX	86
110	Schritt 6 XXXX	87
111	Schritt 6 XXXX	87
112	Schritt 6 XXXX	87
113	Schritt 7 XXXX	88
114	Schritt 7 XXXX	88
115	Schritt 7 XXXXX	88
116	Schritt 7 XXXX	89
117	Schritt 7 XXXX	89
118	Schritt 7 XXXX	90
119	Schritt 7 XXXX	90
120	Schritt 7 XXXX	90
121	Schritt 7 XXXX	90
122	Schritt 7 XXXX	91
123	Schritt 7 XXXX	91
124	Schritt 7 XXXX	91
125	Schritt 7 XXXX	91
126	Schritt 7 XXXX	92
127	Schritt 7 XXXX	92
128	Schritt 7 XXXX	92
129	Schritt 7 XXXX	93
130	Schritt 7 XXXXX	93
131	Schritt 7 XXXXX	93
132	Schritt 7 XXXXX	94
133	Schritt 7 XXXX	94
134	Schritt 7 XXXX	95
135	Schritt 7 XXXX	95
136	Schritt 7 XXXX	96
137	built_value Live Template	97
138	built_value Live Template	97
139	built_value Live Template	97
140	built_value Live Template	98
141		102
142		103
143		104
144		105
145		105
146		106
147		107
148	Schritt 1 Ein automatisierter Testfall überprüft	108
149	Schritt 2 Die Klasse FoerderklasseChoice	110
150	Schritt 2 Die Menge foerderklasseChoices	111
151	Schritt 2 Die Klasse KategorieChoice	112

1 Anforderungen

Dieses Kapitel behandelt die Anforderungen

- Performance: Hohe Anzahl Eingabefelder
- nested formulars

Wunsch

- Alle Komponenten wiederverwendbar wie etwa Selection Caard nicht nur für Choices
-
- Strategy Entwurfsmuster für compute choice von viewModelType weil viewmodel nötig ist für nested formular
-

Teil I

Einleitung

2 Problemstellung

Das primäre Problem der Formular-Anwendung ist, dass sich die Auswahlfelder untereinander beeinflussen. Wird eine Option in einem Auswahlfeld selektiert, so werden die möglichen Auswahlfelder von potenziell jedem anderen Auswahlfeld dadurch manipuliert. Es muss eine Möglichkeit gefunden werden, die Abhängigkeiten in einer einfachen Art und Weise für jede Auswahloption zu hinterlegen und bei Bedarf abzurufen.

Das sekundäre Problem, welches sich vom primären Problem ableiten lässt, ist die Laufzeit-Geschwindigkeit. Wenn die Auswahl in einem Auswahlfeld die Auswahlmöglichkeiten in potenziell allen anderen Auswahlfeldern manipuliert, so könnte dies zu einer hohen Last beim erneuten Zeichnen der Oberfläche zur Folge haben. Wann immer der Nutzer eine Selektion tätigt, müsste das gesamte Formular neu gezeichnet werden, um sicherzustellen, dass invalide Auswahloptionen gekennzeichnet werden. Bei einem Formular mit wenigen Auswahlfeldern wäre das kein Problem. Doch die nötigen Auswahlfelder für das eintragen von Maßnahmen des Europäische Landwirtschaftsfonds für die Entwicklung des ländlichen Raums (ELER) sind zahlreich. Ein automatisierter Integrationstest, welcher im Formular Daten einer beispielhaften Maßnahme einträgt, zählt zum Zeitpunkt der Erstellung dieser Arbeit bereits 58 aufgerufene Auswahlfelder und 107 darin selektierte Auswahloptionen. Das bedeutet, dass bei jedem dieser 107 Selektionen die 58 Auswahlfelder und all ihre Kinder neu gezeichnet werden müssten. Es könnten also Wartezeiten nach jedem Auswählen einer Option entstehen. Das Formular soll in Zukunft zudem noch erweitert und auch für die Eingabe ganz anderer Datensätze mit potenziell noch mehr Auswahlfeldern eingesetzt werden. Es ist also erforderlich, dass ein Mechanismus gefunden wird, der nur die Elemente neu zeichnet, die sich wirklich ändern.

Für Form wurde Flutter gewählt

3 Grundlagen

Für die Formular Anwendung wurde die Programmiersprache Dart und das Oberflächen Framework Flutter gewählt. Kapitel Kapitel einfügen erläutert die Entscheidungs-Grundlage dafür.

Nachfolgend soll auf Grundlagen der beiden Technologien eingegangen werden.

3.1 Flutter

Flutter ist ein Framework zur Entwicklung von Oberflächen von Google. Es unterstützt eine breite Anzahl an Ziel-System. Dazu gehören:

- Desktop:¹
 - Windows:

¹Vgl. Google LLC, *Desktop support for Flutter*.

- * Win32,
- * Universal Windows Platform,
- macOS,
- Linux,
- Mobile Endgeräte²:
 - Android,
 - iOS,
- und das Web³.

Flutter ist inspiriert durch das Web-Framework react und deren Oberflächenelemente, die Components genannt werden⁴. Die visuellen Oberflächen-Elemente in Flutter werden dagegen Widgets genannt. „react“ „Components“ verfügen über einen Zustand – „State“ genannt – der bei Veränderung das neu zeichnen der visuellen Repräsentation bewirkt. Flutter unterscheidet allerdings zwischen zwei Arten von Widgets: denen, die einen Zustand pflegen – den „Stateful Widgets“ – und solchen, die keinen Zustand haben – den „Stateless Widgets“.

„Stateful Widgets“ pflegen einen Zustand, der mittels der Methode `setState` gesetzt werden kann. Bei Aufrufen der Methode wird das gesamte Widget neu gezeichnet. Der Zustand selbst ist dabei im visuellen Baum als Vater der visuellen Elemente des Widgets verankert und bleibt erhalten, während die dazugehörigen Oberflächenelemente ausgetauscht werden.

„Stateless Widgets“ haben dagegen keinen solchen Mechanismus. Wie alle Widgets werden sie neu gezeichnet, wenn es durch das Framework angeordnet wurde. Das kann unter anderem der Fall sein, wenn das Widget zum ersten Mal in der Oberfläche auftaucht, oder das Vater-Element und damit alle Kinder Elemente neu gezeichnet werden.

„StatefulWidgets“ sind nur eine von vielen Möglichkeiten den Zustand des Programms zu verwalten. Die Formular-Anwendung verwendet ausschließlich `StatelessWidget`s, da die Verwaltung des Zustands über das sogenannte BloC Pattern umgesetzt ist. Mehr dazu im Kapitel **Kapitel einfügen**.

3.2 Dart

Flutter-Anwendungen werden in der Programmiersprache Dart geschrieben. Nachfolgend soll auf eine Reihe von Besonderheiten von Dart im Vergleich zu anderen objektorientierten Programmiersprachen eingegangen werden.

Dart ist eine Hochsprache, die hauptsächlich für die Entwicklung von Oberflächen entwickelt wurde, sich jedoch ebenso dazu eignet, Programme für das Back-End zu entwickeln.

Ein Hauptaspekt bei dem Design der Sprache ist die Produktivität des Entwicklers. Mechanismen wie das „hot reload“ verkürzen die Entwicklungszyklen erheblich. Das „hot reload“ ermöglicht es, während eine Anwendung im Debug-Modus ausgeführt wird, Änderungen an dessen Quellcode vorzunehmen. Daraufhin werden nur die Teile der laufenden Applikation aktualisiert, die tatsächlich verändert wurden. Währenddessen bleibt die Anwendung in der gleichen Ansicht, anstatt zum Hauptbildschirm zurückgesetzt werden, von der aus der Entwickler erneut zur gewünschten Ansicht zurücknavigieren müsste.

²Vgl. Google LLC, *Flutter - Beautiful native apps in record time*.

³Vgl. Google LLC, *Web support for Flutter*.

⁴Vgl. Google LLC, *Flutter - Introduction to widgets*.

3.2.1 AOT und JIT

Nicht nur für die reibungslose Entwicklung sondern auch für das Laufzeitverhalten der finalen Applikation wurde die Sprache optimiert. Für die Ziel-Architekturen ARM32, ARM64 und x86_64 wird Dart in Maschinencode kompiliert. `url {https://dart.dev/overview##native-platform}`

Dementsprechend kommt während der Entwicklung eine virtuelle Maschine - die Dart VM - geschickt über Just-in-time-Kompilierung (JIT) zum Einsatz. Für die Kompilierung in Maschinencode wird dagegen Ahead-of-time-Kompilierung (AOT) eingesetzt.

tree shaking Für die Minimierung der Dateigröße des resultierenden Kompilats wird das sogenannte „tree shaking“ eingesetzt. Das Hauptprogramm importiert über das Schlüsselwort `import` Funktionalitäten aus weiteren .dart-Dateien oder sogar ganzen Bibliotheken. Diese Dateien importieren wieder Weitere. Dadurch wird ein Baum aufgespannt. Das „tree shaking“ identifiziert, welche Funktionalitäten tatsächlich vom Programm verwendet werden und welche nicht. Dies bringt aber eine wichtige Einschränkung mit sich. Die Metaprogrammierung (der Zugriff auf sprachinterne Eigenschaften, wie etwa Klassen und ihre Attribute) ist damit stark eingeschränkt.

Meta-Programmierung Bei der Kompilierung werden die Original-Bezeichner durch Symbole ersetzt, welche minimalen Speicher-Bedarf haben. Aber nicht nur das, denn durch das „tree shaking“ werden auch etwaige Eigenschaften und Funktionalitäten entfernt, die nicht verwendet werden. Die sogenannte „Reflexion“ oder „Introspektion“ versucht auf solche Meta-Informationen während der Laufzeit zuzugreifen. Da die Eigenschaften aber nicht mehr verfügbar sind, ist „Reflexion“ nicht anwendbar. Dart greift daher auf eine andere Variante der Meta-Programmierung zurück: die Quellcode Generierung.

Quellcode-Generierung Das Package „source_gen“ erlaubt das Auslesen der Meta-Informationen und ermöglicht das Generieren von Quellcode, der von diesen Eigenschaften abgeleitet werden kann. So verwendet beispielsweise das Package „built_value“ die Quellcode-Generierung. Zunächst werden Eigenschaften wie Klassennamen, Instanzvariablen mit ihren Bezeichnern und Datentypen gelesen. Die Eigenschaften können dann genutzt werden, um unveränderliche Werte-Typen und dazugehörige sogenannte „Builder“-Objekte des Erbauer-Entwurfsmusters, sowie Funktionen zum Serialisieren und Deserialisieren von Objekten zu generieren. `Referenzen`

3.2.2 Set und Map Literale

Dart erlaubt es Listen (`List`), Mengen (`Set`) und Hashtabellen (`Map`) als sogenannte Literale zu deklarieren. Ein Literal ist die textuelle Repräsentation eines Wertes eines speziellen Datentyps. Beispielsweise ist `"Text"` ein String-Literal für eine Zeichenkette mit den Elementen „T“, „e“, „x“, „t“. So ist auch `{ "Text" }` ein Literal für eine Menge (`Set`). Eine Menge mit den gleichen Werten könnte genauso auch wie in Listing 1 erstellt werden.

```
1 var menge = Set();  
2 menge.add("Text");
```

Listing 1: Ein Set, Quelle: Eigenes Listing

Es entfällt also die Instanziierung einer Liste, einer Menge oder einer Hashtabelle über den Klassennamen und der darauffolgenden Zuweisung der einzelnen Werte. Stattdessen startet das `Set` und `Map` Literal mit einer öffnenden geschweiften Klammer und endet mit einer schließenden geschweiften Klammer. Innerhalb der Klammern werden die Werte im Fall eines Sets mit `,` getrennt nacheinander aufgeführt (`{1,2}`). Im Fall einer Map werden der Schlüssel und der Wert durch einen `:` voneinander getrennt und die Schlüssel-Wertepaare wiederum durch `,` getrennt nacheinander aufgelistet (`{1: "erster Wert", 2: "zweiter Wert"}`). Eine Liste wiederum wird mit eckigen Klammern geöffnet und geschlossen. Die Werte werden erneut mit `,` getrennt voneinander angegeben (`[1,2]`).

Collection for Dart erlaubt es Schleifen innerhalb von Listen-, Mengen- und Hashtabellen-Literalen zu verwenden. Dabei darf die Schleife jedoch keinen Schleifen-Körper besitzen. Lediglich der Schleifen-Kopf wird dazu im Literal geschrieben. Darauf folgt der Wert, der bei jedem Schleifendurchlauf hinzugefügt werden soll. Dabei kann der Wert von der Schleifenvariable genutzt oder davon abgeleitet werden. Listing 2 geht beispielsweise durch die Liste von der Temperatur-Angaben 97.7, 105.8, die in Fahrenheit gelistet sind. Für jeden Schleifendurchlauf wird die Schleifen-Variable `f` mit der entsprechenden Formel in Grad Celsius umgewandelt. Das Ergebnis ist somit äquivalent mit dem `Set`-Literal `{36.5, 38.5, 41}`.

```
1 var gradCelsiusTemperaturen = {  
2   for (var f in [97.7, 101.3, 105.8])  
3     (f - 32) * 5 / 9  
4 };
```

Listing 2: Collection-for in einer Menge, Quelle: Eigenes Listing

Gleiches gilt für Hashtabellen. Hierbei wird ein Schlüssel-Werte-Paar übergeben. Links vom einem `:` ist der Schlüssel und rechts davon der Wert. In Listing 3 wird durch die gleiche Liste von Temperaturen in Fahrenheit iteriert. Für jede Schleifenvariable `f` wird für das resultierende Schlüsselwörterpaar das Ergebnis in Grad Celsius als Schlüssel und das Ergebnis als Wert eingetragen. Das Ergebnis von `celsiusUndFahrenheit` ist dementsprechend eine `Map` mit dem Wert: `{36.5: 97.7, 38.5: 101.3, 41: 105.8}`

```
1 var celsiusUndFahrenheit = {  
2   for (var f in [97.7, 101.3, 105.8])  
3     (f - 32) * 5 / 9 : f  
4 };
```

Listing 3: Collection-for in einer Hashtabelle, Quelle: Eigenes Listing

Collection-if Neben dem Collection-for ist auch die Nutzung von Fallunterscheidungen in Kollektionen erlaubt. Vor dem Wert, der in die Kollektion aufgenommen werden soll oder nicht, kann das Schlüsselwort `if` mit einer darauffolgenden Bedingung in Klammern gesetzt werden. Listing 4 iteriert durch eine Anzahl von Temperaturen in Grad Celsius. Nur in dem Fall, dass die Temperatur der Schleifen-Variable `c` größer oder gleich 38,5 ist, wird die Temperatur der Liste zugefügt. Das Ergebnis der Liste `fieberTemperaturen` ergibt also `[38.5, 41]`.

```
1 var fieberTemperaturen = [  
2   for (var c in [36.5, 38.5, 41])  
3     if (c >= 38.5) c  
4 ];
```

Listing 4: Collection-if in einer Liste, Quelle: Eigenes Listing

3.2.3 Typen ohne NULL-Zulässigkeit

Im Vergleich zu vielen anderen Programmiersprachen wie beispielsweise in Java wird in Dart zwischen gewöhnlichen Typen und nullable Typen unterschieden. In Sprachen wie beispielsweise Java ist es nur bei atomaren Datentypen wie `int` und `float` vorgeschrieben einen Wert anzugeben. `null` ist bei diesen primitiven Datentypen nicht als Wert erlaubt. Doch nicht atomare Datentypen erlauben immer die Angabe von `null` als Wert. `null` drückt dabei immer das Nichtvorhandensein von Daten aus. Ab Dart 2.12 kann allen Datentypen standardmäßig kein Null-Wert zugewiesen werden. Das hat den Vorteil, dass der Compiler sich darauf verlassen kann, dass eine Variable niemals den Wert `null` haben kann. Das ist besonders dann nützlich, wenn auf einem Objekt eine Methode aufgerufen wird. Ist die Referenz das Objekt ist in Wahrheit `null` so gibt es erst zur Laufzeit einen Fehler, da die Methode auf der Referenz `null` nicht aufgerufen werden kann. Damit ein Laufzeitfehler geworfen werden kann, muss vor jedem Aufruf einer Methode auf einer Referenz überprüft werden, ob die Referenzen nicht `null` sind. Würde diese Überprüfung nicht stattfinden, so könnte kein Laufzeitfehler geworfen werden und das Programm würde ohne Fehlermeldung abstürzen. Handelt es sich allerdings um eine Referenz, die niemals den Wert `null` annehmen kann, so kann der Compiler die Überprüfung auf Null-Werte für diese Referenzen überspringen. Damit erhöht sich zusätzlich die Ausführungsgeschwindigkeit, da die Überprüfung Zeit in Anspruch nimmt. Vor allem aber ist es vorteilhaft für den Entwickler, da der Compiler Fehlermeldungen und Warnungen mitteilen kann, wenn Operationen auf Variablen mit potenziellen Null-Werten verwendet werden. Die Abwesenheit von Daten ist jedoch bei der Entwicklung sehr wichtig. Nicht alle Variablen können immer einen Wert haben. Aus diesem Grund gibt es in Dart auch die Typen, die auch Null-Werte zulassen. Allerdings gelten besondere Regeln für diese Typen.

3.2.4 Typen mit Null-Zulässigkeit

Wird in Dart hinter einem Typen ein `?` angegeben, so kann die Variable nicht nur Werte annehmen, die dieser Datentyp zulässt sondern zusätzlich auch noch den Wert `null`. Methoden auf Objekten mit Null-Zulässigkeit aufzurufen ist nicht ohne weiteres möglich.

Im Listing 5 wird versucht die auf die Variable `fahrenheitTemperature` den Operator `-` anzuwenden um sie mit `32` zu subtrahieren. Der Compiler liefert jedoch einen Fehler, da der Wert der Variable `null` sein kann, wie die Notation `int?` anzeigt. Solange nicht feststeht, dass die Variable zur Laufzeit tatsächlich nicht `null` ist, kann das Programm nicht kompiliert werden.

```
1 void printTemperatureInCelsius(int? fahrenheitTemperature) {  
2   print((fahrenheitTemperature - 32) * 5 / 9);  
3 }
```

Listing 5: Collection-if in einer Liste, Quelle: Eigenes Listing

Zu diesem Zweck macht Dart von der sogenannten Type Promotion - deutsch Typ Be-

förderung - gebrauch. Mithilfe einer Fallunterscheidung kann vor Anwenden der Operation nachgesehen werden, ob der Wert der Variable nicht `null` ist. Innerhalb des Körpers der Fallunterscheidung wird der Typ der Variable automatisch in einen Typ ohne Null-Zulässigkeit befördert. Der Code in Listing 6 lässt sich daher wieder kompilieren.

```
1 void printTemperatureInCelsius(int? temperature) {
2   if (temperature != null) {
3     print((temperature - 32) * 5 / 9);
4   }
5 }
```

Listing 6: Collection-if in einer Liste, Quelle: Eigenes Listing

Eine Besonderheit stellen dabei allerdings Instanzvariablen dar. In Dart wird syntaktisch nicht zwischen dem Aufruf einer Getter-Methode oder einer Instanzvariable unterschieden. In Listing 7 könnte sich hinter den Aufrufen von `temperature` in den Zeilen 6 und 7 die Instanzvariable verbergen, die in Zeile 2 deklariert ist.

```
1 class Patient {
2   num? temperature;
3   Patient({this.temperature});
4
5   void printTemperatureInCelsius() {
6     if (temperature != null) {
7       print((temperature - 32) * 5 / 9);
8     }
9   }
10 }
```

Listing 7: Collection-if in einer Liste, Quelle: Eigenes Listing

Genauso könnte es aber auch sein, dass eine Klasse von `Patient` erbt und das Feld `temperature` mit einer gleichnamigen Getter-Methode überschreibt. Auch wenn es sehr unwahrscheinlich ist, könnte es trotzdem vorkommen, dass der Aufruf von `temperature` in Zeile 6 einen Wert zurückgibt, der nicht `null` ist und der darauffolgende Aufruf in Zeile 7 `null` liefert. So provoziert es die Klasse `UnusualPatient` im Listing 8. Beim ersten Aufruf von `temperature` wird die Zähl-Variable `counter` von 0 auf 1 erhöht. Die Abfrage, ob es sich bei dem Wert von `Counter` um eine ungerade Zahl handelt ist erfolgreich (Z. 6), weshalb mit 97,7 ein valider Wert zurückgegeben wird. Beim zweiten Aufruf erhöht sich `counter` allerdings auf 2. Die gleiche Abfrage schlägt dieses Mal fehl. Deshalb liefert die Getter-Methode nun `null` (Z. 9). Ein solches Szenario ist schon sehr unwahrscheinlich, doch die Typ-Überprüfung des Compilers arbeitet mit Beweisen. Im Fall von Instanzvariablen kann nicht bewiesen werden, dass zur Laufzeit ein solcher Fall ausgeschlossen werden kann.

Sollte sich der Entwickler sicher sein, dass die Variable nicht `null` sein kann, so kann er mit einem nachgestellten `!` erzwingen, dass die Variable als nicht `null` angesehen wird (Listing. Z. 3). Sollte es dann dennoch passieren, dass die Variable `null` ist, so wird eine Fehlermeldung beim Aufruf der Variable geworfen.

Eine noch sicherere Variante ist es, die Instanzvariable zuvor in eine lokale Variable zu speichern (Listing. 10 Z. 2). Die lokale Variable hat keine Möglichkeit zwischen den zwei Aufrufen einen unterschiedlichen Wert anzunehmen. Somit kann auch das Suffix `!` weggelassen werden (Z. 4).

```

1 class UnusualPatient extends Patient {
2     int counter = 0;
3
4     num? get temperature {
5         counter++;
6         if (counter.isOdd) {
7             return 97.7;
8         } else {
9             return null;
10        }
11    }
12 }

```

Listing 8: Collection-if in einer Liste, Quelle: Eigenes Listing

```

1 void printTemperatureInCelsius() {
2     if (temperature != null) {
3         print((temperature! - 32) * 5 / 9);
4     }
5 }

```

Listing 9: Collection-if in einer Liste, Quelle: Eigenes Listing

3.2.5 Asynchrone Programmierung

Wird auf eine externe Ressource zugegriffen - wie zum Beispiel das Abrufen einer Information von einem Webserver, oder das Lesen einer Datei im lokalen Dateisystem - so handelt es sich um asynchrone Operationen.

Im Sprachkern stellt Dart Schlüsselwörter und Datentypen für die asynchrone Programmierung bereit. Das sind unter anderem die Datentypen `Future` und `Stream` sowie die Schlüsselwörter `async` und `await`.

Future Ein `Future`-Objekt repräsentiert einen potenziellen einmaligen Wert, der in der erst in der Zukunft bereit steht. Er gleicht damit dem sogenannten `Promise` - deutsch Versprechen – in JavaScript. [Nurl {https://developer.mozilla.org/de/docs/orphaned/Web/JavaScript/Reference](https://developer.mozilla.org/de/docs/orphaned/Web/JavaScript/Reference)

Das Listing 11 zeigt mit dem Lesen einer Datei ein Beispiel für den Aufruf einer asynchronen Operation.

Anders als erwartet, befindet sich in der Variable `fileContent` in Wahrheit kein Text mit dem Inhalt der Datei. Stattdessen hat die Variable den Datentyp `Future<String>` und ist lediglich ein sogenannter „Handle“ - deutsch Referenzwert - für das potenzielle und zukünftige Ergebnis der Operation.

Mit der Übergabe einer Funktion, die bei Vollendung der Operation aufgerufen wird, kann der Wert ausgewertet werden. Man nennt diese Operation auch „Callback-Funktion“ - deutsch Rückruffunktion. Listing 12 zeigt, wie auf den Dateiinhalt zugegriffen werden kann. Über die Methode `then` wird eine Funktion übergeben, die genau einen Parameter hat. In diesem Parameter wird der Text der gelesenen Datei bei Vollendung der Operation übergeben.

Der Einsatz von „Callback-Funktionen“ kann den Quellcode stark verkomplizieren. Man spricht von der sogenannten „callback hell“ - deutsch Rückruffunktionen-Hölle -, wenn

```
1 void printTemperatureInCelsius() {
2   num? temperature = this.temperature;
3   if (temperature != null) {
4     print((temperature - 32) * 5 / 9);
5   }
6 }
```

Listing 10: Collection-if in einer Liste, Quelle: Eigenes Listing

```
1 var fileContent = file.readAsString();
```

Listing 11: Collection-if in einer Liste, Quelle: Eigenes Listing

solche „Callback-Funktionen“ über etliche Level hinweg ineinander verschachtelt sind.

Um genau das zu verhindern, existieren in Dart die Schlüsselwörter `async` und `await`. Genauso heißen sie auch in anderen Sprachen wie etwa C# ab Version 4.5 und JavaScript ab Version ES8. [Referenz](#)

Listing 13 zeigt, dass durch das Anwenden des Schlüsselwortes `await` vor der Operation `file.readAsString` dafür sorgt, dass der zukünftige Wert direkt in `fileContent` gespeichert wird. Ganz ohne „Callback-Funktion“ kann der Dateieinhalt in der darauffolgenden Zeile ausgegeben werden.

Doch jede Funktion, die auf andere Funktionsaufrufe wartet, muss selbst als asynchron gekennzeichnet werden. Dazu dient das `async` Schlüsselwort vor Beginn des Methoden-Körpers.

Streams „Streams“ liefern nicht nur einen Wert – wie im Fall eines `Future` – sondern eine Serie von Werten, die in der Zukunft geliefert werden. Listing 14 zeigt wie auf einen solchen Stream gehorcht werden kann. `countStream` liefert jede Sekunde einen neuen Wert, nämlich die aktuelle Sekunde - von 0 beginnt. Mit `countStream.listen` kann eine Funktion übergeben werden, die immer dann ausgeführt wird, wenn dem `countStream` ein neuer Wert hinzugefügt wurde. Der erste Parameter ist dabei der hinzugefügte Wert.

Es wird zwischen zwei Arten von Streams unterschieden. Solche, die genau einen Empfänger haben - „single subscription streams“ - und solche, die beliebig viele Empfänger haben können - „broadcast streams“.

Für die Formular Anwendung sind ausschließlich „broadcast streams“ zu berücksichtigen. Die Streams sollen verwendet werden, um Änderungen in der Eingabemaske zu behandeln. Die Oberflächenelemente horchen auf diese Änderungen. Teile der Oberfläche und damit die Oberflächenelemente, welche auf die Streams horchen, werden immer wieder neu gezeichnet. Dabei werden die Elemente entfernt und durch neu konstruierte ersetzt. Damit melden sich immer wieder Zuhörer vom „Stream“ ab und neue Elemente melden sich an. Daher kommen nur „broadcast streams“ infrage.

```
1 fileContent.then((text) {  
2   print("Der Datei-Inhalt ist: $text");  
3 });
```

Listing 12: Collection-if in einer Liste, Quelle: Eigenes Listing

```
1 printFileContent() async {  
2   var fileContent = await file.readAsString();  
3   print("Der Datei-Inhalt ist: $fileContent");  
4 }
```

Listing 13: Collection-if in einer Liste, Quelle: Eigenes Listing

```
1 var countStream = Stream<num>.periodic(const Duration(seconds: 1), (count) {  
2   return count;  
3 });  
4  
5 countStream.listen((count) {  
6   print("Gezählte Sekunden: $count");  
7 });
```

Listing 14: Collection-if in einer Liste, Quelle: Eigenes Listing

4 Technologie Auswahl

Dieses Kapitel behandelt die Auswahl der Frontend-Technologie für die Umsetzung der Formular-Anwendung. Dazu werden im ersten Schritt die dafür in Frage kommende Technologien identifiziert. Anschließend wird der Trend der Popularität dieser Technologien miteinander verglichen. Die daraus resultierenden Kandidaten sollen dann detaillierter untersucht werden. In Hinblick auf die Anforderungen an die Formular-Anwendung soll dabei die angemessenste Frontend-Technologie ausgewählt werden.

4.1 Trendanalyse

Zwei Quellen wurden für die Analyse der Technologie-Trends ausgewählt: die Ergebnisse der jährlichen Stack Overflow Umfragen und das Such-Interesse von Google Trends.

Stack Overflow Umfrage Die Internet-Plattform Stack Overflow richtet sich an Softwareentwickler und bietet ihren Nutzern die Möglichkeiten, Fragen zu stellen, Antworten einzustellen und Antworten anderer Nutzer auf- und abzuwerten. Besonders für Fehlermeldungen, die häufig während der Softwareentwicklung auftreten, findet man auf dieser Plattform rasch die Erklärung und den Lösungsvorschlag gleich mit. Dadurch lässt sich auch die Herkunft des Domain-Namens herleiten:

We named it Stack Overflow, after a common type of bug that causes software to crash – plus, the domain name stackoverflow.com happened to be available.
- Joel Spolsky, Mitgründer von Stack Overflow ⁵

Aufgrund des Erfolgsrezepts von Stack Overflow ist die Plattform kaum einem Softwareentwickler unbekannt. Dementsprechend nehmen auch jährlich tausende Entwickler an den von Stack Overflow herausgegebenen Umfragen teil. Seit 2013 beinhalten die Umfragen auch die Angabe der aktuell genutzten und in Zukunft gewünschten Frontend-Technologien. Stackoverflow erstellt aus diesen gesammelten Daten Auswertungen und Übersichten. Doch gleichzeitig werden die zugrundeliegenden Daten veröffentlicht. ⁶

Um den Trend der Beliebtheit der Frontend-Technologien aufzuzeigen, wurde ein Jupyter Notebook erstellt. Es transformiert die Daten in ein einheitliches Format, da die Umfrageergebnisse von Jahr zu Jahr in einer unterschiedlichen Struktur abgelegt wurden. Anschließend erstellt es Diagramme, die im Folgenden analysiert werden. Das Jupyter Notebook ist im Anhang zu finden.

Google Trends Suchanfragen die an die Suchmaschine Google abgesetzt werden, lassen sich über den Dienst Google Trends als Trenddiagramm Visualisieren. Um das relative Such-Interesse abzubilden, werden die Ergebnisse normalisiert, um die Ergebnisse auf einer Skala von 0 bis 100 darstellen zu können. ⁷

Google Trends ist keine wissenschaftliche Umfrage und sollte nicht mit Umfragedaten verwechselt werden. Es spiegelt lediglich das Suchinteresse an bestimmten Themen wider. ⁸

⁵Spolsky, „How Hard Could It Be?: The Unproven Path“

⁶Stack Exchange, Inc., *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*

⁷Vgl. Google LLC, *Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe*

⁸Google LLC, *Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe*

Genau aus diesem Grund wird Google Trends im Folgenden lediglich zum Abgleich der Ergebnisse der Stack Overflow Umfrage eingesetzt.

4.1.1 Frameworks mit geringer Relevanz

NativeScript, Sencha (bzw. Sencha Touch) und Appcelerator spielen in den Umfrageergebnisse eine Untergeordnete Rolle. Dies ist in den aufsummierten Stimmen von 2013 bis 2020 für alle in der Umfrage auftauchenden Frontend-Technologien zu sehen (Abb. ??).

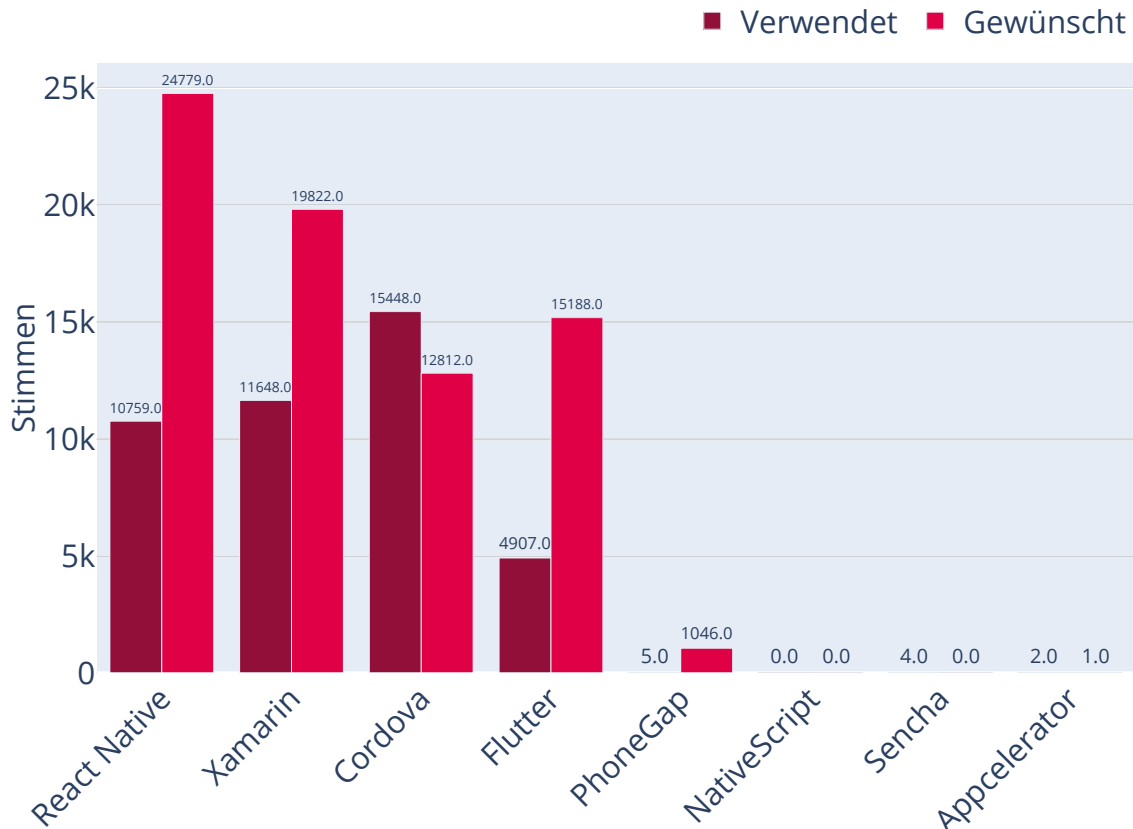


Abbildung 1: Summe der Stimmen der Stack Overflow Umfrage von 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

Auch das Suchinteresse auf Google ist für diese Frameworks äußerst gering. In Abbildung 2 werden NativeScript, Sencha, Appcelerator und auch Adobe PhoneGap mit Apache Cordova für das relative Suchinteresse verglichen.

Verwandte Technologien zu Apache Cordova Das Ionic Framework taucht in den Ergebnissen der Stack Overflow Umfragen nicht auf. Ein Grund dafür könnte sein, dass es auf Apache Cordova aufbaut¹⁰, welches bereits in den Ergebnissen vorkommt. Adobe PhoneGap taucht zwar in den Ergebnissen von 2013 mit 1043 Stimmen auf (Siehe Abbildung 3), verliert jedoch in den Folgejahren mit weniger als 10 Stimmen abruht an Relevanz. Das stimmt nicht mit dem Suchinteresse auf Google überein, da es dort ab 2013 sogar steigt, wie in Abbildung 2 zu sehen ist. 2013 existierte PhoneGap noch als extra Mehrfachauswahlfeld in den Daten, während es ab 2014 nur noch in dem Feld für die sonstigen Freitext Angaben auftaucht¹¹. Auch Adobe PhoneGap baut auf Apache Cordova auf¹². Für diese

¹⁰Lynch, *The Last Word on Cordova and PhoneGap*

¹¹Vgl. Stack Exchange, Inc., *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*

¹²Vgl. Adobe Inc., *FAQ / PhoneGap Docs*

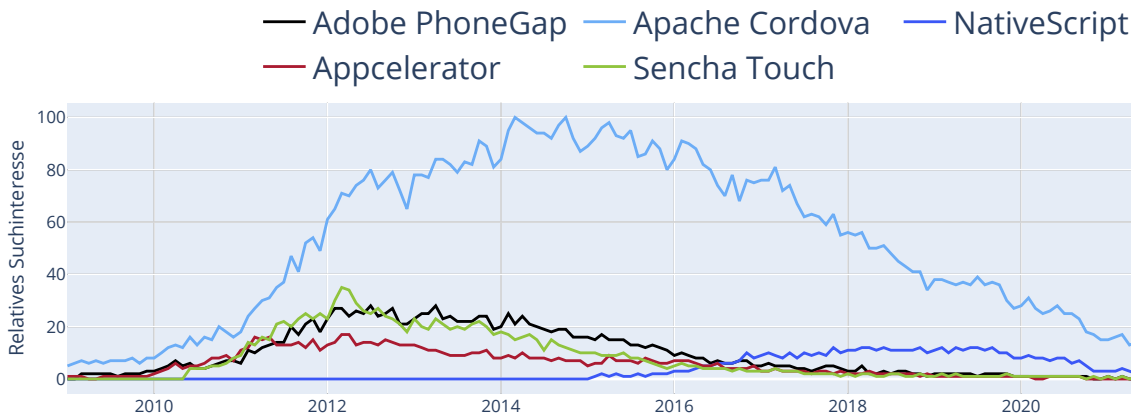


Abbildung 2: Suchinteresse der Frameworks mit geringer Relevanz, Quelle: Eigene Abbildung, Notebook: [Charts/GoogleTrends/GoogleTrends.ipynb](#), Daten-Quelle: Google Trends⁹

Auswertung spielen diese verwandten Technologien eine untergeordnete Rolle, da sie auch in den Google Trends weit hinter Apache Cordova zurückbleiben (Abb. 2).

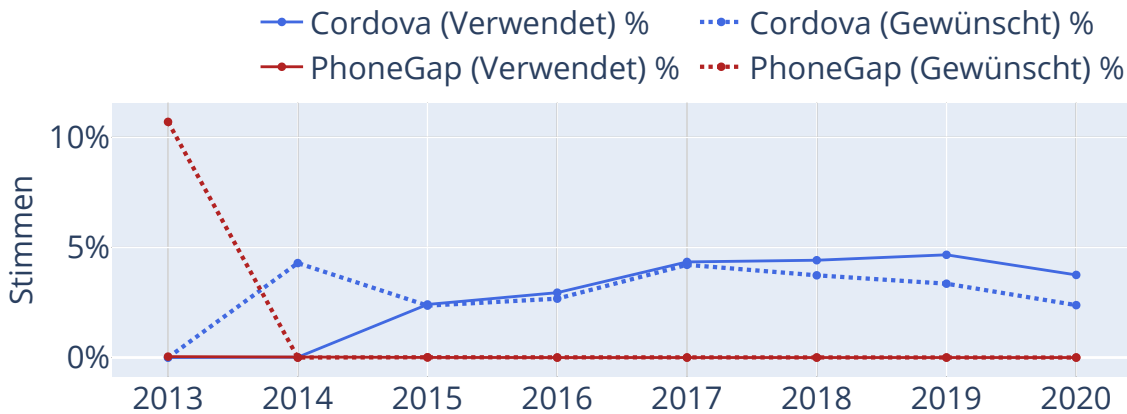


Abbildung 3: Stimmen für Cordova und PhoneGap 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

Am Beispiel von Adobe PhoneGap wird deutlich, wie wichtig es ist, auf eine Technologie zu setzen, die weit verbreitet ist. Im schlimmsten Fall wird die Technologie sogar vom Betreiber aufgrund zu geringer Nutzung komplett eingestellt, wie es bereits bei PhoneGap geschehen ist. Adobe gab am 11. August 2020 bekannt, dass die Entwicklung an PhoneGap eingestellt wird und empfiehlt die Migration hin zu Apache Cordova.¹³

4.1.2 Frameworks mit sinkender Relevanz

Die Technologien Xamarin und Cordova zeigen bereits einen abfallenden Trend, wie in Abbildung 4 ersichtlich ist. Im Fall von Xamarin gibt es immerhin mehr Entwickler, die sich wünschen, mit dem Framework zu arbeiten, als Entwickler, die tatsächlich mit Xamarin arbeiten. Cordova scheint in diesem Hinblick dagegen eher unbeliebt: es gibt mehr Entwickler, die mit Cordova arbeiten, als tatsächlich damit arbeiten wollen.

In Abbildung 5 ist noch einmal zu sehen, dass Google Trends die Erkenntnisse aus der Stack Overflow Umfrage reflektiert und es wird auch sichtbar, welche beiden Technologien möglicherweise der Grund für den Rückgang von Xamarin und Cordova sind.

¹³Vgl. Adobe Inc., *Update for Customers Using PhoneGap and PhoneGap Build*

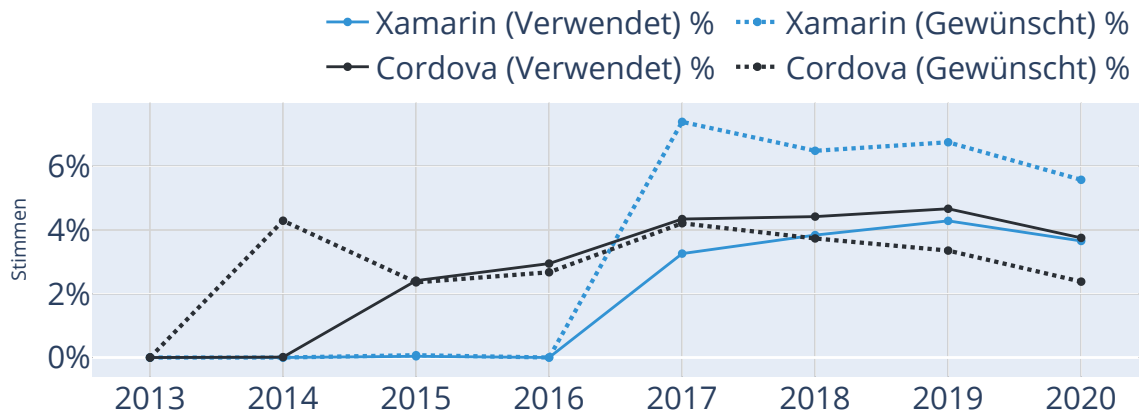


Abbildung 4: Stimmen für Xamarin und Cordova 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

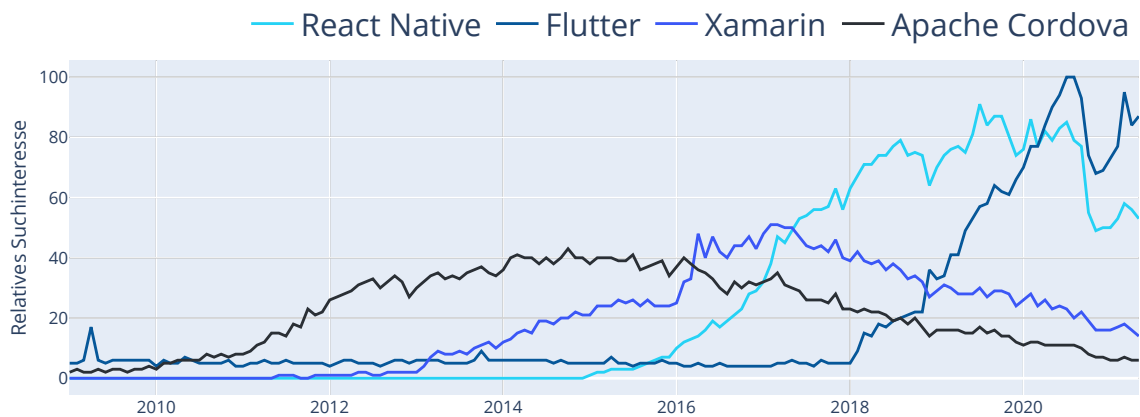


Abbildung 5: Suchinteresse sinkende und steigende Relevanz, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

4.1.3 Frameworks mit steigender Relevanz

Besser ist es, auf Technologien zu setzen, die noch einen steigenden Trend der Verbreitung und Beliebtheit zeigen. In Abbildung 6 wird sichtbar, dass es sich dabei um Flutter und immerhin im Hinblick auf die Verbreitung auch für React Native handelt. Ungünstigerweise wird React Native in der Stack Overflow Umfrage erst seit 2018 als tatsächliches Framework abgefragt. Vorher erschien lediglich das Framework React, welches nicht für den Vergleich der Cross-PlatformFrameworks eignet, da es sich um ein reines Web-Framework handelt. Doch auch die Ergebnisse von Google Trends zeigen einen ähnlichen Verlauf für die Jahre 2019 und 2020 (Abb. 5).

Im Vergleich von dem Jahr 2019 mit 2020 wird sichtbar, dass die Zahl der Entwickler, die sich wünschen, mit React Native zu arbeiten, gesunken ist. Dennoch ist die Anzahl der Entwickler, die mit React Native arbeiten möchten noch weit höher, als die der Entwickler, die tatsächlich mit React Native arbeiten.

Es ist möglich, dass der abfallende Trend daran liegt, dass die Zahl der Entwickler, die mit Flutter arbeiten möchten im selben Jahr gestiegen ist. React Native hat im Vergleich zu Flutter jedoch noch immer mehr aktive Entwickler und die Tendenz ist steigend. Doch die Anzahl der aktiven Flutter Entwickler zeigt einen noch stärker steigenden Trend. So könnte es sein, dass die Zahl der Flutter Entwickler die der React Native Entwickler in einem der nächsten Jahre überholt. Im Such-Interesse hat sich diese Entwicklung bereits vollzogen (Abb. 5).

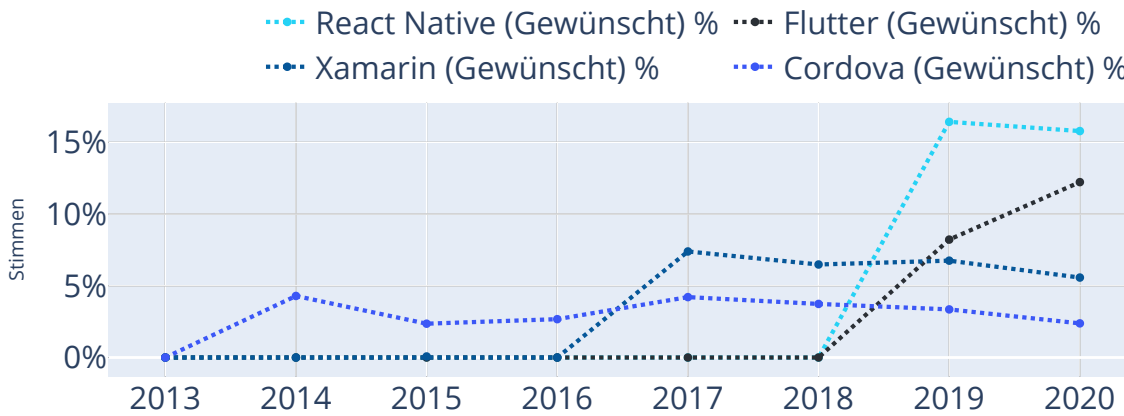


Abbildung 6: Stimmen für React Native und Flutter 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

Nichtsdestotrotz scheinen beide Technologien als Kandidaten für einen detaillierteren Vergleich für dieses Projekt in Frage zu kommen. Im nächsten Kapitel soll evaluiert werden, welches Framework für die Entwicklung der Formular-Anwendung angemessener ist.

4.2 Vergleich React Native und Flutter

4.2.1 Vergleich zweier minimaler Beispiele für Formulare und Validierung

Es soll eine Formularanwendung mit komplexer Validierung im Rahmen dieser These erstellt werden. Es ist durchaus sinnvoll, die beiden Technologien anhand von Beispielanwendungen, welche Formulare und die Validierung dieser beinhalten, zu vergleichen. Deshalb sollen nachfolgend jeweils eine solche Beispielanwendung der jeweiligen Technologie gefunden werden. Die Anwendung werden sich stark voneinander unterscheiden, weshalb sie im nächsten Schritt vereinfacht und aneinander angeglichen werden. Anschließend wird ersichtlich werden, nach welchen Kriterien sich die Technologien im Hinblick auf die Entwicklung der Formularanwendung vergleichen lassen.

React Native React native stellt nur eine vergleichsweise geringe Anzahl von eigenen Komponenten zur Verfügung und zu diesen gehören keine, welche die Validierung von Formularen ermöglichen. Doch die im react.js Raum sehr bekannten Bibliotheken Formic, Redux Forms und React Hook Form sind alle drei kompatibel mit React Native.^{14,15,16}

Für die Formular-Anwendung ist die Validierung komplexer Bedingungen nötig. Die Formular-Validierungs-Bibliotheken bieten in der Regel Funktionen an, welche überprüfen, ob ein Feld gefüllt ist oder der Inhalt einem speziellen Muster entspricht – wie etwa einem regulären Ausdruck. Doch solche mitgelieferten Validierungs-Funktionen reichen nicht aus um die Komplexität der Bedingungen abzubilden. Stattdessen müssen benutzerdefinierte Funktionen zum Einsatz kommen.

Keiner der drei oben genannten Validierungs-Bibliotheken ist in dieser Hinsicht limitiert. Sie alle bieten die Möglichkeit, eine JavaScript Funktion für die Validierung zu übergeben. Diese Funktion gibt einen Wahrheitswert zurück – wahr, wenn das Feld oder die Felder valide sind, falsch, falls nicht. In React Hook Form ist es die Funktion register,

¹⁴Vgl. *React Native / Formik Docs*

¹⁵Vgl. *Does redux-form work with React Native?*

¹⁶Vgl. *React Native / React Hook Form - Get Started*

die ein Parameter-Object namens Register Options erhält, dessen Eigenschaft validate die JavaScript Funktion zugewiesen werden kann.¹⁷ In Redux Form ist es die initialisierungsfunktion reduxForm, die ein Konfigurations-Objekt mit dem Namen config erhält, in welchem die Eigenschaft ebenfalls validate heißt.¹⁸ Auch in Formic ist der Bezeichner validate, und ist als Attribut in der Formic Komponente zu finden.¹⁹

Es ist also absehbar, dass die Formular-Anwendung in React Native entwickelt werden kann. Die nötigen Funktionen werden von den Bibliotheken bereitgestellt. Einziger Nachteil hierbei ist, dass es sich um Drittanbieter Bibliotheken handelt, welche im Verlauf der Zeit an Beliebtheit gewinnen und verlieren können. Möglicherweise geht die Beliebtheit einer der Bibliothek mit der Zeit zurück, weshalb es weniger Kontributionen wie etwa neue Funktionalitäten oder Fehlerbehebungen, sowie Fragen und Antworten und Anleitungen zu diesen Bibliotheken geben wird, da die Entwickler sich für andere Bibliotheken entscheiden. Die Wahl der Bibliothek kann also schwerwiegende Folgen wie Mangel an Dokumentation oder Limitationen im Vergleich zu anderen Bibliotheken mit sich bringen. Eine Migration von der einen Bibliothek zu einer anderen könnte in Zukunft notwendig werden, wenn diese Limitationen während der Entwicklung auffallen. Aus dem Grund ist es in der Regel von Vorteil, wenn solche Funktionalitäten bereits im Kern der Frontend-Technologie integriert sind. Der Fall, dass die Kern-Komponenten an Relevanz verlieren und empfohlen wird, auf externe Bibliotheken zuzugreifen, ist zwar nicht ausgeschlossen, geschied aber im wesentlichen seltener.

Flutter Die Flutter Dokumentation stellt in ihrer cookbook Sektion ein Beispiel einer minimalistischen Formularanwendung mit Validierung bereit.²⁰ Das Rezept ist Teil einer Serie von insgesamt fünf Anleitungen, welche Formulare in Flutter behandeln.²¹

4.2.2 Automatisiertes Testen

Automatisierte Test in React Native Die React Native Dokumentation führt genau eine Seite mit einem Überblick über die unterschiedlichen Testarten. Dabei wird das Konzept von Unit Tests, Mocking, Integrations Tests, Komponenten Tests und Snapshot Tests kurz erläutert, jedoch ohne ein Beispiel zu geben oder zu verlinken. Vier Quellcodeschnipsel sind auf der Seite zu finden: Ein Schnipsel zeigt den minimalen Aufbau eines Tests, zwei weitere Schnipsel veranschaulichen beispielhaft, wie Nutzerinteraktionen getestet werden können und Letzteres zeigt die textuelle Representation der Ausgabe einer Komponente, die für einen Snapshottest verwendet wird. Weiterhin wird auf die Jest API Dokumentation verwiesen, sowie auf ein Beispiel für einen Snapshot Test in der Jest Dokumentation.^I

Um die notwendigen Anleitungen für das Erstellen der jeweiligen Tests ausfindig zu machen, ist es notwendig, die Dokumentation von React Native zu verlassen.

Die Dokumentation von Jest enthält mehr Details zum Einsatz der Testbibliothek, welches für mehrere auf Javascript basierende Frontend Framework kompatibel ist^{II}. Somit muss zum Erstellen der Unit-Tests immerhin nur dieses Framework studiert werden.

Zum Entwickeln von Test von React Testing Library React Native Komponenten wird

¹⁷Vgl. *register* / *React Hook Form* - API

¹⁸Vgl. *reduxForm* / *Redux Form* - API

¹⁹Vgl. *<Formik />* / *Formik Docs* API

²⁰Vgl. Google LLC, *Build a form with validation*

²¹Vgl. Google LLC, *Forms* / *Flutter Docs Cookbook*

^I<https://jestjs.io/docs/snapshot-testing>

^{II}<https://jestjs.io/docs/getting-started>

unter anderem auf die Bibliothek React Native Testing Library verwiesen. Anders als der Name vermuten lässt, handelt es sich nicht um eine von React Native bereitgestellte Bibliothek. Im Unterschied zur React Testing Library, von der sie inspiriert ist, läuft sie ebenso wie React Native selbst nicht in einer Browser-Umgebung.²² Herausgegeben wird die react native testing Library vom Drittanbieter Callstack - ein Partner im React Native Ökosystem.²³

Sie verwendet im Hintergrund den React Test Renderer^{III}, welcher wiederum vom React Team angeboten wird und auch zum Testen von react.js Anwendungen geeignet ist. Der React Test Renderer Wird ebenfalls empfohlen, um Komponententest zu kreieren, die keine React Native spezifischen Funktionalitäten nutzen.

Um Integrationstest so entwickeln, welche die Applikation auf einem physischen Gerät oder auf einem Emulator testen, wird auf zwei weitere Drittanbieter Bibliotheken verlinkt: Appium^{IV} und Detox^V. Es wird darauf hingewiesen, dass Detox speziell für die Entwicklung von React Native Integrationstest entwickelt wurde. Appium wird lediglich als ein weiteres bekanntes Werkzeug erwähnt.

Es lässt sich damit zusammenfassen, dass der Aufwand der Einarbeitung für automatisiertes Testen in React Native vergleichsweise hoch ist. Die Dokumentation ist auf die Seiten der jeweiligen Anbieter verteilt. Der Entwickler muss sich den Überblick selbst verschaffen und zusätzlich die für das Framework React Native relevanten Inhalte identifizieren. Notwendig ist auch das Erlernen von mehreren APIs um alle Testarten abzudecken. Für einen Anfänger kommt erschwerend hinzu, dass eine Entscheidung für die eine oder andere Bibliothek notwendig wird. Um diese Entscheidung treffen zu können, ist eine Auseinandersetzung mit den Vor- und Nachteile der Technologien im Vorfeld vom Entwickler zu leisten.

Automatisierte Test in Flutter Die Flutter Dokumentation erklärt sehr umfangreich auf 11 Unterseiten die unterschiedlichen Testarten mit Quellcodebeispielen und verlinkt für jede Testart eine bis mehrere detaillierte Schritt-für-Schritt-Anleitungen, wie ein solcher Test erstellt wird.

Einer Seite erklärt den Unterschied zwischen Unit Test, Widget Test und Integrationstest^{VI}. Eine weitere Seite erklärt Integrationstests in mehr Detail^{VII}.

Ein sogenanntes Codelab führt durch die Erstellung einer minimalistischen App und zwei Unit-, fünf Widget- und zwei Integrationstest für diese App^{VIII}

Im sogenannten Kochbuch tauchen folgende Rezepte auf:

- 2 Rezepte für Unit Tests

– eine grundlegende Anleitung zum Erstellen von Unit-Tests^{IX}

²²Vgl. Borenkraout, *Native Testing Library Introduction | Testing Library Docs*

²³Vgl. Facebook Inc., *The React Native Ecosystem*

^{III}<https://reactjs.org/docs/test-renderer.html>

^{IV}<http://appium.io/>

^V<https://github.com/wix/detox/>

^{VI}<https://flutter.dev/docs/testing>

^{VII}<https://flutter.dev/docs/testing/integration-tests>

^{VIII}<https://codelabs.developers.google.com/codelabs/flutter-app-testing>

^{IX}<https://flutter.dev/docs/cookbook/testing/unit/introduction>

- Eine weitere Anleitung zum Nutzen von mocks in Unit Test mithilfe der Bibliothek mockito ^X
- 3 Rezepte für Widget Tests
 - Eine grundlegende Anleitung zum Erstellen von Widget Test ^{XI}
 - Ein Rezept mit detaillierteren Beispielen zum Finden von widgets zur Laufzeit eines Widget Tests ^{XII}
 - Ein Rezept zum Testen von Nutzerverhalten wie dem Tab, dem Drag und dem eingeben von Text ^{XIII}
- 3 Rezepte für Integrationstests
 - Eine grundlegende Anleitung zum Erstellen eines Integrationstest ^{XIV}
 - eine Anleitung zum simulieren von scrollen in der Anwendung während der Laufzeit eines Integrationstest ^{XV}
 - eine Anleitung zum Performance Profiling ^{XVI}

Zusammengefasst: Der Aufwand der Einarbeitung in das Testen in Flutter ist gering. Alle Werkzeuge werden vom Dart- und Flutter-Team bereitgestellt. Die Dokumentation ist umfangreich, folgt jedoch einem roten Faden. Eine Übersichtsseite fasst die Kerninformationen zusammen und verweist auf die jeweiligen Seiten für detailliertere Informationen und Übungen.

^X<https://flutter.dev/docs/cookbook/testing/unit/mocking>
^{XI}<https://flutter.dev/docs/cookbook/testing/widget/introduction>
^{XII}<https://flutter.dev/docs/cookbook/testing/widget/finding>
^{XIII}<https://flutter.dev/docs/cookbook/testing/widget/tap-drag>
^{XIV}<https://flutter.dev/docs/cookbook/testing/integration/introduction>
^{XV}<https://flutter.dev/docs/cookbook/testing/integration/scrolling>
^{XVI}<https://flutter.dev/docs/cookbook/testing/integration/profiling>

Teil II

Implementierung

5 Schritt 1 - Formular in Grundstruktur erstellen

Im ersten Schritt soll die Formular-Anwendung in ihrer Grundstruktur entwickelt werden. Das beinhaltet alle drei Oberflächen, welche in den darauf folgenden Schritten lediglich erweitert werden. Das Formular erhält noch keine Validierung. Somit sind alle Eingaben oder nicht kompatible Selektionen erlaubt. Die erste Ansicht, welche der Benutzer sieht, soll die Übersicht der bereits eingetragenen Maßnahmen sein (Abb. 7).

Zuletzt bearbeitet am	Maßnahmentitel
2021-7-9 18:44	Massnahme 1

Zuletzt bearbeitet am	Maßnahmentitel
2021-7-9 18:44	Massnahme 2
2021-7-9 18:44	Massnahme 3
2021-7-9 18:44	Massnahme 4
2021-7-9 18:58	Massnahme 5

Abbildung 7: Der Übersicht-Bildschirm zeigt in Schritt 1 zunächst nur die Maßnahmen mit ihrem Titel und Bearbeitungsdatum in den Kategorien „Abgeschlossen“ und „In Bearbeitung“, Quelle: Eigene Abbildung

Die Auflistung der Maßnahmen erfolgt in den Kategorien „In Bearbeitung“ und „Abgeschlossen“. Innerhalb dieser Rubriken werden die Maßnahmen in einer Tabelle angezeigt. Mit einem Klick auf den Button unten rechts im Bild wird der Benutzer auf die zweite Ansicht weitergeleitet: die Eingabemaske (Abb. 8).

Status
in Bearbeitung

Maßnahmentitel
Massnahme 7

Abbildung 8: Die Eingabemaske zeigt im Schritt 1 eine Karte zum Selektieren des Status und ein Eingabefeld für den Titel, Quelle: Eigene Abbildung

Sie ermöglicht die Eingabe des Maßnahmen-Titels über ein simples Eingabefeld. Darüber hinaus ist die Selektions-Karte für den Status zu sehen. Mit einem Klick auf diese Karte öffnet sich der Selektions-Bildschirm. Er ermöglicht die Auswahl der Auswahloptionen, in

diesem Fall die Optionen „in Bearbeitung“ und „abgeschlossen“ (Abb. 9).



Abbildung 9: Der Selektions-Bildschirm für das Feld Status erlaubt die Auswahl der Optionen „in Bearbeitung“ und „abgeschlossen“, Quelle: Eigene Abbildung

5.1 Auswahloptionen hinzufügen

Dart verfügt – anders als beispielsweise Java²⁴ – nicht über Aufzählungstypen mit zusätzlichen Eigenschaften. Das Schlüsselwort `enum` in Dart erlaubt lediglich die Auflistung konstanter Symbole²⁵. Für die Auswahl Optionen ist es jedoch notwendig, dass es zwei Eigenschaften gibt:

- die Abkürzung, die in der resultierenden Datei gespeichert werden soll
- und der Beschreibungstext, welcher in der Oberfläche angezeigt wird.

Das hat den Hintergrund, dass die Abkürzungen weniger Speicherplatz einnehmen und die Beschreibung sich in Zukunft auch ändern darf. Würde anstatt der Abkürzung die Beschreibung als Schlüssel verwendet werden, so würde eine Datei, die mit einer älteren Version des Formulars erstellt wurde, nicht mehr von neueren Versionen der Applikation-eingelesen werden können. Der alte Beschreibungstext würde nicht mehr mit dem Text übereinstimmen, der als Schlüssel in der Anwendung verwendet wird.

Die beiden Zustände „in Bearbeitung“ und „abgeschlossen“ werden daher in Listing ?? als statische Klassenvariablen deklariert (Z. 6-7). Die beiden Konstruktor-Aufrufe übergeben dabei als erstes Argument die Abkürzung und als zweites Argument die Beschreibung. Der Konstruktor selbst (Z. 9-10) deklariert die beiden Parameter als positionale Parameter.

```
5 class LetzterStatus extends Choice {  
6   static final bearb = LetzterStatus("bearb", "in Bearbeitung");  
7   static final fertig = LetzterStatus("fertig", "abgeschlossen");  
8  
9   LetzterStatus(String abbreviation, String description)  
10    : super(abbreviation, description);  
11 }
```

Listing 15: Die Klasse `LetzterStatus`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/choices/choices.dart](#)

Positionale Parameter Im Vergleich zu den benannten Parametern ist bei den positionalen Parametern nur ihre Reihenfolge in der Parameterliste ausschlaggebend. Das Argument für die `abbreviation` steht dabei also immer an erster Stelle und das Argument für `description` immer an der zweiten (Z. 6-7). Positionale Parameter sind vorgeschrieben. Werden sie ausgelassen, so gibt es einen Compilerfehler.²⁶

²⁴Vgl. Gosling u. a., *The Java® Language Specification Java SE 16 Edition*, S. 321.

²⁵Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. 74f.

²⁶Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. 74f.

Die Klasse `LetzterStatus` erbt von der Basisklasse `Choice` (Z. 5). Der Konstruktor der Klasse (Z. 9) übergibt beide Parameter als Argumente an den Konstruktor der Klasse `Choice`. Genau wie in Java wird mithilfe des Schlüsselwortes `super` (Z. 10) der Konstruktor der Basisklasse aufgerufen. Doch anders als in Java erfolgt der Aufruf des `super` Konstruktors nicht in der ersten Zeile des Konstruktor-Körpers²⁷. Weil das Aufrufen des Konstruktors der Basisklasse zum statischen Teil der Objekt-Instanziierung gehört, muss der Aufruf von `super` in der Initialisierungsliste erfolgen. Die Initialisierungsliste wird mit dem `:` nach der Parameterliste eingeleitet (Z. 10)²⁸.

Die Basisklasse `Choice` (Listing. 16) deklariert lediglich die beiden Felder `description` und `abbreviation` jeweils als `String` (Z. 4-5). Beide sind mit `final` gekennzeichnet, was sie zu unveränderlichen Instanzvariablen macht. Nach der Initialisierung, können sie keine anderen Werte annehmen.²⁹ Die Initialisierung der beiden Variablen muss im statischen Kontext der Instanziierung erfolgen. Mit der abgekürzten Schreibweise `this.abbreviation` und `this.description` im Konstruktor (Z. 7) werden die Parameter den Feldern zugewiesen.

```
3 class Choice {
4   final String description;
5   final String abbreviation;
6
7   const Choice(this.abbreviation, this.description);
```

Listing 16: Die Klasse `Choice`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/choices/base/choice.dart](#)

Dies erübrigt sowohl die Angabe des Parametertypes mittels `(String abbreviation, String description)`, denn der Typ des Parameters kann bereits durch Angabe des Typs in der Instanzvariablen-Deklaration (Z. 4-5) abgeleitet werden. Außerdem entfällt auch die Zuweisung, die man ansonsten in der Form `this.abbreviation = abbreviation` und `this.description = description` in der Initialisierungsliste erreichen würde.³⁰

```
13 final letzterStatusChoices = Choices<LetzterStatus>(
14     {LetzterStatus.bearb, LetzterStatus.fertig},
15     name: "Status");
```

Listing 17: Die Menge `letzterStatusChoices`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/choices/choices.dart](#)

Die Variable `letzterStatusChoices` fasst die beiden statischen Klassenvariablen als eine Kollektion zusammen. Da es sich um eine solche Kollektion handelt, in der jedes Element nur ein einziges Mal vorkommen darf, ist hier von einer Menge zu sprechen. Auffällig hier ist, dass das Schlüsselwort `new` fehlt. In Dart ist das Schlüsselwort für die Konstruktion von Instanzen optional. Die Klasse, die zur Konstruktion dieser Menge verwendet wird, ist die selbst erstellte Klasse `choices`. Über das Typargument `LetzterStatus` wird erreicht, dass ausschließlich Variablen dieses Typs in der Menge eingefügt werden dürfen. Wird stattdessen eine Variable eingefügt, die weder vom selben Typ, noch von einem Typ, der von `letzterStatus` erbt, so gibt es einen Compilerfehler. Dies dient einzig und allein dem Zweck, dem Fehler vorzubeugen, dass aus Versehen falsche Optionen in der Menge eingetragen werden. Über den Parameter `name` ist es möglich dieser Menge die Beschriftung "Status" hinzuzufügen. Es handelt sich hier um einen benannte Parameter.

Listing 18 zeigt die Klasse `Choices`. Sie erbt von `UnmodifiableSetView` und erlaubt damit die Erstellung einer eigenen Menge - auch `Set` genannt **Referenz**. Methoden, die man von

²⁷Vgl. Gosling u. a., *The Java® Language Specification Java SE 16 Edition*, S. 310.

²⁸Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. 42.

²⁹Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. S16.

³⁰Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. 40f.

einem `Set` erwartet, lassen sich somit direkt auf Instanzen der Klasse `Choices` aufrufen. Darunter unter anderem die `contains` Methode, welche erlaubt, das Vorhandensein eines Objektes im `Set` zu überprüfen [Referenz](#).

```
10 class Choices<T extends Choice> extends UnmodifiableSetView<T> {
11     final String name;
12     final Map<String, T> choiceByAbbreviation;
13
14     T? fromAbbreviation(String? abbreviation) => choiceByAbbreviation[abbreviation];
15
16     Choices(Set<T> choices, {required this.name})
17         : choiceByAbbreviation = {
18         for (var choice in choices) choice.abbreviation: choice,
19         },
20         super(choices);
21 }
```

Listing 18: Die Klasse `Choices`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/choices/base/choice.dart](#)

Instanzvariable `name` (Z. 11) wird im Konstruktor 16 zugewiesen. Auffällig hierbei ist, dass der Parameter in geschweiften Klammern geschrieben steht und das Schlüsselwort `required` vorangestellt ist. Das macht den Parameter zu einem vorgeschriebenen benannten Parameter.

Vorgeschriebene benannte Parameter Gewöhnlicher benannte Parameter sind optional. Wird ihnen das Schlüsselwort `required` vorangestellt, so müssen sie gesetzt werden, denn sonst gibt es einen Compilerfehler. An dieser Stelle ist das `required` Schlüsselwort sinnvoll, denn es handelt sich um den Datentyp `String` der nicht den wert `null` annehmender. Würde der Parameter aber optional sein, so wäre es möglich, das programm zu kompilieren, auch wenn bei Aufrufen des Konstruktors kein Argument für den Parameter übergeben wurde. Doch in diesem Fall gäbe es keinen Initialwert für Name und somit müsste der Instanzvariable `null` zugewiesen werden. in der statischen Analyse wird daher sichergestellt, das Instanzvariablen durch benannte Parameter nur dann initialisiert werden dürfen, wenn dieser durch `required` als vorgeschrieben gekennzeichnet sind und damit unter keinem Umstand ausgelassen werden können. Dürfte `name` den Wert `Null` annehmen, So würde es sich um den nullable Datentyp `String?` Handeln.

Neben `name` wird mit `choiceByAbbreviation` eine weitere Instanzvariable deklariert (Z. 12). Es handelt sich um den Datentyp `Map` - eine Kollektion die Daten mittels Schlüssel Wertepaaren ablegen kann. Als Schlüssel wird die Abkürzung mit dem Datentyp `String` verwendet. Als Wert ist der generische Typ-Parameter `T` angegeben. Er ist in Zeile 10 deklariert und muss mindestens von der Klasse `Choice` erben. In `choiceByAbbreviation` werden also die Auswahlmöglichkeiten über ihre Abkürzung abgelegt und können über dieselbe wieder referenziert werden. Da es sich auch hier um eine unveränderliche Instanzvariable handelt, muss sie schon in der Initialisierungsliste initialisiert werden (Z. 17-19). Dabei wird zunächst mit der öffnenden geschweiften Klammer (Z. 17) ein sogenanntes Literal einer Map begonnen, welches mit schließenden geschweiften Klammer (Z. 19) endet. Mehr zu `Map` Literalen in dem Grundlagenkapitel [Kapitel einfügen](#).

Auffällig ist jedoch, dass In Zeile 18 dem Set lateral keine einfache Auflistung von Werten übergeben wird. Stattdessen wird das mit dem sogenannte collection for eine wiederholung verwendet.

In Zeile 18 wird durch die Menge aller Auswahloptionen `choices` iteriert und dabei in jedem Schleifendurchlauf die Auswahloption in die Variablen `choice` gespeichert. Während des Schleifendurchlauf wird dann ein Schlüssel-Wertepaar gebildet wobei `choice.abbreviation`

der Schlüssel ist und das Objekt `choice` selbst der Wert.

Die Map `choiceByAbbreviation` erlaubt es nach der Initialisierung mit Hilfe der Methode `fromAbbreviation` (Z. 14) über die Abkürzung das dazugehörige `Choice`-Objekt abzurufen. Beispielsweise gibt der Befehl `letzterStatusChoices.fromAbbreviation("fertig")` das Objekt `LetzterStatus("fertig", "abgeschlossen")` zurück. Auffällig dabei ist das der Parameter `abbreviation` mit dem Typen `String?` und der generische Rückgabetypp mit `T?` gekennzeichnet ist. Der Suffix `?` macht beide zu Typen mit Null-Zulässigkeit.

Die Methode `fromAbbreviation` soll für die Deserialisierung genutzt werden. Sollten im Formular Auswahlfelder leer gelassen worden sein, so haben entsprechenden Variablen den Wert `null`. Wenn nun das Formular abgespeichert wird, so tauchen auch in der abgespeicherten Json-Datei keine Werte für das Feld auf. Wird die Datei gelesen wird die Methode `fromAbbreviation` genutzt um aus der in der Json-Datei gespeicherten Abkürzung wieder die entsprechende Auswahl Option abzurufen. Sollte jedoch kein Wert hinterlegt sein, so wird `letzterStatusChoices.fromAbbreviation(null)` aufgerufen werden. Dadurch wird klar, dass der Parameter `null` zulassen muss. Es impliziert auch, dass potenziell `null` zurückgegeben werden kann, da für den Schlüssel `null` kein Wert in der Map hinterlegt sein kann. Deshalb erlaubt auch der Rückgabetypp `T?` Null-Werte.

5.2 Serialisierung einer Maßnahme

Damit die Daten angezeigt und verändert werden können, müssen sie zunächst serialisierbar sein, sodass sie auf einen Datenträger geschrieben und von dort auch wieder gelesen werden können. Die zwei bekanntesten Bibliotheken zum Serialisieren in Dart heißen `json_serializable` und `built_value`. Beide haben gemeinsam, dass sie Quellcode generieren, welcher die Umwandlung der Objekte in JSON übernimmt. `built_value` bietet im Gegensatz zu JSON Serializable jedoch die Möglichkeit unveränderbare Werte-Typen - sogenannte *immutable value types* - zu erstellen. Da diese unveränderbaren Werte noch bei der Erstellung des sogenannten ViewModels - Mehr dazu im Kapitel **Einfügen** - hilfreich werden, wurde sich für diese Bibliothek entschieden.

Ein Werte-Typ für `built_value` erfordert etwas Boilerplate-Code, um den generierten Quellcode mit der selbstgeschriebenen Klasse zu verknüpfen. Entwicklungsumgebungen wie Visual Studio Code und Android Studio erlauben solchen Boilerplate Code generieren zu lassen und dabei nur die erforderlichen Platzhalter einzugeben. In Visual Studio Code werden diese Templates „Snippets“ genannt, in Android Studio heißen sie „Live Templates“. Listing 140 zeigt, wie das live Template für das Generieren eines Wertetyps für `built_value` aussieht. Templates für `built_value` wie dieses und weitere müssen nicht vom Nutzer eingegeben werden, sondern existieren bereits als Plugin für die beiden Entwicklungsumgebungen^{XVII}, ^{XVIII}.

```
6 part '$file_name$.dart';
7
8 abstract class $ClassName$ implements Built<$ClassName$, $ClassName$Builder> {
9     $todo$
10
11     $ClassName$. _();
12     factory $ClassName$([void Function($ClassName$Builder) updates]) = _$$$ClassName$;
13 }
```

Listing 19: Live Template für die Erstellung von `built_value` Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

`$ClassName$` Wird dabei jeweils durch den gewünschten Klassennamen ersetzt. Android Studio erlaubt, dass bei Einfügen des live templates der Klassenname einmalig eingegeben werden muss. Anschließend wird mithilfe des live templates der Boilerplate Code generiert.

In Listing 20 ist der Werte-Typ `Maßnahme` zu sehen. Die Zeilen 11 bis 13, sowie 23 bis 28 wurden dabei automatisch erstellt. Die Zeilen 14 bis 21 wurden hinzugefügt. Zunächst soll die Maßnahme über die `guid` eindeutig identifiziert werden können.

Globally Unique Identifier Ein GUID – Kurzform von Globally Unique Identifier – ist eine Folge von 128 Bits, die zur Identifikation genutzt werden kann. Eine solche GUID hat einer textuelle Repräsentation wie beispielsweise die folgende: `'f81d4fae-7dec-11d0-a765-00a0c91e6bf6'`

Die Attribute `letzteBearbeitung` und `identifikatoren` sind im Gegensatz zu dem String-Attribut `guid` zusammengesetzte Datentypen, die im Folgenden weiter beleuchtet werden.

Auffällig ist, dass es sich hier um eine abstrakte Klasse handelt und die drei Attribute jeweils Getter-Methoden ohne Implementierung sind. Eine solche Getter-Methode speichert keinen wert, sondern gibt lediglich den Wert eines Feldes zurück. Die dazugehörigen

^{XVII}<https://plugins.jetbrains.com/plugin/13786-built-value-snippets>

^{XVIII}<https://marketplace.visualstudio.com/items?itemName=GiancarloCode.built-value-snippets>

```

6 part 'massnahme.g.dart';
7
8 abstract class Massnahme implements Built<Massnahme, MassnahmeBuilder> {
9   String get guid;
10
11   LetzteBearbeitung get letzteBearbeitung;
12
13   Identifikatoren get identifikatoren;
14
15   static void _initializeBuilder(MassnahmeBuilder b) =>
16     b..guid = const Uuid().v4();
17
18   Massnahme._();
19
20   factory Massnahme([void Function(MassnahmeBuilder) updates]) = _$Massnahme;
21
22   static Serializer<Massnahme> get serializer => _$massnahmeSerializer;
23 }

```

Listing 20: Der Werte-Typ `Massnahme`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/massnahme.dart](#)

Felder, Setter-Methoden, die konkrete Klasse und der restliche generierte Code ist in der gleichnamigen Datei mit der Endung `.g.dart` (Zeile 11) zu finden.

Die Klassen-Methode `_initializeBuilder` kann in jedem Werte-Typ hinterlegt werden, um Standardwerte für Felder festzulegen. Die Methode wird intern von „built_value“ aufgerufen. Bei dem Feld „guid“ handelt es sich um einen String, der keine Null-Werte zulässt. Könnte das Feld auch Null-Werte annehmen, so wäre die Notation in Dart dafür stattdessen `String? get guid;` „built_value“ erwartet also immer einen Wert für dieses Feld. Sollte die Datei gelesen werden, welche die Maßnahmen enthält, so enthält jede Maßnahme bei der Deserialisierung den abgespeicherten Wert für die `guid` und somit wird das Feld gefüllt. Doch sollte eine leere Maßnahme über einen Konstruktor erstellt werden, so wäre das Feld „guid“ leer und „built_value“ würde einen Fehler auslösen. Aus diesem Grund wird in der Zeile 21 für das Feld `guid` ein Standardwert festgelegt, nämlich eine zufällige generierte ID die dem Standard Uuid der Version 4 entspricht. Zu diesem Zweck wird das Builder-Objekt verwendet. Die Klasse `MassnahmeBuilder` gehört dabei zu dem von „built_value“ generierten Quellcode. Der Parametername wird hier – wie so häufig im builder pattern – mit einem `b` für Builder abgekürzt. Die Syntax `=>` leitet einen sogenannten „arrow function body“ ein. Dabei handelt es sich schlicht um einen Funktions-Körper, der genau eine Anweisung ausführt und deshalb nicht in öffnenden und schließenden geschweiften Klammern gesetzt werden muss.³¹ Auf dem Builder-Objekt können dann die Eigenschaften so gesetzt werden, als wären sie die Eigenschaften von dem Objekt `Massnahme`. In Wahrheit werden sie aber nur auf den Builder-Objekt angewendet. Ebenfalls auffällig ist die Syntax `b..guid`. Statt dem Punkt zum Zugriff auf Attribute des Objektes wird hier der sogenannte Kaskadierungs-Operator benutzt.

Der Kaskadierungs-Operator Durch Eingabe von zwei aufeinanderfolgende Punkten `..` statt nur einem `.` können mehrere Operationen an einem Objekt ausgeführt werden, ohne das Objekt zuvor einer Variable Zuzuweisen oder die Operationen über dessen Namen wiederholt aufzurufen.³² Beispiel: die Aufrufe `objekt.tueEtwas();` `objekt.tueEtwasAnderes();` und `objekt..tueEtwas()..tueEtwasAnderes();` sind äquivalent.

Da der Kaskadierungs-Operator jedoch dazu verwendet wird, mehrere Operationen auf ei-

³¹Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. 18f., 234.

³²Vgl. Google LLC, *Dart Programming Language Specification 5th edition*, S. 149f.

nem Objekt auszuführen, hat er in Zeile 16 keine Funktion. Doch bei Änderung eines Objektes über das builder pattern werden für gewöhnlich mehrere Operationen am gleichen Builder Objekt ausgeführt, weshalb - der Stringenz halber - der Kaskadierung-Operator immer im Zusammenhang mit dem Bilder Objekt verwendet wird.

Die Attribute `letzteBearbeitung` und `identifikatoren` (Z. 11, 13) erhalten dagegen ganz automatisch Standardwerte in Form von Instanzen der dazugehörigen Klassen. Diese wiederum konfigurieren ihre eigenen Felder und deren initialen Werte.

Der Werte-Typ `Identifikatoren` ist in Listing 21 zu sehen. Er enthält das Attribut `massnahmenTitel`, welcher im Eingabeformular durch das Texteingabefeld gefüllt werden wird.

```
25 abstract class Identifikatoren
26     implements Built<Identifikatoren, IdentifikatorenBuilder> {
27     String get massnahmenTitel;
28
29     static void _initializeBuilder(IdentifikatorenBuilder b) =>
30         b..massnahmenTitel = "";
```

Listing 21: Der Werte-Typ `Identifikatoren`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/massnahme.dart](#)

Schließlich enthält der Werte-Typ `LetzteBearbeitung` in Listing 22 noch die Attribute `letztesBearbeitungsDatum` in Zeile 43 und `letzterStatus` in Zeile 50. Im Eingabeformular wird der Selektions-Bildschirm den Inhalt des Feldes `letzterStatus` Bestimmen. Der initiale Wert auf wird in Zeile 54 auf einen konstanten Wert gesetzt, der dem Zustand `'in Bearbeitung'` entspricht - mehr dazu im Kapitel **Kapitel einfügen**.

```
41 abstract class LetzteBearbeitung
42     implements Built<LetzteBearbeitung, LetzteBearbeitungBuilder> {
43     DateTime get letztesBearbeitungsDatum;
44
45     String get formattedDate {
46         final date = letztesBearbeitungsDatum;
47         return "${date.year}-${date.month}-${date.day} ${date.hour}:${date.minute}";
48     }
49
50     String get letzterStatus;
51
52     static void _initializeBuilder(LetzteBearbeitungBuilder b) => b
53         ..letztesBearbeitungsDatum = DateTime.now().toUtc()
54         ..letzterStatus = LetzterStatus.bearb.abbreviation;
```

Listing 22: Der Werte-Typ `LetzteBearbeitung`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/massnahme.dart](#)

Das Attribut `letztesBearbeitungsDatum` ist dagegen nicht im Formular änderbar, sondern wird einmalig in Zeile 53 auf den aktuellen Zeitstempel gesetzt. Zugehörig zu diesem Attribut gibt es noch eine abgeleitete Eigenschaft namens `formattedDate` (Z. 45-48). Es ist eine Hilfsmethode, die das letzte Bearbeitungsdatum in ein für Menschen lesbares Datumsformat umwandelt. In dem Übersichts-Bildschirm Abbildung 7 ist das Datumsformat sichtbar.

Da diese Getter-Methode eine Implementierung besitzt, wird für sie von „built_value“ kein Quellcode für die Serialisierung generiert.

Bevor die Werte-Typen serialisiert werden können, muss `built_value` jedoch noch mitgeteilt werden, für welche Werte-Typen Serialisierungs-Funktionen generiert werden sollen. Dazu werden über die Annotation `@SerializersFor` die gewünschten Klassen aufgelistet (Listing. 23 Z. 10). Die Zeilen 11 und 12 sind dabei immer gleich, es sei denn, es ist ein anderer Se-

rialisierung Algorithmus gewünscht. In diesem Fall wird das `StandardJsonPlugin` verwendet.

```
10 @SerializersFor([Massnahme, Storage])
11 final Serializers serializers =
12     (_$serializers.toBuilder()..addPlugin(StandardJsonPlugin())).build();
```

Listing 23: Der Serialisierer für Massnahme und Storage, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/serializers.dart](#)

Wird nun der Befehl `flutter pub run build_runner build` ausgeführt, so wird der Quellcode generiert und die Werte-Typen können für die Serialisierung genutzt werden.

5.3 Test der Serialisierung einer Maßnahme

Das Ergebnis der Serialisierung wird im dazugehörigen Unit-Test ersichtlich (Listing. 24). In Zeile 8 wird ein Objekt der Klasse `Massnahme` instanziiert. Anders als bei gewöhnlichen Datentypen lassen sich bei diesem unveränderlichen Datentyp keine Attribute nach der Erstellung anpassen. Die einzige Möglichkeit besteht darin, ein neues Objekt mit dem gewünschten Attributwert zu erstellen und die restlichen Werte des alten Objektes zu übernehmen. Dies ist mit Hilfe des sogenannten Builder-Entwurfsmuster es möglich, welches in `built_value` Anwendung findet.

```
6 test('Massnahme serialises without error', () {
7   var massnahme = Massnahme();
8   massnahme = massnahme
9       .rebuild((b) => b.identifikatoren.massnahmenTitel = "Massnahme 1");
10
11   var actualJson = serializers.serializeWith(Massnahme.serializer, massnahme);
12
13   var expectedJson = {
14     'guid': massnahme.guid,
15     'letzteBearbeitung': {
16       'letztesBearbeitungsDatum': massnahme
17         .letzteBearbeitung.letztesBearbeitungsDatum.microsecondsSinceEpoch,
18       'letzterStatus': 'bearb'
19     },
20     'identifikatoren': {'massnahmenTitel': 'Massnahme 1'}
21   };
22
23   expect(actualJson, equals(expectedJson));
```

Listing 24: Serialisierung einer Maßnahme Unittest, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/test/data_model/massnahme_test.dart](#)

Erbauer-Entwurfsmuster Das Erbauer-Entwurfsmuster - englisch builder pattern - ist ein Erzeugungsmuster, welches die Konstruktion komplexer Objekte von ihrer Repräsentation trennt. Es gehört zu der Serie von Entwurfsmustern der Gang of Four.³³ Im Fall von `built_value` trennt es die unveränderlichen Objekte von ihrer Konstruktion. Über den Builder lassen sich Änderungen an diesen unveränderlichen Objekten vornehmen, wodurch eine Kopie dieses unveränderlichen Objekt mit der gewünschten Änderung, zurückgegeben wird.

In den Zeilen 9 bis 10 wird so ein neues Objekt von der Klasse `Maßnahme` mit Hilfe der Methode `rebuild` erzeugt und anschließend der Referenz `massnahme` zugewiesen, wodurch sie ihren alten Wert verliert. Über die generierte Methode `serializers.serializeWith` kann das Objekt in Json übersetzt werden. Der erste Parameter `Massnahme.serializer` gibt dabei

³³Vgl. Gamma u. a., *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*, S. 119.

an, wie diese Serialisierung erfolgen soll und auch dieses Objekt wurde von „built_value“ generiert. Der zweite Parameter ist die tatsächliche `massnahme`, die in Json umgewandelt werden soll. Die Zeilen 13 bis 21 erstellen das Json Dokument, mit dem das serialisierte Ergebnis am Ende verglichen werden soll. Dabei werden die gleichen Eigenschaften eingetragen. So etwa die `guid` (Z. 14), welcher bei der Initialisierung der Maßnahme automatisch und zufällig erstellt wurde. Außerdem das letzte Bearbeitungsdatum, welches den Zeitstempel erhält, zudem die Maßnahme generiert wurde. Da `built_value` bei der Serialisierung die Datumswerte in Mikrosekunden umwandelt, muss für das erwartete Json-Dokument das gleiche passieren (Z. 16-17). Der `'letzterStatus'` (Z. 18) wird hierbei auf den Standardwert `'bearb'` gesetzt und der `'massnahmenTitel'` (Z. 20) auf den gleichen Wert, der in Zeile 9 übergeben wurde. Schließlich vergleicht die Methode `expect` das tatsächlich serialisierte Json-Dokument mit dem, welches zuvor zum Vergleich aufgebaut wurde. Der zweite Parameter ist ein sogenannter `Matcher` und die Variante mit dem Namen `equals` überprüft auf absolute Gleichheit.

Analog zur Serialisierung testet der Unit-Test in Listing 25 auch die Deserialisierung. Das Json-Dokument ist dabei sehr ähnlich und unterscheidet sich lediglich in zwei Details. Der `'guid'` wird auf einen festen Wert festgelegt (Z. 38), statt - wie zuvor - durch den in dem Initialisierungs-Prozess der Maßnahme zufällig generiert zu werden. Außerdem wird auch das `letztesBearbeitungsDatum` festgesetzt, nämlich auf die Microsekunde `0` (Z. 40).

```

36 test('Massnahme deserialises without error', () {
37   var json = {
38     'guid': "test massnahme id",
39     'letzteBearbeitung': {
40       'letztesBearbeitungsDatum': 0,
41       'letzterStatus': 'bearb'
42     },
43     'identifikatoren': {'massnahmenTitel': 'Massnahme 1'}
44   };
45
46   var expectedMassnahme = Massnahme((b) => b
47     ..guid = "test massnahme id"
48     ..identifikatoren.massnahmenTitel = "Massnahme 1"
49     ..letzteBearbeitung.update((b) {
50       b.letztesBearbeitungsDatum =
51         DateTime.fromMicrosecondsSinceEpoch(0, isUtc: true);
52     }));
53   var actualMassnahme =
54     serializers.deserializeWith(Massnahme.serializer, json);
55
56   expect(actualMassnahme, equals(expectedMassnahme));

```

Listing 25: Deserialisierung einer Maßnahme Unittest, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/test/data_model/massnahme_test.dart](#)

Zum Vergleich wird in den Zeilen 46 bis 52 eine Maßnahme über das Builder-Entwurfsmuster generiert und die gleichen feste Werte werden für die Eigenschaften übergeben. Dabei ist darauf zu achten, dass die Instanzvariable `letzteBearbeitung` keinen Wert über den Zuweisungs-Operator `=` erhält, sondern stattdessen die Methode `update` darauf aufgerufen wird.

Da es sich bei der Instanzvariable `letzteBearbeitung` genauso um ein Object eines Wertetypen - handelt, ist sie ebenso unveränderlich. Deshalb kann sie nur über einen Builder manipuliert werden. Ein Blick in den generierten Quellcode offenbart, dass es sich in Wahrheit um einen Bilder handelt (Listing. 26 Z. 224-225).

Außerdem müssen die Microsekunden für das Datum zunächst in ein Objekt von `DateTime` umgewandelt werden muss. Dafür wird der benannte Konstruktor `fromMillisecondsSinceEpoch`

```

216 class MassnahmeBuilder implements Builder<Massnahme, MassnahmeBuilder> {
217   _$Massnahme? _$v;
218
219   String? _guid;
220   String? get guid => _$this._guid;
221   set guid(String? guid) => _$this._guid = guid;
222
223   LetzteBearbeitungBuilder? _letzteBearbeitung;
224   LetzteBearbeitungBuilder get letzteBearbeitung =>
225     _$this._letzteBearbeitung ??= new LetzteBearbeitungBuilder();
226   set letzteBearbeitung(LetzteBearbeitungBuilder? letzteBearbeitung) =>
227     _$this._letzteBearbeitung = letzteBearbeitung;

```

Listing 26: Instanzvariable `letzteBearbeitung` gibt einen `LetzteBearbeitungBuilder` zurück, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/massnahme.g.dart](#)

von `DateTime` (Z. 51) aufgerufen.

Benannte Konstruktoren In Programmiersprachen wie beispielsweise Java können Methoden überladen werden, indem ihre Methodensignatur geändert wird. Beim Aufruf der Methode kann über die Anzahl und die Typen der übergebenen Argumente die gewünschte Methode gewählt werden. Das gleiche gilt für Konstruktoren. Wird ein weiterer Konstruktor für eine Klasse in Java benötigt, so besteht einzig und allein die Möglichkeit darin, den Konstruktor zu überladen. Sowohl überladene Methoden als auch überladene Konstruktoren existieren in Dart nicht. Wird also in dart ein Alternative constructor gewünscht, so muss er einen Namen bekommen. Beim Aufruf des Konstruktors wird dieser Name dann mit einem `.` nach dem Klassennamen angegeben, um den gewünschten Konstruktor zu benennen.

Ganz ähnlich wie bei der Serialisierung wird nun mit dem Befehl `serializers . deserializeWith` unter Angabe des Objektes, welches die Deserialisierung übernehmen soll - nämlich wiederum `Massnahme.serializer` - das Json-Dokument in ein Objekt des Wert-Typs `Massnahme` deserialisiert (Z. 53-54). Schließlich wird in Zeile 56 das Ergebnis der Deserialisierung mit dem gewünschten Ergebnis verglichen.

Gibt man in der Kommandozeile den Befehl `flutter test test /data_model /massnahme_test.dart` ein, so werden die Tests in der Testdatei ausgeführt. Die Ausgabe `00:01 +2: All tests passed!` teilt mit, dass beide Tests ausgeführt und beide Ergebnisse mit den verglichenen Werten übereinstimmen.

5.4 Serialisierung der Maßnahmenliste

Damit alle Maßnahmen - statt nur einer einzigen - in einer Datei zusammengefasst werden können, müssen die Maßnahmen zunächst zu einer Menge zusammengefasst werden, die ebenfalls serialisierbar ist. Der Wert-Typ `Storage` ist dafür vorgesehen (Listing. 27). Er deklariert allein das `BuiltSet massnahmen` (Z. 10). Ein `BuiltSet` ist die Abwandlung eines gewöhnlichen Sets, jedoch unter anderem mit der Möglichkeit, es mit einem Builder zu erstellen und das Set zu serialisieren. Die Übergabe des Typarguments `<Massnahme>` gewährleistet, dass keine anderen Objekte eingefügt werden können, die weder eine Instanz der Klasse `Massnahme` sind, oder einer Klasse, die von `Massnahmen` erbt.

Der Befehl `flutter pub run build_runner build` stößt erneut die Quellcodegenerierung für den Wert-Typen `Storage` an.

```

9 abstract class Storage implements Built<Storage, StorageBuilder> {
10   BuiltSet<Massnahme> get massnahmen;

```

Listing 27: Der Werte-Typ `Storage`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/storage.dart](#)

5.5 Test der Serialisierung der Maßnahmenliste

Nun soll noch überprüft werden, ob die Menge von Maßnahmen mit genau einer eingetragenen Maßnahme korrekt serialisiert. Auch das wird von einem Unit Test überprüft (Listing. 28). In Zeile 8 wird das leere Objekt `storage` erstellt. In Zeile 9 wird es dann wiederverwendet, um aufbauend auf der Kopie Änderungen mithilfe der `rebuild`-Methode durchzuführen.

```

7 test('Storage with one Massnahme serialises without error', () {
8   var storage = Storage();
9   storage = storage.rebuild((b) => b.massnahmen.add(
10     Massnahme((b) => b.identifikatoren.massnahmenTitel = "Massnahme 1")));
11
12   var actualJson = serializers.serializeWith(Storage.serializer, storage);
13
14   var expectedJson = {
15     "massnahmen": [
16       {
17         "guid": storage.massnahmen.first.guid,
18         "letzteBearbeitung": {
19           "letztesBearbeitungsDatum": storage
20             .massnahmen
21             .first
22             .letzteBearbeitung
23             .letztesBearbeitungsDatum
24             .microsecondsSinceEpoch,
25         "letzterStatus": "bearb"
26       },
27       "identifikatoren": {"massnahmenTitel": "Massnahme 1"}
28     ]
29   };
30   expect(actualJson, equals(expectedJson));
31

```

Listing 28: Ein automatisierter Testfall überprüft, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/test/data_model/storage_test.dart](#)

Bei der Instanzvariable `massnahmen` der Klasse `Storage` handelt es sich um ein `BuiltSet`. Der Aufruf von `b.massnahmen` gibt allerdings nicht dieses `BuiltSet` zurück. Wäre es so, so könnte die Operation `add` nicht darauf angewendet werden. Ein `BuiltSet` stellt keine Methoden zur Manipulation des Sets zur Verfügung. In Wahrheit gibt der Ausdruck `b.massnahmen` einen `SetBuilder` zurück. Das kann im generierten Quellcode nachgesehen werden (Listing. 29 Z. .)

```

91 class StorageBuilder implements Builder<Storage, StorageBuilder> {
92   _$Storage? _$v;
93
94   SetBuilder<Massnahme>? _massnahmen;
95   SetBuilder<Massnahme> get massnahmen =>
96     _$this._massnahmen ??= new SetBuilder<Massnahme>();

```

Listing 29: Instanzvariable `massnahmen` gibt einen `SetBuilder` zurück, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_model/storage.g.dart](#)

Der `SetBuilder` wiederum erlaubt es, Änderungen am Set vorzunehmen und stellt dafür die

- für ein Set übliche - Methode `add` bereit. Im Aufruf von `add` wird dann ein Objekt des Werte-Typs `Maßnahme` konstruiert (Z. 10). Dazu wird dieses Mal die anonyme Funktion zum Konstruieren der `Maßnahme` gleich im Konstruktor übergeben.

Diesmal konstruiert die Methode `serializers.serializeWith` mit dem Serialisierer `Storage.serializer` ein weiteres JSON-Object (Z. 12). Genau wie zuvor wird ein `Json-Dokument` vorbereitet (Z. 14-30), welches der `Matcher equals` gegen das serialisierte Dokument des soeben konstruierten Objects `storage` vergleicht (Z. 31). Das `Json-Dokument` unterscheidet sich nur darin, dass es einen Knoten namens `'massnahmen'` enthält, der als Wert eine Liste hat. Die Liste hat nur ein Element. Weil dieses Mal das Objekt des Typs `Massnahme` nicht direkt zugreifbar ist, muss es zunächst über die Liste der `Maßnahmen` aus dem `storage`-Object abgerufen werden. Das ist mit dem Befehl `first` möglich, der das erste Objekt - und in diesem Fall einzige Objekt - der Kollektion zurückgibt (Z. 17, 21). Darüber kann erneut die `guid` und das `letztesBearbeitungsDatum` abgerufen werden.

Ein weiterer Unit-Test überprüft, ob auch die Deserialisierung eines `storage`-Objects erfolgreich ist. Er ist in Listing 148 im Anhang D zu finden. Auch dieser Test ist der Deserialisierung des Objektes des Typs `Massnahme` sehr ähnlich. Er unterscheidet sich nur darin, dass das `Massnahme`-Objekt in der Liste `massnahmen` des `storage`-Objektes enthalten ist.

5.6 Der Haupteinstiegspunkt

Das Listing 30 zeigt den Haupteinstiegspunkt des Programms. Darin ist erkennbar, dass sich die Applikation in drei Rubriken einteilen lässt:

- das Model (Z. 27-30)
- der View (Z. 41-44)
- das ViewModel. (Z. 25-26)

Model View ViewModel Das `ModelViewViewModel` Entwurfsmuster wurde zunächst von John Gossman für die `Windows Presentation Foundation` beschrieben. Das Model beschreibt die Datenzugriffs-Komponente welche die Daten in relationalen Datenbanken oder hierarchischen Datenstrukturen wie XML oder JSON ablegt. Der View beschreibt die Oberflächenelemente wie Texteingabefelder und Buttons. Diese beiden Komponenten sind auch aus dem `ModelViewController` Entwurfsmuster bekannt. Das `ModelViewViewModel` Entwurfsmuster ist eine Weiterentwicklung davon und integriert das sogenannte `ViewModel`. Es ist dafür zuständig als Schnittstelle zwischen View und Model zu fungieren. Die Daten des Models lassen sich in der Regel nicht direkt mit Oberflächen Elementen verknüpfen. Denn es kann es notwendig sein, dass die Oberfläche weitere temporäre Daten benötigt, die aber nicht mit den Daten des Models gespeichert werden sollen. Das `ViewModel` übernimmt diese Arbeit, indem es die Daten des Models abrufen und sie in veränderter Form den Oberflächen-Elementen zur Verfügung steht. Andersherum formt es die Eingaben in der Nutzeroberfläche so um, dass sie im strikten Datenmodell des Models Platz finden.³⁴

`MassnahmenModel` (Z. 29) verwaltet die eingegebenen Daten der `Maßnahmen` und nutzt die Abhängigkeit `MassnahmenJsonFile` (Z. 27) um die Daten auf einem Datenträger als eine `JSON-Datei` zu speichern. Somit gehören diese beiden Klassen dem Model an.

`MassnahmenFormViewModel` (Z. 25) greift die Daten des Models ab und formt diese um, so dass sie von dem View `MassnahmenDetailScreen` (Z. 43) verändert werden können. Sollen

³⁴Vgl. Gossman, *Introduction to Model/View/ViewModel pattern for building WPF apps*.

```

18 class MassnahmenFormApp extends StatelessWidget {
19   const MassnahmenFormApp({Key? key}) : super(key: key);
20
21   @override
22   Widget build(BuildContext context) {
23     return AppState(
24       model: MassnahmenModel(MassnahmenJsonFile()),
25       viewModel: MassnahmenFormViewModel(),
26       child: MaterialApp(
27         title: 'Maßnahmen',
28         theme: ThemeData(
29           primarySwatch: Colors.lightGreen,
30           accentColor: Colors.green,
31           primaryIconTheme: const IconThemeData(color: Colors.white),
32         ),
33         initialRoute: MassnahmenMasterScreen.routeName,
34         routes: {
35           MassnahmenMasterScreen.routeName: (context) =>
36             const MassnahmenMasterScreen(),
37           MassnahmenDetailScreen.routeName: (context) =>
38             const MassnahmenDetailScreen()
39         },
40       ));
41   }
42 }

```

Listing 30: Der Haupteinstiegspunkt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/main.dart](#)

die Daten gespeichert werden, so stellt `MassnahmenFormViewModel` ebenfalls Methoden zur Verfügung um die Daten wieder in das Format des Models einpflegen zu können.

`MassnahmenMasterScreen` (Z. 41) stellt eine Ausnahme dar, denn dieser View präsentiert die Daten aus dem Model ohne eine Schnittstelle über ein ViewModel. Das ist möglich, weil die Daten nicht manipuliert, sondern nur angezeigt werden müssen.

Damit sowohl ViewModel als auch Model von jedem View heraus abrufbar sind, werden sie in eine Art Service eingefügt (Z. 23). Das Widget AppState ist dieser Service. Er erhält das Model (Z. 24) und das ViewModel (Z. 25) im Konstruktor. Die Abhängigkeit zum Schreiben des Models in eine JSON Datei `MassnahmenJsonFile` bekommt das Model ebenfalls im Konstruktor übergeben (Z. 24). `AppState` ist das erste Element, welches im Widget-Baum auftaucht. Die gesamte restliche Applikation ist als Kind-Element hinterlegt (Z. 26). Damit können alle Widgets auf den Service zugreifen.

Service Locator und Dependency Injection Das Service Locator Entwurfsmuster folgt dem Umsetzungsparadigma Inversion of Control – deutsch Umkehrung der Steuerung. Frameworks folgen diesem Muster, indem sie als erweiterbare Skelett-Applikationen fungieren. Anstatt, dass die Applikation den Programmfluss steuert und dabei selbst Funktionen aufruft, wird die Programmflusssteuerung an das Framework abgegeben und mit Hilfe von Ereignissen ermöglicht, dass das Framework Funktionen des Nutzers aufruft.³⁵ Das Entwurfsmuster ist auch unter dem Namen „hollywood pattern“ bekannt, da es ähnlich wie die typische Antwort auf eine Bewerbung für einen Hollywood Film - don't call us, we'll call you - funktioniert.³⁶ Im Service Locator Entwurfsmuster werden Komponenten darüber hinaus zentral registriert und über dieses Register anderen Komponenten zur Interaktion zur Verfügung gestellt.³⁷ Anstatt die Komponenten direkt miteinander zu

³⁵Vgl. Johnson und Foote, „Designing reusable classes“.

³⁶Vgl. Fowler, *InversionOfControl*.

³⁷Vgl. Fowler, *Inversion of Control Containers and the Dependency Injection pattern*.

verknüpfen, werden Sie für den Zugriff von praktisch überall vorbereitet. Vor allem für automatisierte Tests ist dies von Vorteil, da solche Abhängigkeiten ausgetauscht werden können, um ganz spezielle Teil-Funktionalitäten eines Programms zu testen. Mehr dazu im Kapitel **Kapitel einfügen**.

Anders als der Name vermuten lässt, steuert `MaterialApp` nicht nur das Aussehen der Applikation im Material Design Look. Darüber hinaus stellt das Widget auch Grundfunktionalitäten einer App, wie etwa den Navigator bereit. Damit hat die Applikation die Möglichkeit – ähnlich wie bei einer Website – auf Unterseiten zu navigieren. Hat der Benutzer die Arbeit in der Unterseite vollendet, so kann der Navigator gebeten werden, zur vorherigen Ansicht zurückzukehren. Mit dem Parameter `routes` (Z. 34-39) erfolgt die Angabe der Unterseiten, die besucht werden können. Über `initialRoute` (Z. 39) kann die Startseite angegeben werden.

5.7 Der Service für den Applikations übergreifenden Zustand

Um Daten an alle Kind Elementen im Widgets mitzugeben, finden die sogenannten „InheritedWidgets“ Anwendung. Der Service `AppState` 31 ist genauso ein solches.

```
5 class AppState extends InheritedWidget {
6   const AppState({
7     Key? key,
8     required this.model,
9     required this.viewModel,
10    required Widget child
11  }) : super(key: key, child: child);
12
13   final MassnahmenFormViewModel viewModel;
14   final MassnahmenModel model;
15
16   static AppState of(BuildContext context) {
17     final AppState? result = context.dependOnInheritedWidgetOfExactType<AppState>();
18     assert(result != null, "Kein AppState im 'context' gefunden");
19     return result!;
20   }
21
22   @override
23   bool updateShouldNotify(covariant AppState oldWidget) => false;
24 }
```

Listing 31: Der Service `AppState`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/widgets/app_state.dart](#)

Im Konstruktor erhält er zunächst bei den Parameter des Typs `key` (Z. 7). Es ist gängige Praxis in Flutter, jedem Widget im Konstruktor zu ermöglichen, einen solchen Schlüssel zu übergeben. Es ist jedoch optional. Ein solcher Schlüssel kann genutzt werden, um das Widget eindeutig zu identifizieren und es unter anderem über den Schlüssel wiederzufinden.

In den Zeilen 8 und 9 werden das Model und das ViewModel dem Objekt im Konstruktor übergeben. In den Zeilen 14 und 15 sind sie deklariert.

Das letzte Element im Konstruktor ist das `child`. Ihm muss der Widget-Baum übergeben werden, dem der Zustand verfügbar gemacht werden soll.

Der Aufruf des Basis-Konstruktors mit den Argumenten `key` und `child` ist in Zeile 11 zu sehen.

Die Basisklasse von `InheritedWidget` ist `ProxyWidget` und erhält exakt dieselben Argumente. Das `ProxyWidget` verwendet das Kindelement, um es im WidgetBaum unterhalb

von sich selbst zu zeichnen. Eine eigene Methode zum Zeichnen musst also nicht für das `InheritedWidget` implementiert werden. Die einzige Methode, welche implementiert werden muss, ist `updateShouldNotify` (Z. 24). Immer dann, wenn das `InheritedWidget` selbst aktualisiert wird, kann es alle Widgets, die davon abhängig sind, benachrichtigen. In dem Fall werden diese Widgets ebenfalls neu gezeichnet. Für die Formular-Applikation ist das allerdings nicht gewünscht. Die Aktualisierung der Oberfläche soll in den nachfolgenden Schritten selbst kontrolliert werden. Deshalb erfolgt die Rückgabe `false`, da in Zukunft nicht gewünscht ist, den Applikations-Zustand komplett auszutauschen. Um die Aktualisierung der Oberfläche kümmern sich sowohl Model als auch ViewModel.

Damit ein Widget Abhängigkeit von dem `AppState` anmelden kann, verwendet es in seiner eigenen `build`-Methode die Methode `dependOnInheritedWidgetOfExactType<AppState>()`. Der Aufruf der Methode erfolgt auf dem Objekt vom Typ `BuildContext`. Weil dieser Kontext bei jedem Zeichnen allen Kindern übergeben wird, kann jedes Kind darüber die Vater-Elemente wiederfinden.

Damit der Aufruf leichter lesbar und kürzer ist, empfiehlt das Flutter-Team eine eigene Klassenmethode zu erstellen, welche die Methode für den Benutzer aufruft (Z. 16-17). Auch eine Fehlermeldung kann bei dieser Auslagerung geworfen werden, sollte im Kontext kein Objekt des gewünschten Typs vorhanden sein (Z. 18). Das Widget, welches auf den `AppState` zugreifen möchte, kann es mit der einfachen Schreibweise `AppState.of(context)` abrufen.

5.8 Speichern der Maßnahmen in eine Json-Datei

Das Model wird durch die Klasse `MassnahmenJsonFile` in eine JSON-Datei gespeichert (Listing. 32). Der Dateipfad wird dabei durch die Methode `_localMassnahmenJsonFile` (Z. 8-11) abgerufen. Die Hilfsmethode `getApplicationSupportDirectory` (Z. 9) gibt aus dem Nutzerverzeichnis des aktuellen Nutzers den zur Applikation zugeordneten Datei-Ordner zurück. Auf Windows-Betriebssystemen wäre das beispielsweise `C:\Users\AktuellerNutzer\AppData\Roaming\com.example\conditional_form`.

Dadurch, dass dem Methoden-Bezeichner `_localMassnahmenJsonFile` ein Unterstrich vorangestellt ist, ist die Methode privat und kann nur innerhalb der Klasse aufgerufen werden. Dart hat damit eine Konvention zum Standard werden lassen. In Programmiersprachen wie beispielsweise C++ wurde der Unterstrich zusätzlich den Bezeichnern von Instanz Attributen vorangestellt, die mit dem `private` Schlüsselwort gekennzeichnet sind, damit sie überall im Quellcode als private Attribute identifizierbar sind, ohne dazu die Klassendefinition ansehen zu müssen. In Dart gibt es dagegen das `private` Schlüsselwort nicht. Stattdessen wird der Unterstrich vor dem Bezeichner verwendet, um ein Instanzattribut privat zu deklarieren.

Die Getter-Methode `_localMassnahmenJsonFile` hat den Rückgabotyp `Future<File>` und ist zudem mit dem Schlüsselwort `async` gekennzeichnet. Asynchron muss die Methode deshalb sein, weil sie auf den Aufruf `getApplicationSupportDirectory` warten muss, der ebenfalls asynchron abläuft.

Der Funktion `saveMassnahmen` (Z. 13-16) wird ein JSON Objekt in Form einer Hashtabelle übergeben. Sie ruft die Hilfs-Getter-Methode `_localMassnahmenJsonFile` (Z. 14) auf und schreibt den Dateinhalt in die Datei des abgefragten Pfades (Z. 15). Zuvor wird dazu das JSON-Object in eine textuelle Repräsentation überführt. Dazu dient die Funktion `jsonEncode`.

Das Äquivalent dazu stellt die Methode `readMassnahmen` (Z. 18-30) dar. Auch sie ruft den

```

7 class MassnahmenJsonFile {
8   Future<File> get _localMassnahmenJsonFile async {
9     var directory = await getApplicationSupportDirectory();
10    return File("${directory.path}/Maßnahmen.json");
11  }
12
13  Future<void> saveMassnahmen(Map<String, dynamic> massnahmenAsJson) async {
14    var file = await _localMassnahmenJsonFile;
15    await file.writeAsString(jsonEncode(massnahmenAsJson));
16  }
17
18  Future<Map<String, dynamic>> readMassnahmen() async {
19    var file = await _localMassnahmenJsonFile;
20
21    var fileExists = await file.exists();
22    if (fileExists) {
23      final fileContent = await file.readAsString();
24      final jsonObject = jsonDecode(fileContent) as Map<String, dynamic>;
25
26      return jsonObject;
27    } else {
28      throw MassnahmenFileDoesNotExistException("$file was not found");
29    }
30  }
31 }

```

Listing 32: Die Klasse `MassnahmenJsonFile`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/persistence/massnahmen_json_file.dart](#)

Dateipfad ab (Z. 19), überprüft allerdings im nächsten Schritt, ob die Datei bereits existiert (Z. 21). Sollte das der Fall sein, so wird die Datei eingelesen (Z. 23). Die textuelle Repräsentation aus der Datei wird mittels Methode `jsonDecode` in ein Json-Objekt in der Form einer Hashtabelle gespeichert (Z. 24) und schließlich zurückgegeben (Z. 26). Sollte die Dateien nicht existieren, führt das zu einer Ausnahme (Z. 28), welche von der aufrufenden Funktion behandelt werden kann.

5.9 Abhängigkeit zum Verwalten der Maßnahmen

Die Art und Weise, wie die Maßnahmen abgerufen werden, sollte nach Möglichkeit abstrahiert werden. Das erlaubt, den Mechanismus in Zukunft auszutauschen, ohne dabei den Rest der Applikation verändern zu müssen. So wäre es beispielsweise denkbar, statt einer Json-Datei eine direkte Verbindung zu einer relationalen Datenbank herzustellen. Auch das Austauschen der Abhängigkeit mit einem Platzhalter, der lediglich die Aufrufe der Methoden zählt, ist damit möglich. Ein solches Platzhalterobjekt wird „Mock“ genannt und für automatisiertes Testen eingesetzt (siehe Kapitel [Kapitel einfügen](#)). Ebenso abstrahiert werden soll der Umgang mit Ausnahmen. Sollte die Datei nicht verfügbar sein, so muss die Oberfläche davon nicht zwingend betroffen sein. Stattdessen kann der Service sich entscheiden, eine leere Liste von Maßnahmen zurückzugeben. Sobald die Liste manipuliert wird, kann eine neue Datei angelegt werden und sie mit den eingegebenen Daten beschreiben. Die Klasse `MassnahmenModel` (Listing. 33) tut genau das.

Sie bekommt `MassnahmenJsonFile` im Konstruktor übergeben (Z. 11). Daraufhin ruft der Konstrukteur gleich die `init` auf (Z. 12), welche in den Zeilen 15-22 deklariert ist. Darin wird der Stream `storage` (Z. 19) initialisiert. Es handelt sich um eine Erweiterung eines „broadcast streams“ mit dem Namen `BehaviorSubject` (Z. 9). Es entstammt dem Paket `rx.dart`, welches die Streams in Dart um eine Reihe von weiteren Funktionalitäten erweitert. Ein `BehaviorSubject` hat die Besonderheit, dass es den Wert des letzten Ereignisses

```

7 class MassnahmenModel {
8   final MassnahmenJsonFile jsonFile;
9   final storage = BehaviorSubject<Storage>.seeded(Storage());
10
11   MassnahmenModel(this.jsonFile) {
12     init();
13   }
14
15   init() async {
16     try {
17       final massnahmenAsJson = await jsonFile.readMassnahmen();
18
19       storage.value =
20         serializers.deserializeWith(Storage.serializer, massnahmenAsJson!);
21     } on MassnahmenFileDoesNotExistException {}
22   }
23
24   putMassnahmeIfAbsent(Massnahme massnahme) async {
25     storage.value = storage.value.rebuild((b) => b.massnahmen
26       ..removeWhere((m) => m.guid == massnahme.guid)
27       ..add(massnahme));
28
29     var serializedMassnahmen =
30       serializers.serializeWith(Storage.serializer, storage.value);
31
32     await jsonFile.saveMassnahmen(serializedMassnahmen as Map<String, dynamic>);
33   }
34 }

```

Listing 33: Die Klasse `MassnahmenModel`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/data_access/massnahmen_model.dart](#)

zwischenspeichern. Die „broadcast streams“ haben für gewöhnlich den Nachteil, das neue Zuhörer des Streams nur die neuen Ereignisse erhalten. Alle in der Vergangenheit erfolgten Ereignisse sind nicht mehr verfügbar. Vor allem dann, wenn in der Oberfläche der letzte Wert eines Streams verwendet werden soll, um Elemente zu zeichnen, ist das von einem besonderen Nachteil. Denn wenn der Stream zuvor initialisiert wurde, so gibt es keine Daten zu dem Zeitpunkt, wenn die Oberfläche gezeichnet wird. Sollte die Oberfläche jedoch gezeichnet werden, bevor der Stream initialisiert wurde, so existieren ebenfalls keine Daten. Hier kommt das `BehaviorSubject` ins Spiel. Sobald die Oberfläche gezeichnet wird und der Stream bereits initialisiert ist, kann dennoch auf den zuletzt übertragenen Wert zurückgegriffen werden. Anschließend überträgt der Stream die folgenden Aktualisierungen für die Oberfläche mit jedem neuen Ereignis, so wie es für Streams üblich ist.

Der Stream kann nicht bereits in der Initialisierungsliste des Konstruktors mit den Daten aus der Json-Datei gefüllt werden. Das liegt daran, dass die Json-Daten dazu zunächst gelesen werden müssen, was nur durch eine Reihe von asynchronen Operation möglich ist. In einer Initialisierungsliste können allerdings keine asynchronen Operationen ausgeführt werden. Deshalb wird `init` erst im Konstruktor-Körper aufgerufen (Z. 7).

Damit der Stream anfangs nicht leer ist, füllt ihn der benannte Konstruktor `seeded` mit einem leeren Objekt des Typs `Storage` (Z. 9). Sobald die Datei gelesen (Z. 17) und anschließend deserialisiert wurde (Z. 20), erhält der Stream über die Setter-Methode `value` ein neues Ereignis mit dem gelesenen Wert (Z. 19).

Die Initialisierung ist von einem `try`-Block umgeben. Sollte die Initialisierung versagen, weil die Json-Datei nicht existiert, wird die entsprechende Fehlerbehandlung ausgeführt (Z. 21). Diese ist leer, da sich im Stream bereits ein leeres `Storage`-Objekt befindet. Mit diesem leeren Objekt kann die Oberfläche weiterarbeiten. In Zukunft könnte es sinnvoll sein, innerhalb der Fehlerbehandlung eine Meldung an den Benutzer zu geben, um darüber

zu informieren, dass eine neue Datei angelegt wurde.

Mit `putMassnahmeIfAbsent` (Z. 24-33) steht eine Methode bereit, um gleichzeitig sowohl die Oberfläche, als auch die Json-Datei zu aktualisieren. Sollte die eingetragene Maßnahme schon existieren, wird sie zunächst gelöscht (Z. 26). In jedem Fall wird die neue Maßnahme dem Stream hinzugefügt (Z. 27). Durch Austauschen des gesamten Objektes mit der Zuweisung von `storage.value` (Z. 25) erhält der Stream erneut ein neues Ereignis, womit er die Oberfläche benachrichtigen kann, sich neu zu zeichnen. Außerdem wird die Serialisierung des `Storage`-Objektes und angestoßen (Z. 29-30) und die neue Liste von Maßnahmen im darauffolgenden Schritt zurück in die Json-Datei gespeichert (Z. 32).

5.10 Übersichtsbildschirm der Maßnahmen

Der erste Bildschirm - die Übersicht der Maßnahmen - kann auf das im letzten Schritt erstellten Model zugreifen. In Listing 34 ist die Struktur des Übersicht-Bildschirms zu sehen. Über die Route `/massnahmen_master` ist der Bildschirm erreichbar (Z. 16). Die `build`-Methode zeichnet die Oberfläche 21-111. Da ein Objekt des Typs `MassnahmenPool` im zentralen Register der Provider hinterlegt wurde, kann mit der Methode `Provider.of` darauf zugegriffen werden.

Mittels `AppState.of(context)` ist nun der Zugriff auf sowohl Model als auch ViewModel möglich. Zur einfacheren Verwendung sind sie als lokale Variablen zwischengespeichert (Z. 20-21).

Das Widget `Scaffold` - deutsch Gerüst - stellt ein grundlegendes Layout mit einer Überschrift und einem Bereich für den Inhalt bereit (Z. 23). Das `Scaffold` kann auch Mitteilungen an den Benutzer am unteren Bildschirmrand einblenden.

Die Überschrift wird in der sogenannten `AppBar` hinterlegt (Z. 24). Sie unterstützt weitere Funktionalitäten. Sollte es sich bei der aktuell besuchten Route um eine Unterseite handeln, taucht links von der Titel-Überschrift einen Button zum Zurücknavigieren auf. Weiterhin können rechts von der Titelleiste Aktionsbuttons hinzugefügt werden, welches für die Formular Anwendung allerdings nicht nötig ist.

Zusätzlich kann dem `Scaffold` ein Button für die primäre Aktion auf diesem Bildschirm hinzugefügt werden: der sogenannte `FloatingActionButton` (Z. 88-97). Bei Aktivierung des Buttons navigiert die Applikation zur Eingabemaske, um eine neue Maßnahme anzulegen (Z. 98).

Das Eingabeformular sollte den Benutzer auffordern, tatsächlich leere Eingabefelder zu füllen. Deshalb muss die Aktivierung des Buttons auch das ViewModel neu initialisieren. Dies geschieht durch Zuweisung einer leeren Maßnahme zu zur Setter-Methode `vm.model` (Z. 95). Ohne die Neuinitialisierung würde die Eingabemaske immer die zuletzt eingetragene Maßnahme enthalten. Dies würde große Verwirrung beim Benutzer stiften.

Der `FloatingActionButton` erhält den Schlüssel `createNewMassnahmeButtonKey` (Z. 89). Er ist als `GlobalKey` deklariert (Z. 11). Er findet beim Integrationstest Anwendung, um den Button zu finden (Siehe Kapitel `Kapitel einfügen`).

Der Inhaltsbereich des `Scaffold` beinhaltet das Widget `StreamBuilder` (Z. 27). Er kann auf Streams horchen, die Ereignisse des Typs `Storage` übermitteln. Er horcht auf Änderungen im Model, um genau zu sein auf Änderungen des Streams `model.storage` (Z. 28). Sobald der `StreamBuilder` ein Ereignis erhält, so führt er die Methode aus, die als Argument des Parameters `builder` hinterlegt ist. Alle Widgets außerhalb davon, wie etwa das `Scaffold`, erhalten dabei keine Aufforderung zum Neuzeichnen, sobald eine Maßnahme hinzugefügt

wird. Das wirkt sich positiv auf die Laufzeit-Geschwindigkeit aus.

```
11 final createNewMassnahmeButtonKey = GlobalKey();
12
13 class MassnahmenMasterScreen extends StatelessWidget {
14   static const routeName = '/massnahmen_master';
15
16   const MassnahmenMasterScreen({Key? key}) : super(key: key);
17
18   @override
19   Widget build(BuildContext context) {
20     final model = AppState.of(context).model;
21     final vm = AppState.of(context).viewModel;
22
23     return Scaffold(
24       appBar: AppBar(
25         title: const Text('Maßnahmen Master'),
26       ),
27       body: StreamBuilder<Storage>(
28         stream: model.storage,
29         builder: (context, _) {
30           return SingleChildScrollView(
31             // ...
32           );
33         },
34       ),
35       floatingActionButton: FloatingActionButton(
36         key: createNewMassnahmeButtonKey,
37         child: const Icon(
38           Icons.post_add_outlined,
39           color: Colors.white,
40         ),
41         onPressed: () {
42           vm.model = Massnahme();
43           Navigator.of(context).pushNamed(MassnahmenDetailScreen.routeName);
44         },
45       ),
46     );
47   }
48 }
49 }
```

Listing 34: Die Struktur der Klasse MassnahmenMasterScreen, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_master.dart](#)

5.10.1 Auflistung der Maßnahmen im Übersichtsbildschirm

Der Inhalt der `builder`-Methode ist in Listing 35 dargestellt. Das erste Widget ist ein `SingleChildScrollView` (Z. 30). Das Argument `scrollDirection` ist nicht gefüllt, weshalb die Standardoption - die vertikale Scrollrichtung - gewählt wird. Sollte die Liste der Maßnahmen die Höhe des Fensters überschreiten, so kann der Benutzer vertikal über die Liste scrollen.

Das Kind des Scrollbereichs ist ein `Column`-Widget (Z. 31). Sie zeichnet Widgets, die als Argument des Parameters `children` gesetzt sind, von oben nach unten. Der Parameter `crossAxisAlignment` gibt an, wie die Kindelemente ausgerichtet sein sollen. `crossAxis` bedeutet dabei die zur Anzeige-Richtung entgegengesetzten Richtung. Da die `Column` vertikal zeichnet, ist mit `crossAxis` die horizontale Achse gemeint. `CrossAxisAlignment.start` beschreibt, dass Elemente entlang der horizontalen Achse an dessen Startpunkt auszurichten sind. Dadurch sind alle Elemente der Liste linksbündig.

Zuerst kommt die Auflistung der abgeschlossenen Maßnahmen. Die Überschrift `"Abgeschlossen"` (Z. 37), soll einen Abstand von jeweils 16 Pixel in alle Richtungen haben. Das ermöglicht das Widget `Padding` (Z. 35-40) und das Argument `EdgeInsets.all(16.0)`. Nach der Überschrift erscheint als zweites Element in der `Column` ein weiterer `SingleChildScrollView` (Z. 41-57), allerdings dieses Mal mit horizontaler Scroll-Richtung (Z. 42). Sollten die Informationen der Maßnahmen die Breite des Fensters überschreiten, kann der Nutzer von links nach rechts scrollen.

Die Informationen der Maßnahmen werden in einer Tabelle angezeigt. Dies übernimmt das selbstgeschriebene „Widget“ `MassnahmenTable` (Z. 45). Als erstes Argument erfolgt die Übergabe der anzuzeigenden Maßnahmen aus dem Model. `storage.value.massnahmen` gibt den aktuellen Wert des Streams des `storage`-Objektes zurück und greift auf die Liste der Maßnahmen zu. Mit der Methode `where` (Z. 47) kann ein Filter auf die Liste angewendet werden. Die übergebene anonyme Funktion (Z. 47-49) überprüft, ob der letzte Status auf fertig gesetzt ist. Dazu reicht der Vergleich der Abkürzung. Nur wenn die Bedingung erfüllt ist, bleibt die Maßnahme in der gefilterten Kollektion zurück. Ein solcher Filter gibt ein sogenanntes „lazy“ `Iterable` zurück. Erst beim Zugriff auf das Ergebnis findet der Filter Anwendung. Doch es gibt keinen Zwischenspeicher für die gefilterten Elemente. Jeder Zugriff filtert die Elemente also neu. Der Aufruf `toSet` bewirkt allerdings das Speichern der Ergebnisse in einer Menge (Z. 50). Das Resultat erhält das Widget `MassnahmenTable` zur Anzeige.

Ein weiterer Parameter ist `onSelect` (Z. 50). Als Argument kann eine Funktion mit genau einem Parameter gesetzt werden. Sollte der Benutzer in der Tabelle eine Maßnahme auswählen, so löst er damit die Funktion aus. Der erste Parameter enthält dann die ausgewählte Maßnahme. Daraufhin soll sich wieder die Eingabemaske öffnen (Z. 55-56). Dann beinhalten die Eingabefehler jedoch die Werte der ausgewählten Maßnahme. Um das zu erreichen, reicht eine Zuweisung der Maßnahme an das ViewModel (Z. 51). Allerdings soll die Maßnahme zuvor ein neues letztes Bearbeitungsdatum mit dem aktuellen Zeitstempel erhalten (Z. 51-53).

Unterhalb der Rubrik der finalen Maßnahmen, listed die Übersicht die Maßnahmen, welche sich noch im Entwurf befinden (Z. 59-83). Daher ist das dritte Element der `Column` wiederum eine Überschrift: `"In Bearbeitung"` (Z. 62) gefolgt von einem weiteren horizontalen Scrollbereich (Z. 66-83) mit einer Tabelle von Maßnahmen (Z. 70-82). Der einzige Unterschied hier: die Bedingung der Filterfunktion. Dieses Mal filtert die Kollektion auf Maßnahmen in Bearbeitung (Z. 73-74).

```

30 return SingleChildScrollView(
31   child: Column(
32     crossAxisAlignment: CrossAxisAlignment.start,
33     children: [
34       const Padding(
35         padding: EdgeInsets.all(16.0),
36         child: Text(
37           "Abgeschlossen",
38           style: TextStyle(fontSize: 20),
39         ),
40     ),
41     SingleChildScrollView(
42       scrollDirection: Axis.horizontal,
43       child: Padding(
44         padding: const EdgeInsets.all(16.0),
45         child: MassnahmenTable(
46           model.storage.value.massnahmen
47             .where((m) =>
48               m.letzteBearbeitung.letzterStatus ==
49                 LetzterStatus.fertig.abbreviation)
50             .toSet(), onSelect: (selectedMassnahme) {
51               vm.model = selectedMassnahme.rebuild((m) => m
52                 ..letzteBearbeitung.letztesBearbeitungsDatum =
53                   DateTime.now().toUtc());
54             },
55             Navigator.of(context)
56               .pushNamed(MassnahmenDetailScreen.routeName);
57           }),
58     ),
59     const Padding(
60       padding: EdgeInsets.all(16.0),
61       child: Text(
62         "In Bearbeitung",
63         style: TextStyle(fontSize: 20),
64       ),
65     ),
66     SingleChildScrollView(
67       scrollDirection: Axis.horizontal,
68       child: Padding(
69         padding: const EdgeInsets.all(16.0),
70         child: MassnahmenTable(
71           model.storage.value.massnahmen
72             .where((m) =>
73               m.letzteBearbeitung.letzterStatus ==
74                 LetzterStatus.bearb.abbreviation)
75             .toSet(), onSelect: (selectedMassnahme) {
76               vm.model = selectedMassnahme.rebuild((m) => m
77                 ..letzteBearbeitung.letztesBearbeitungsDatum =
78                   DateTime.now().toUtc());
79             },
80             Navigator.of(context)
81               .pushNamed(MassnahmenDetailScreen.routeName);
82           }),
83     ),
84   ],
85 ),
86 );

```

Listing 35: Die Ausgabe der Maßnahmen, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_master.dart](#)

5.11 Widget MassnahmenTable

Die `MassnahmenTable` ist ein `StatelessWidget` (Listing. 36 Z. 6). Zur Anzeige eignet sich das Widget `Table` (Z. 15-31). Im Verlauf der Erstellung der Arbeit, wurde versucht das Widget `DataTable` zu verwenden. Doch im Gegensatz zur `DataTable` erlaubt es die `Table`, unterschiedlich hohe Zeilen zu zeichnen. Die Höhe der Zeile wird dazu in Abhängigkeit von dem benötigten Inhalt der Zellen berechnet. Die Breite und Ausrichtung der Spalten kann konfiguriert werden. Die Eigenschaft `IntrinsicColumnWidth` sorgt dafür, dass die Spalten immer genau so groß sind, wie der Inhalt es benötigt (Z. 17). Zeilenumbrüche für die Texte in den Spalten sind somit nicht notwendig. `TableCellVerticalAlignment.middle` lässt die Tabelle die Inhalte zentriert darstellen (Z. 18).

Der Parameter `children` erhält als Argument eine Liste von `TableRow` Elementen (Z. 20-30). Die erste Tabellenzeile 20-23 beinhaltet die Spalten-Bezeichnungen. Jede `TableRow` hat wiederum den Parameter `children`. Das Argument bezieht sich hier auf die Zellen in der Zeile. Dabei ist wichtig, dass jede `TableRow` die gleiche Anzahl von Zellen hat. Weicht nur eine Zeile davon ab, zeichnet sich die gesamte Tabelle nicht und eine Ausnahme wird ausgelöst. Für die Spaltenbezeichnungen wurde eine Hilfsmethode kreiert: `_buildColumnHeader` (Z. 34-37). Sie zeichnet die Spalten mit einem Abstand von 8 Pixel in alle Richtungen.

Nach den Spaltenbezeichnungen folgen die Zeilen für die Daten der Maßnahmen (Z. 24-29). Die Methode `map` (Z. 24) ermöglicht es dazu durch die Liste der Maßnahmen zu iterieren und für jede Maßnahme ein Element eines völlig anderen Typs - in diesem Fall `TableRow` - zurückzugeben. Bei den vorangestellten Punkten `...` in Zeile 24 handelt sich um den spread operator. Die Filtermethode `map` und die darauffolgende Methode `toList` liefert eine Liste von `TableRow` Elementen. Die umgebende Liste der Zeilen `children` (Z. 19-30) erwartet jedoch Elemente des Typs `TableRow` und keine Elemente des Typs `List`. Der spread operator ermöglicht alle Elemente der inneren Liste in die äußere Liste einzufügen.³⁸

Eine weitere Hilfsmethode `_buildSelectableCell` erstellt Zellen, die anklickbar sind (Z. 39-51). Das Widget `TableRowInkWell` (Z. 41-51) kann in Tabellen verwendet werden, um einen anklickbaren Bereich zu erstellen. Beim Anklicken breitet sich ausgehend von der Position des Klicks ein Tintenklecks aus. Dabei überschreitet der Tintenklecks nicht den Bereich, der von der umgebenden Zeile begrenzt ist. Bei auslösen des Ereignisses `onTap` erfolgt die Ausführung des Callbacks `onSelect` (Z. 44) mit der ausgewählten Maßnahme. Doch zuvor muss überprüft werden, ob der Callback auch initialisiert wurde (Z. 43). Wie hier zu sehen ist, reicht es nicht aus, abzufragen, ob `onSelect` gesetzt ist. Trotzdem erfolgt keine Typ-Beförderung zu einem Typen ohne Null-Zulässigkeit, denn es handelt sich um eine Instanzvariable. Deshalb muss der Suffix `!` gesetzt sein (Siehe Grundlagenkapitel 3.2.4 Typen mit Null-Zulässigkeit). `onSelect` ist in Zeile 8 deklariert. Es handelt sich um eine sogenannte konkrete Strategie nach dem Strategie-Entwurfsmuster

Strategie Entwurfsmuster Das Strategie Entwurfsmuster ist ein Verhaltensmuster. Es erlaubt Algorithmen zu kapseln und auszutauschen. Die Typdefinition `OnSelectCallback` (Z. 4) kann nach dem Strategie-Entwurfsmuster als die Schnittstelle namens „Strategie“ interpretiert werden. Sie definiert, welche Voraussetzung an die Schnittstelle gegeben ist. In diesem Fall ist die Voraussetzung, dass es sich um eine Funktion ohne Rückgabewert handelt, der eine Maßnahme als erstes Argument übergeben wird. Entsprechend ist der Parameter `onSelect` im Konstruktor die sogenannte „konkrete Strategie“, die dieser Schnittstelle entsprechen muss. Der „Kontext“ ist schließlich die aufrufende Oberfläche `MassnahmenMasterScreen`. Die konkrete Strategie, die der Übersichts-Bildschirm der Tabelle übergibt, verwendet die selektierte Maßnahme, um damit die Eingabemaske zu öffnen.

³⁸Vgl. Google LLC, *Dart - Language tour - spread operator*.

```

4 typedef OnSelectCallback = void Function(Massnahme selectedMassnahme);
5
6 class MassnahmenTable extends StatelessWidget {
7   final Set<Massnahme> _massnahmenToDisplay;
8   final OnSelectCallback? onSelect;
9
10  const MassnahmenTable(this._massnahmenToDisplay, {this.onSelect, Key? key})
11    : super(key: key);
12
13  @override
14  Widget build(BuildContext context) {
15    return Table(
16      border: TableBorder.all(width: 3),
17      defaultColumnWidth: const IntrinsicColumnWidth(),
18      defaultVerticalAlignment: TableCellVerticalAlignment.middle,
19      children: [
20        TableRow(children: [
21          _buildColumnHeader(const Text("Zuletzt bearbeitet am")),
22          _buildColumnHeader(const Text("Maßnahmentitel"))
23        ]),
24        ..._massnahmenToDisplay.map((m) {
25          return TableRow(children: [
26            _buildSelectableCell(m, Text(m.letzteBearbeitung.formattedDate)),
27            _buildSelectableCell(m, Text(m.identifikatoren.massnahmenTitel)),
28          ]);
29        }).toList(),
30      ],
31    );
32  }
33
34  Widget _buildColumnHeader(Widget child) => Padding(
35    padding: const EdgeInsets.all(8.0),
36    child: child,
37  );
38
39  Widget _buildSelectableCell(Massnahme m, Widget child,
40    {double padding = 8.0}) =>
41    TableRowInkWell(
42      onTap: () {
43        if (onSelect != null) {
44          onSelect!(m);
45        }
46      },
47      child: Padding(
48        padding: EdgeInsets.all(padding),
49        child: child,
50      ),
51    );
52 }

```

Listing 36: Die Klasse MassnahmenTable, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/widgets/massnahmen_table.dart](#)

5.12 Das View Model

Listing 37 Zeig das ViewModel. Im ersten Schritt enthält es nur drei Streams vom Typ `BehaviorSubject`. Eines für den letzten Status (Z. 6), eines für die „guid“ (Z. 8) und eines für den Titel der Maßnahme (Z. 10). Anhand dessen wird offensichtlich, warum ein ViewModel nötig ist. Die Daten, die in der Oberfläche angezeigt werden, sind Streams, die neue Werte annehmen können. Wann immer sich ein Wert ändert, löst der Stream ein neues Ereignis aus. Auf dieses Ereignis kann der View reagieren. Das Model bietet die Eigenschaften der Maßnahmen dagegen nicht als Stream an.

Weil sich ich Model und ViewModel in ihrer Struktur unterscheiden, gibt es zwei Methoden, die die Konvertierung in beide Richtungen vornehmen. Die Setter-Methode `model` (Z. 12-18) erhält ein Objekt des Wert des Typs `Massnahme` - das Format des Models. Die einzelnen Eigenschaften werden dann in das Format des ViewModels umgewandelt: in Streams. Darüber wird der Setter-Methode `value` von jedem `BehaviorSubject` der entsprechende Wert aus dem Model zugewiesen. Besonders das auch, wie die Auswahloptionen sich im Model und ViewModel unterscheiden. Im ViewModel sind es abgeleitete Objekte der Basisklasse `Choice`, wie z.b. `LetzterStatus`. Im Gegensatz dazu speichert das Modell die Optionen lediglich über die Abkürzung als String ab. Mit Hilfe der Methode `fromAbbreviation` kann Anhand der Abkürzung wieder das entsprechende Objekt wiedergefunden werden (Z. 16).

Die Getter-Methode dagegen konvertiert in das exakte Gegenteil. Die aktuellen Werte von jedem `BehaviorSubject` werden über die Getter-Methode `value` ausgelesen und anschließend der entsprechenden Eigenschaft des Objektes vom Werte-Typ `Massnahme` gespeichert. Die Auswahloption, die für den letzten Status hinterlegt wurde, wird dabei wiederum nur als Abkürzung eingetragen. Dementsprechend ist bloß die Eigenschaft `abbreviation` abzufragen (Z. 22).

Allerdings kann bei Auswahlfeldern auch keine Option gewählt sein. Die Getter-Methode `value` kann daher also auch `null` zurück geben. Der Compiler gibt einen Fehler aus, wenn versucht wird, auf `value` eine Operation auszuführen, sollte es sich um einen Typ mit Null-Zulässigkeit handeln. So ist es bei dem Aufruf von `abbreviation` der Fall (Z. 22). Der Fehler kann nur damit behoben werden, indem das Prefix `?` der Operation vorangestellt wird. In diesem Fall wird die Methode aufgerufen, sollte `value` nicht `null` sein. Ist `value` dagegen `null`, so wird die Operation nicht ausgeführt und der gesamte Ausdruck gibt direkt `null` zurück.

```

5 class MassnahmenFormViewModel {
6   final letzterStatus = BehaviorSubject<LetzterStatus?>.seeded(null);
7
8   final guid = BehaviorSubject<String?>.seeded(null);
9
10  final massnahmenTitel = BehaviorSubject<String>.seeded("");
11
12  set model(Massnahme model) {
13    guid.value = model.guid;
14
15    letzterStatus.value = letzterStatusChoices
16      .fromAbbreviation(model.letzteBearbeitung.letzterStatus);
17    massnahmenTitel.value = model.identifikatoren.massnahmenTitel;
18  }
19
20  Massnahme get model => Massnahme((b) => b
21    ..guid = guid.value
22    ..letzteBearbeitung.letzterStatus = letzterStatus.value?.abbreviation
23    ..letzteBearbeitung.letztesBearbeitungsDatum = DateTime.now().toUtc()
24    ..identifikatoren
25    .update((b) => b..massnahmenTitel = massnahmenTitel.value));
26 }

```

Listing 37: Die Klasse `MassnahmenFormViewModel`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

5.13 Eingabeformular

Das soeben erstellte ViewModel kann nun für die Eingabemaske verwendet werden. Listing 38 zeigt die grundlegende Struktur der Klasse `MassnahmenDetailScreen`.

Wiederum werden das ViewModel und das Model über das „InheritedWidget“ `AppState` abgerufen und in die jeweiligen lokalen Variablen gespeichert (Z. 16,17). Nachfolgend werden zwei Hilfsfunktionen innerhalb der `build`-Methode deklariert. Solche sogenannten nested functions - deutsch verschachtelten Funktionen - sind im Dart erlaubt, was zu einer weiteren Besonderheit führt. Der Sichtbarkeitsbereich von Variablen ist in Dart lexikalisch. Die Bindung der Variablen ist also durch den umgebenden Quelltext bestimmt. Die lokalen Variablen `model` und `vm` sind also im gesamten Bereich sichtbar, der durch die öffnenden und schließenden geschweiften Klammern der Methode `build` aufgespannt wird (Z. 15-103). Damit sind sie auch innerhalb der beiden verschachtelten Funktionen verfügbar. Innerhalb der Funktionen kann auf `model` und `vm` zugegriffen werden, ohne sie über einen Parameter übergeben zu müssen.

Das erste Widget im Inhaltsbereich des Scaffold ist ein `WillPopScope`. Es erlaubt das Verlassen einer Route an eine Abhängigkeit zu knüpfen. Bei dem Eingabeformular handelt es sich um eine Unterseite. Dadurch erscheint in der `AppBar` (Z. 50-52) links von der Überschrift ein Button, der ermöglicht, zur letzten Ansicht zurück zu navigieren (Abb. 8). Dabei stellt sich jedoch die Frage, was mit der bis zu diesem Zeitpunkt eingetragenen Maßnahme passieren soll. Für die Formular-Anwendung soll in diesem Fall die Maßnahme im aktuellen Zustand abgespeichert werden. Dazu wird dem Parameter `onWillPop` als Argument die Funktion `saveRecordAndGoBackToOverviewScreen`.

Anders als im Übersicht-Bildschirm erhält das `Scaffold` kein Argument für den Parameter `floatingActionButton`. Der Hintergrund dafür ist, dass auf diesem Bildschirm in den nächsten Schritten nicht nur ein, sondern zwei solcher Buttons zur Verfügung stehen sollen. Daher muss der Button manual angelegt werden. Das ist nur mit Hilfe eines `Stack`-Widgets möglich, welcher als Kind des `WillPopScope` eingetragen ist. Ein `Stack` erlaubt es mehrere Ebenen in der Tiefe anzulegen. Das unterste Element soll die Auflistung der Eingabefelder sein. Der `SingleChildScrollView` (Z. 57-82) bietet einen vertikalen Scrollbereich an, in dem die Eingabefelder in einer `Column` (Z. 61-79) untereinander aufgelistet sind. Die Ebene, die über den Eingabefeldern eingeblendet wird, soll die beiden Aktions-Buttons zeichnen. Das Widget `Align` erlaubt in dieser Ebene festzulegen, wo die Elemente angeordnet sein sollen (Z. 83-99). Wie für den `FloatingActionButton` üblich wurde die untere rechte Bildschirm-Ecke gewählt (Z. 84). Die Buttons sollen in Zukunft übereinander angeordnet sein, weshalb ein `Column`-Widget zum Einsatz kommt. Zum ersten Mal taucht der Parameter `mainAxisSizeAuf`. Mit dem Argument `MainAxisSize.min` nimmt die `Column` in der Höhe nur so viel Platz ein, wie durch die Kindelemente notwendig. Als bisher einziges Element in der `Column` taucht nun der `FloatingActionButton` auf (Z. 90-95), der die aktuell eingetragenen Daten abspeichern soll. Wenn der Nutzer den Mauszeiger über diesen Button bewegt, wird ein Tooltip angezeigt: "Validiere und speichere Massnahme" (Z. 91). Der Tooltip ist als Konstante angelegt (Z. 7). Das hat vor allem den Grund, dass er auch für den folgenden Integrationstest genutzt wird. Elemente können darin über einen beinhaltenden Text oder Tooltip gefunden werden.

```

7  const saveMassnahmeTooltip = "Validiere und speichere Massnahme";
8
9  class MassnahmenDetailScreen extends StatelessWidget {
10   static const routeName = '/massnahmen-detail';
11
12   const MassnahmenDetailScreen({Key? key}) : super(key: key);
13
14   @override
15   Widget build(BuildContext context) {
16     final vm = AppState.of(context).viewModel;
17     final model = AppState.of(context).model;
18
19     Future<bool> saveRecordAndGoBackToOverviewScreen() {
...    // ...
29   }
30
31   Widget createMassnahmenTitelTextFormField() {
...    // ...
47   }
48
49   return Scaffold(
50     appBar: AppBar(
51       title: const Text('Maßnahmen Detail'),
52     ),
53     body: WillPopScope(
54       onWillPop: () => saveRecordAndGoBackToOverviewScreen(),
55       child: Stack(
56         children: [
57           SingleChildScrollView(
58             child: Center(
59               child: Padding(
60                 padding: const EdgeInsets.all(8.0),
61                 child: Column(
...                   // ...
79               ),
80             ),
81           ),
82         ],
83         Align(
84           alignment: Alignment.bottomRight,
85           child: Padding(
86             padding: const EdgeInsets.all(16.0),
87             child: Column(
88               mainAxisAlignment: MainAxisAlignment.min,
89               children: [
90                 FloatingActionButton(
91                   tooltip: saveMassnahmeTooltip,
92                   heroTag: 'save_floating_action_button',
93                   child: const Icon(Icons.check, color: Colors.white),
94                   onPressed: () => saveRecordAndGoBackToOverviewScreen(),
95                 )
96               ],
97             ),
98           ),
99         ],
100       ),
101     ),
102   ));
103 }
104 }

```

Listing 38: Die Struktur des Bildschirms MassnahmenDetailScreen, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```

61 child: Column(
62   crossAxisAlignment: CrossAxisAlignment.start,
63   children: [
64     SelectionCard<LetzterStatus>(
65       title: letzterStatusChoices.name,
66       allChoices: letzterStatusChoices,
67       initialValue: {
68         if (vm.letzterStatus.value != null)
69           vm.letzterStatus.value!
70       },
71       onSelect: (selectedChoice) =>
72         vm.letzterStatus.value = selectedChoice,
73       onDeselect: (selectedChoice) =>
74         vm.letzterStatus.value = null,
75     ),
76     createMassnahmenTitelTextFormField(),
77     const SizedBox(height: 64)
78   ],
79 ),

```

Listing 39: Die Ausgabe der Formularfelder, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

31 Widget createMassnahmenTitelTextFormField() {
32   final focusNode = FocusNode();
33   return Card(
34     child: Padding(
35       padding: const EdgeInsets.all(16.0),
36       child: TextFormField(
37         focusNode: focusNode,
38         initialValue: vm.massnahmenTitel.value,
39         decoration: const InputDecoration(
40           hintText: 'Maßnahmentitel', labelText: 'Maßnahmentitel'),
41         onChanged: (value) {
42           vm.massnahmenTitel.value = value;
43         },
44       ),
45     ),
46   );
47 }

```

Listing 40: Die Funktion createMassnahmenTitelTextFormField, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

19 Future<bool> saveRecordAndGoBackToOverviewScreen() {
20   ScaffoldMessenger.of(context)
21     ..hideCurrentSnackBar()
22     ..showSnackBar(
23       const SnackBar(content: Text('Massnahme wird gespeichert ...')));
24
25   model.putMassnahmeIfAbsent(vm.model);
26   Navigator.of(context).pop();
27
28   return Future.value(true);
29 }

```

Listing 41: Die Funktion saveRecordAndGoBackToOverviewScreen, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

5.14 Widget SelectionCard

```
7 typedef OnSelect<ChoiceType extends Choice> = void Function(  
8     ChoiceType selectedChoice);  
9  
10 typedef OnDeselect<ChoiceType extends Choice> = void Function(  
11     ChoiceType selectedChoice);  
12  
13 const confirmButtonTooltip = 'Auswahl übernehmen';  
14  
15 class SelectionCard<ChoiceType extends Choice> extends StatelessWidget {  
16     final String title;  
17     final BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel;  
18     final Choices<ChoiceType> allChoices;  
19     final OnSelect<ChoiceType> onSelect;  
20     final OnDeselect<ChoiceType> onDeselect;  
21  
22     SelectionCard(  
23         {required this.title,  
24         required Iterable<ChoiceType> initialValue,  
25         required this.allChoices,  
26         required this.onSelect,  
27         required this.onDeselect,  
28         Key? key})  
29         : selectionViewModel = BehaviorSubject<BuiltSet<ChoiceType>>.seeded(  
30             BuiltSet.from(initialValue)),  
31         super(key: key);
```

Listing 42: Die Klasse SelectionCard, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/widgets/selection_card.dart](#)

```

34 Widget build(BuildContext context) {
35   final focusNode = FocusNode();
36
37   navigateToSelectionScreen() async {
38     focusNode.requestFocus();
39
40     Navigator.push<Set<Choice>>(
41       context,
42       MaterialPageRoute(
43         builder: (context) =>
44           createMultipleChoiceSelectionScreen(context)));
45   }
46
47   return StreamBuilder(
48     stream: selectionViewModel,
49     builder: (context, snapshot) {
50       final selectedChoices = selectionViewModel.value;
51       return Card(
52         child: Column(
53           crossAxisAlignment: CrossAxisAlignment.start,
54           children: [
55             ListTile(
56               focusNode: focusNode,
57               title: Text(title),
58               subtitle:
59                 Text(selectedChoices.map((c) => c.description).join(", ")),
60               trailing: const Icon(Icons.edit),
61               onTap: navigateToSelectionScreen,
62             )
63           ],
64         ),
65       );
66     });
67 }

```

Listing 43: Die Build Methode der SelectionCard, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/widgets/selection_card.dart](#)

```

69 Widget createMultipleChoiceSelectionScreen(BuildContext context) {
70   Future<bool> goBack() {
71     Navigator.of(context).pop();
72     return Future.value(true);
73   }
74
75   return Scaffold(
76     appBar: AppBar(
77       title: Text(title),
78     ),
79     body: Builder(builder: (context) {
80       return StreamBuilder(
81         stream: selectionViewModel,
82         builder: (context, snapshot) {
83           final selectedChoices = selectionViewModel.value;
84           return ListView(children: [
85             ...allChoices.map((ChoiceType c) {
86               bool isSelected = selectedChoices.contains(c);
87
88               onTapTileTab() {
89                 selectionViewModel.value =
90                   selectionViewModel.value.rebuild((b) {
91                     b.replace(isSelected ? [] : [c]);
92                   });
93
94               if (isSelected) {
95                 onDeselect(c);
96               } else {
97                 onSelect(c);
98               }
99             })
100           ]
101           return ListTile(
102             key: Key(
103               "valid choice ${allChoices.name} - ${c.abbreviation}"),
104             title: Column(
105               crossAxisAlignment: CrossAxisAlignment.start,
106               children: [Text(c.description)],
107             ),
108             leading: IconButton(
109               icon: Icon(isSelected
110                 ? Icons.check_box
111                 : Icons.check_box_outline_blank),
112               onPressed: onTapTileTab,
113             ),
114             onTap: onTapTileTab,
115           );
116         }).toList(),
117       ]);
118     });
119   },
120   floatingActionButton: FloatingActionButton(
121     onPressed: goBack,
122     tooltip: confirmButtonTooltip,
123     child: const Icon(Icons.check),
124   ),
125 );
126 }

```

Listing 44: Die Funktion createMultipleChoiceSelectionScreen, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/lib/widgets/selection_card.dart](#)

5.15 Integration Test zum Test der Oberfläche

```
3 Future<void> main() => integrationDriver();
```

Listing 45: Der Integration Test Driver, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/driver.dart](#)

```
18 const durationAfterEachStep = Duration(milliseconds: 1);
19
20 @GenerateMocks([MassnahmenJsonFile])
21 void main() {
22   testWidgets('Can fill the form and save the correct json', (tester) async {
23     final binding = IntegrationTestWidgetsFlutterBinding.ensureInitialized()
24       as IntegrationTestWidgetsFlutterBinding;
25     binding.framePolicy = LiveTestWidgetsFlutterBindingFramePolicy.fullyLive;
26
27     final massnahmenJsonFileMock = MockMassnahmenJsonFile();
28     when(massnahmenJsonFileMock.readMassnahmen()).thenAnswer((_) async => {});
```

Listing 46: Initialisierung des Integrations Tests, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```
30 await tester.pumpWidget(AppState(
31   model: MassnahmenModel(massnahmenJsonFileMock),
32   viewModel: MassnahmenFormViewModel(),
33   child: MaterialApp(
34     title: 'Maßnahmen',
35     theme: ThemeData(
36       primarySwatch: Colors.lightGreen,
37       accentColor: Colors.green,
38       primaryIconTheme: const IconThemeData(color: Colors.white),
39     ),
40     initialRoute: MassnahmenMasterScreen.routeName,
41     routes: {
42       MassnahmenMasterScreen.routeName: (context) =>
43         const MassnahmenMasterScreen(),
44       MassnahmenDetailScreen.routeName: (context) =>
45         const MassnahmenDetailScreen()
46     },
47   ));
```

Listing 47: Initialisierung des Widgets für den Integrations Tests, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```

49 Future<void> tabSelectionCard(Choices choices, {Finder? ancestor}) async {
50   final Finder textLabel;
51   if (ancestor != null) {
52     textLabel =
53       find.descendant(of: ancestor, matching: find.text(choices.name));
54   } else {
55     textLabel = find.text(choices.name);
56   }
57
58   expect(textLabel, findsWidgets);
59
60   final card = find.ancestor(of: textLabel, matching: find.byType(Card));
61   expect(card, findsOneWidget);
62
63   await tester.ensureVisible(card);
64   await tester.tap(card);
65   await tester.pumpAndSettle(durationAfterEachStep);
66 }
67
68 Future<void> tabConfirmButton() async {
69   var confirmChoiceButton = find.byTooltip(confirmButtonTooltip);
70   await tester.tap(confirmChoiceButton);
71   await tester.pumpAndSettle(durationAfterEachStep);
72 }
73
74 Future<Finder> tabOption(Choice choice, {bool tabConfirm = false}) async {
75   final choiceLabel = find.text(choice.description);
76   expect(choiceLabel, findsOneWidget);
77
78   var listTileKey = tester
79     .element(choiceLabel)
80     .findAncestorWidgetOfExactType<ListTile>()!
81     .key!;
82   var listTile = find.byKey(listTileKey);
83
84   expect(listTile, findsOneWidget);
85
86   await tester.ensureVisible(choiceLabel);
87   await tester.tap(choiceLabel);
88   await tester.pumpAndSettle(durationAfterEachStep);
89
90   if (tabConfirm) {
91     await tabConfirmButton();
92   }
93
94   return listTile;
95 }
96
97 Future<void> fillTextFormField(
98   {required String title, required String text}) async {
99   final textFormField = find
100     .ancestor(of: find.text(title), matching: find.byType(TextFormField))
101     .first;
102   expect(textFormField, findsOneWidget);
103
104   await tester.ensureVisible(textFormField);
105   await tester.enterText(textFormField, text);
106   await tester.pumpAndSettle(durationAfterEachStep);
107 }

```

Listing 48: Die Hilfsmethode `tabSelectionCard`, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```

109 await tester.pumpAndSettle(durationAfterEachStep);
110
111 var createNewMassnahmeButton = find.byKey(createNewMassnahmeButtonKey);
112 final gesture = await tester.press(createNewMassnahmeButton);
113 await tester.pumpAndSettle(durationAfterEachStep);
114 await gesture.up();
115 await tester.pumpAndSettle(durationAfterEachStep);

```

Listing 49: Der Button zum Kreieren einer Maßnahme wird ausgelöst, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```

117 await tabSelectionCard(letzterStatusChoices);
118 await tabOption(LetzterStatus.fertig, tabConfirm: true);

```

Listing 50: Der letzte Status wird ausgewählt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```

120 final now = DateTime.now();
121 var massnahmeTitle =
122     "Test Maßnahmen ${now.year}-${now.month}-${now.day} ${now.hour}:${now.minute}";
123 await fillTextFormField(title: "Maßnahmentitel", text: massnahmeTitle);

```

Listing 51: Der Maßnahmentitel wird eingegeben, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```

125 var saveMassnahmeButton = find.byTooltip(saveMassnahmeTooltip);
126 await tester.tap(saveMassnahmeButton);
127 await tester.pumpAndSettle(durationAfterEachStep);

```

Listing 52: Der Button zum Speichern wird ausgelöst, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

```

129 var capturedJson =
130     verify(massnahmenJsonFileMock.saveMassnahmen(captureAny)).captured.last;
131
132 var actualMassnahme = capturedJson['massnahmen'][0] as Map;
133 actualMassnahme.remove("guid");
134 actualMassnahme["letzteBearbeitung"].remove("letztesBearbeitungsDatum");
135
136 var expectedJson = {
137     'letzteBearbeitung': {'letzterStatus': 'fertig'},
138     'identifikatoren': {'massnahmenTitel': massnahmeTitle},
139 };
140
141 expect(actualMassnahme, equals(expectedJson));

```

Listing 53: Der Button zum Speichern wird ausgelöst, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart](#)

5.16 Schritt 2

```
123 await fillTextFormField(title: "Maßnahmentitel", text: massnahmeTitle);
124
125 await tabSelectionCard(foerderklasseChoices);
126 await tabOption(FoerderklasseChoice.aukm_ohne_vns, tabConfirm: true);
127
128 await tabSelectionCard(kategorieChoices);
129 await tabOption(KategorieChoice.extens, tabConfirm: true);
130
131 await tabSelectionCard(zielflaecheChoices);
132 await tabOption(ZielflaecheChoice.al, tabConfirm: true);
133
134 await tabSelectionCard(zieleinheitChoices);
135 await tabOption(ZieleinheitChoice.ha, tabConfirm: true);
136
137 await tabSelectionCard(hauptzielsetzungLandChoices);
138 await tabOption(ZielsetzungLandChoice.biodiv, tabConfirm: true);
139
140 var saveMassnahmeButton = find.byTooltip(saveMassnahmeTooltip);
141 await tester.tap(saveMassnahmeButton);
142 await tester.pumpAndSettle(durationAfterEachStep);
143
144 var capturedJson =
145     verify(massnahmenJsonFileMock.saveMassnahmen(captureAny)).captured.last;
146
147 var actualMassnahme = capturedJson['massnahmen'][0] as Map;
148 actualMassnahme.remove("guid");
149 actualMassnahme["letzteBearbeitung"].remove("letztesBearbeitungsDatum");
150
151 var expectedJson = {
152     'letzteBearbeitung': {'letzterStatus': 'fertig'},
153     'identifikatoren': {'massnahmenTitel': massnahmeTitle},
154     'massnahmenCharakteristika': {
155         'foerderklasse': 'aukm_ohne_vns',
156         'kategorie': 'extens',
157         'zielflaeche': 'al',
158         'zieleinheit': 'ha',
159         'hauptzielsetzungLand': 'biodiv'
160     },
161 };
```

Listing 54: Der Integrationstest klickt 5 weitere Karten, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/integration_test/app_test.dart](#)

6 HIER EINFÜGEN Status Choice

```
11 static final aukum_ohne_vns = FoerderklasseChoice("aukm_ohne_vns",
12     "Agrarumwelt-(und Klima)Maßnahmen, tw. auch mit Tierwohlaspekten, aber OHNE
    ↳ Vertragsnaturschutz");
```

Listing 55: Die Klasse FoerderklasseChoice, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)

Siehe Anhang D Schritt 2 Anhang

```
13 Identifikatoren get identifikatoren;
14
15 Massnahmencharakteristika get massnahmenCharakteristika;
```

Listing 56: massnahmenCharakteristika wird Massnahme hinzugefügt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/data_model/massnahme.dart](#)

```
67 abstract class Massnahmencharakteristika
68     implements
69         Built<Massnahmencharakteristika, MassnahmencharakteristikaBuilder> {
70     String? get foerderklasse;
71     String? get kategorie;
72     String? get zielflaeche;
73     String? get zieleinheit;
74     String? get hauptzielsetzungLand;
```

Listing 57: Der Werte-Typ Massnahmencharakteristika, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/data_model/massnahme.dart](#)

```
83 buildSectionHeadline("Identifikatoren"),
84 createMassnahmenTitelTextFormField(),
85 buildSectionHeadline("Maßnahmencharakteristika"),
86 buildSelectionCard(
87     allChoices: foerderklasseChoices,
88     selectionViewModel: vm.foerderklasse),
89 buildSelectionCard(
90     allChoices: kategorieChoices,
91     selectionViewModel: vm.kategorie),
92 buildSubSectionHeadline("Zielsetzung"),
93 buildSelectionCard(
94     allChoices: zielflaecheChoices,
95     selectionViewModel: vm.zielflaeche),
96 buildSelectionCard(
97     allChoices: zieleinheitChoices,
98     selectionViewModel: vm.zieleinheit,
99 ),
100 buildSelectionCard<ZielsetzungLandChoice>(
101     allChoices: hauptzielsetzungLandChoices,
102     selectionViewModel: vm.hauptzielsetzungLand),
```

Listing 58: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

10 final massnahmenTitel = BehaviorSubject<String>.seeded("");
11
12 final foerderklasse = BehaviorSubject<FoerderklasseChoice?>.seeded(null);
13 final kategorie = BehaviorSubject<KategorieChoice?>.seeded(null);
14 final zielflaeche = BehaviorSubject<ZielflaecheChoice?>.seeded(null);
15 final zieleinheit = BehaviorSubject<ZieleinheitChoice?>.seeded(null);
16 final hauptzielsetzungLand =
17     BehaviorSubject<ZielsetzungLandChoice?>.seeded(null);
18
19 set model(Massnahme model) {
20     guid.value = model.guid;
21
22     letzterStatus.value = letzterStatusChoices
23         .fromAbbreviation(model.letzteBearbeitung.letzterStatus);
24     massnahmenTitel.value = model.identifikatoren.massnahmenTitel;
25
26     {
27         final mc = model.massnahmenCharakteristika;
28
29         foerderklasse.value =
30             foerderklasseChoices.fromAbbreviation(mc.foerderklasse);
31         kategorie.value = kategorieChoices.fromAbbreviation(mc.kategorie);
32
33         zielflaeche.value = zielflaecheChoices.fromAbbreviation(mc.zielflaeche);
34         zieleinheit.value = zieleinheitChoices.fromAbbreviation(mc.zieleinheit);
35         hauptzielsetzungLand.value =
36             hauptzielsetzungLandChoices.fromAbbreviation(mc.hauptzielsetzungLand);
37     }
38 }
39
40 Massnahme get model => Massnahme((b) => b
41     ..guid = guid.value
42     ..letzteBearbeitung.letzterStatus = letzterStatus.value?.abbreviation
43     ..letzteBearbeitung.letztesBearbeitungsDatum = DateTime.now().toUtc()
44     ..identifikatoren.update((b) => b..massnahmenTitel = massnahmenTitel.value)
45     ..massnahmenCharakteristika.update((b) => b
46         ..foerderklasse = foerderklasse.value?.abbreviation
47         ..kategorie = kategorie.value?.abbreviation
48         ..zielflaeche = zielflaeche.value?.abbreviation
49         ..zieleinheit = zieleinheit.value?.abbreviation
50         ..hauptzielsetzungLand = hauptzielsetzungLand.value?.abbreviation));

```

Listing 59: Maßnahmencharakteristika werden dem ViewModel hinzugefügt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

22 _buildColumnHeader(const Text("Maßnahmentitel")),
23 _buildColumnHeader(const Text("Förderklasse")),
24 _buildColumnHeader(const Text("Kategorie")),
25 _buildColumnHeader(const Text("Zielfläche")),
26 _buildColumnHeader(const Text("Zieleinheit")),
27 _buildColumnHeader(const Text("Hauptzielsetzung Land")),

```

Listing 60: Maßnahmencharakteristika werden dem Tabellenkopf hinzugefügt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/widgets/massnahmen_table.dart](#)

```
32 _buildSelectableCell(m, Text(m.identifikatoren.massnahmenTitel)),
33 _buildSelectableCell(
34   m, Text(m.massnahmenCharakteristika.foerderklasse ?? "")),
35 _buildSelectableCell(
36   m, Text(m.massnahmenCharakteristika.kategorie ?? "")),
37 _buildSelectableCell(
38   m, Text(m.massnahmenCharakteristika.zielflaeche ?? "")),
39 _buildSelectableCell(
40   m, Text(m.massnahmenCharakteristika.zieleinheit ?? "")),
41 _buildSelectableCell(m,
42   Text(m.massnahmenCharakteristika.hauptzielsetzungLand ?? "")),
```

Listing 61: Maßnahmencharakteristika werden dem Tabellenkörper hinzugefügt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/widgets/massnahmen_table.dart](#)

6.1 Schritt 3

```
19 final OnSelect<ChoiceType> onSelect;
20 final OnDeselect<ChoiceType> onDeselect;
21 final String? errorText;
22
23 SelectionCard(
24   {required this.title,
25   required Iterable<ChoiceType> initialValue,
26   required this.allChoices,
27   required this.onSelect,
28   required this.onDeselect,
29   this.errorText,
30   Key? key})
```

Listing 62: errorText wird der SelectionCard hinzugefügt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart](#)

```
63   onTap: navigateToSelectionScreen,
64 ),
65 if (errorText != null)
66   Padding(
67     padding: const EdgeInsets.all(8.0),
68     child: Text(errorText!,
69       style:
70         const TextStyle(fontSize: 12.0, color: Colors.red)),
71   )
```

Listing 63: errorText wird ausgegeben, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart](#)

```

215 FloatingActionButton(
216   mini: true,
217   heroTag: 'save_draft_floating_action_button',
218   child: const Icon(Icons.paste, color: Colors.white),
219   backgroundColor: Colors.orange,
220   onPressed: saveDraft,
221 ),
222 const SizedBox(
223   height: 10,
224 ),
225 FloatingActionButton(
226   tooltip: saveMassnahmeTooltip,
227   heroTag: 'save_floating_action_button',
228   child: const Icon(Icons.check, color: Colors.white),
229   onPressed: validateAndSave,
230 )

```

Listing 64: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

25   ..showSnackBar(
26     const SnackBar(content: Text('Entwurf wird gespeichert ...')));
27
28   var draft = vm.model.rebuild((b) =>
29     b.letzteBearbeitung.letzterStatus = LetzterStatus.bearb.abbreviation);
30
31   model.putMassnahmeIfAbsent(draft);
32   Navigator.of(context).pop();
33 }
34
35 void saveRecordAndGoBackToOverviewScreen() {
36   ScaffoldMessenger.of(context)

```

Listing 65: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

20 final formKey = GlobalKey<FormState>();
21
22 void saveDraft() {
23   ScaffoldMessenger.of(context)

```

Listing 66: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

48   children: [
49     Text(
50       'Fehler im Formular trotz Status "${LetzterStatus.fertig.description}"',
51       const SizedBox(width: 4),
52       ElevatedButton(
53         onPressed: saveDraft,
54         child: Padding(
55           padding: const EdgeInsets.fromLTRB(4, 4, 8, 4),
56           child: Row(
57             children: const [
58               Icon(Icons.paste, color: Colors.white),
59               SizedBox(width: 4),
60               Text(
61                 "Entwurf speichern?",
62                 style: TextStyle(fontSize: 18.0, color: Colors.white),
63               ),
64             ],
65           ),
66         ),
67       ),
68     ],
69   ));
70 }
71
72 bool inputsAreValidOrNotMarkedFinal() {
73   if (vm.letzterStatus.value != LetzterStatus.fertig) {

```

Listing 67: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

75   }
76
77   if (formKey.currentState!.validate()) {
78     return true;
79   }
80
81   return false;
82 }
83
84 Future<bool> validateAndSave() {
85   if (inputsAreValidOrNotMarkedFinal()) {

```

Listing 68: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

87     return Future.value(true);
88   } else {
89     showValidationError();
90     return Future.value(false);
91   }
92 }
93
94 Widget createMassnahmenTitelTextFormField() {
95   final focusNode = FocusNode();

```

Listing 69: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

132 return FormField(
133     validator: (_) {
134         if (selectionViewModel.value == null) {
135             return "Feld ${allChoices.name} enthält keinen Wert!";
136         }
137
138         Iterable<Choice> choices = {
139             if (selectionViewModel.value != null) selectionViewModel.value!
140         };
141
142         if (choices.isEmpty) {
143             return "Feld ${allChoices.name} enthält keinen Wert!";
144         }
145
146         return null;
147     },
148     builder: (field) => SelectionCard<ChoiceType>(
149         title: allChoices.name,
150         allChoices: allChoices,
151         initialValue: {
152             if (selectionViewModel.value != null)
153                 selectionViewModel.value!
154         },
155         onSelect: (selectedChoice) =>
156             selectionViewModel.value = selectedChoice,
157         onDeselect: (selectedChoice) => selectionViewModel.value = null,
158         errorText: field.errorText,
159     ));
160 }
161
162 return Scaffold(
163     appBar: AppBar(

```

Listing 70: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

6.2 Schritt 4

```
15 class SelectionCard<ChoiceType extends Choice> extends StatelessWidget {
16   final String title;
17   final BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel;
18   final Choices<ChoiceType> allChoices;
19   final BehaviorSubject<Set<Choice>> priorChoices;
20   final OnSelect<ChoiceType> onSelect;
21   final OnDeselect<ChoiceType> onDeselect;
22   final String? errorText;
23
24   SelectionCard(
25     {required this.title,
26     required Iterable<ChoiceType> initialValue,
27     required this.allChoices,
28     required this.priorChoices,
29     required this.onSelect,
30     required this.onDeselect,
31     this.errorText,
32     Key? key})
```

Listing 71: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/widgets/selection_card.dart](#)

```
51 return StreamBuilder(
52   stream: selectionViewModel,
53   builder: (context, snapshot) {
54     final selectedChoices = selectionViewModel.value;
55     final bool wrongSelection = selectedChoices
56       .any((c) => !c.conditionMatches(priorChoices.value));
57
58     return Card(
59       child: Column(
60         crossAxisAlignment: CrossAxisAlignment.start,
61         children: [
62           ListTile(
63             focusNode: focusNode,
64             title: Text(title),
65             subtitle:
66               Text(selectedChoices.map((c) => c.description).join(", ")),
67             trailing: const Icon(Icons.edit),
68             onTap: navigateToSelectionScreen,
69             tileColor:
70               wrongSelection || errorText != null ? Colors.red : null,
```

Listing 72: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/widgets/selection_card.dart](#)

```

95 body: Builder(builder: (context) {
96   return StreamBuilder(
97     stream: selectionViewModel,
98     builder: (context, snapshot) {
99       final selectedChoices = selectionViewModel.value;
100
101       Set<ChoiceType> selectedAndSelectableChoices = {};
102       Set<ChoiceType> unselectableChoices = {};
103
104       for (ChoiceType c in allChoices) {
105         if (selectedChoices.contains(c) ||
106           c.conditionMatches(priorChoices.value)) {
107           selectedAndSelectableChoices.add(c);
108         } else {
109           unselectableChoices.add(c);
110         }
111       }

```

Listing 73: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/widgets/selection_card.dart](#)

```

113 return ListView(children: [
114   ...selectedAndSelectableChoices.map((ChoiceType c) {
115     bool isSelected = selectedChoices.contains(c);
116     bool selectedButDoesNotMatch =
117       isSelected && !c.conditionMatches(priorChoices.value);

```

Listing 74: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/widgets/selection_card.dart](#)

```

146 tileColor: selectedButDoesNotMatch ? Colors.red : null,

```

Listing 75: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/widgets/selection_card.dart](#)

```

149 ...unselectableChoices
150   .where((c) => !c.conditionMatches(priorChoices.value))
151   .map((Choice c) {
152     return ListTile(
153       key: Key(
154         "invalid choice ${allChoices.name} - ${c.abbreviation}"),
155       title: Text(c.description),
156       leading: const Icon(Icons.close));
157   }).toList()

```

Listing 76: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/widgets/selection_card.dart](#)

```

20 BehaviorSubject<Set<Choice>> priorChoices =
21     BehaviorSubject<Set<Choice>>.seeded({});
22
23 MassnahmenFormViewModel() {
24     Stream<Set<Choice>> choicesStream = Rx.combineLatest([
25         foerderklasse,
26         kategorie,
27         zielflaeche,
28         zieleinheit,
29         hauptzielsetzungLand,
30     ], (_) {
31         return {
32             if (foerderklasse.value != null) foerderklasse.value!,
33             if (kategorie.value != null) kategorie.value!,
34             if (zielflaeche.value != null) zielflaeche.value!,
35             if (zieleinheit.value != null) zieleinheit.value!,
36             if (hauptzielsetzungLand.value != null) hauptzielsetzungLand.value!,
37         };
38     });
39
40     choicesStream.listen((event) => priorChoices.add(event));
41 }

```

Listing 77: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

132 return FormField(
133     validator: (_) {
134         if (selectionViewModel.value == null) {
135             return "Feld ${allChoices.name} enthält keinen Wert!";

```

Listing 78: Die Ausgabe der Formularfelder, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

147     choices.any((c) => !c.conditionMatches(vm.priorChoices.value));
148
149     if (atLeastOneValueInvalid) {
150         return "Wenigstens ein Wert im Feld ${allChoices.name} enthält ist fehlerhaft!";
151     }
152
153     return null;
154 },
155 builder: (field) => SelectionCard<ChoiceType>(
156     title: allChoices.name,

```

Listing 79: Die Ausgabe der Formularfelder, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

160     if (selectionViewModel.value != null)
161         selectionViewModel.value!
162 },
163 onSelect: (selectedChoice) =>

```

Listing 80: Die Ausgabe der Formularfelder, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

3 typedef Condition = bool Function(Set<Choice> choices);
4
5 class Choice {
6   final String description;
7   final String abbreviation;
8   final bool Function(Set<Choice> choices) condition;
9
10  bool conditionMatches(Set<Choice> choices) => condition.call(choices);
11
12  bool conditionDoesNotMatch(Set<Choice> choices) => !condition.call(choices);
13
14  const Choice(this.abbreviation, this.description, {Condition? condition})
15    : condition = condition ?? _conditionIsAlwaysMet;
16
17  static bool _conditionIsAlwaysMet(Set<Choice> choices) => true;
18 }

```

Listing 81: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/choices/base/choice.dart](#)

```

79 static final al = ZielflaecheChoice("al", "AL",
80   condition: (choices) => !choices.contains(KategorieChoice.zf_us));

```

Listing 82: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/choices/choices.dart](#)

```

95 static final wald = ZielflaecheChoice("wald", "Wald/Forst",
96   condition: (choices) =>
97     (choices.contains(FoerderklasseChoice.ea) ||
98       choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
99       choices.contains(FoerderklasseChoice.aukm_ohne_vns)) &&
100     (!choices.contains(KategorieChoice.zf_us) ||
101       !choices.contains(KategorieChoice.bes_kult_rass)));

```

Listing 83: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/choices/choices.dart](#)

```

33 class KategorieChoice extends Choice {
34   static final zf_us = KategorieChoice(
35     "zf_us", "Anbau Zwischenfrucht/Untersaat",
36     condition: (choices) =>
37       choices.contains(FoerderklasseChoice.aukm_ohne_vns));
38   static final anlage_pflege = KategorieChoice(
39     "anlage_pflege", "Anlage/Pflege Struktur",
40     condition: (choices) =>
41       choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
42       choices.contains(FoerderklasseChoice.aukm_ohne_vns));
43   static final dungmang = KategorieChoice("dungmang", "Düngemanagement",
44     condition: (choices) =>
45       choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
46       choices.contains(FoerderklasseChoice.aukm_ohne_vns));
47   static final extens = KategorieChoice("extens", "Extensivierung");
48   static final flst = KategorieChoice("flst", "Flächenstilllegung/Brache",
49     condition: (choices) =>
50       choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
51       choices.contains(FoerderklasseChoice.aukm_ohne_vns));
52   static final umwandlg = KategorieChoice("umwandlg", "Nutzungsumwandlung",
53     condition: (choices) =>
54       choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
55       choices.contains(FoerderklasseChoice.aukm_ohne_vns));
56   static final bes_kult_rass = KategorieChoice(
57     "bes_kult_rass", "Förderung bestimmter Rassen / Sorten / Kulturen",
58     condition: (choices) => !choices.contains(FoerderklasseChoice.ea));
59   static final contact = KategorieChoice("contact", "bitte um Unterstützung");
60
61   KategorieChoice(String abbreviation, String description,
62     {bool Function(Set<Choice> choices)? condition})
63     : super(abbreviation, description, condition: condition);
64 }

```

Listing 84: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/choices/choices.dart](#)

```

77 class ZielflaecheChoice extends Choice {
78   static final ka = ZielflaecheChoice("ka", "keine Angabe/Vorgabe");
79   static final al = ZielflaecheChoice("al", "AL",
80     condition: (choices) => !choices.contains(KategorieChoice.zf_us));
81   static final gl = ZielflaecheChoice("gl", "GL");
82   static final lf = ZielflaecheChoice("lf", "LF");
83   static final dk_sk = ZielflaecheChoice("dk_sk", "DK/SK",
84     condition: (choices) => !choices.contains(FoerderklasseChoice.twm_ziel));
85   static final hff = ZielflaecheChoice("hff", "HFF");
86   static final biotop_le = ZielflaecheChoice(
87     "biotop_le", "Landschaftselement/Biotop o.Ä.",
88     condition: (choices) =>
89       (choices.contains(FoerderklasseChoice.azl) ||
90         choices.contains(FoerderklasseChoice.ea) ||
91         choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
92         choices.contains(FoerderklasseChoice.aukm_ohne_vns)) &&
93       (!choices.contains(KategorieChoice.zf_us) ||
94         !choices.contains(KategorieChoice.bes_kult_rass)));
95   static final wald = ZielflaecheChoice("wald", "Wald/Forst",
96     condition: (choices) =>
97       (choices.contains(FoerderklasseChoice.ea) ||
98         choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
99         choices.contains(FoerderklasseChoice.aukm_ohne_vns)) &&
100       (!choices.contains(KategorieChoice.zf_us) ||
101         !choices.contains(KategorieChoice.bes_kult_rass)));
102   static final contact = ZielflaecheChoice("contact", "bitte um Unterstützung");
103
104   ZielflaecheChoice(String abbreviation, String description,
105     {bool Function(Set<Choice> choices)? condition})
106     : super(abbreviation, description, condition: condition);
107 }

```

Listing 85: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/choices/choices.dart](#)

```

121 class ZieleinheitChoice extends Choice {
122   static final ka = ZieleinheitChoice("ka", "keine Angabe/Vorgabe");
123   static final m3 = ZieleinheitChoice("m3", "m³ (z.B. Gülle)",
124     condition: (choices) =>
125       (choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
126         choices.contains(FoerderklasseChoice.aukm_ohne_vns)) &&
127       (choices.contains(KategorieChoice.dungmang) ||
128         choices.contains(KategorieChoice.extens)) &&
129       (!choices.contains(ZielflaecheChoice.ka) &&
130         !choices.contains(ZielflaecheChoice.contact)));
131   static final pieces = ZieleinheitChoice(
132     "pieces", "Kopf/Stück (z.B. Tiere oder Bäume)",
133     condition: (choices) =>
134       (choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
135         choices.contains(FoerderklasseChoice.aukm_ohne_vns) ||
136         choices.contains(FoerderklasseChoice.twm_ziel)) &&
137       (!choices.contains(KategorieChoice.zf_us) ||
138         !choices.contains(KategorieChoice.flst) ||
139         !choices.contains(KategorieChoice.umwandlg)) &&
140       (!choices.contains(ZielflaecheChoice.ka) &&
141         !choices.contains(ZielflaecheChoice.contact)));
142   static final gve = ZieleinheitChoice("gve", "GV/GVE",
143     condition: (choices) =>
144       (choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
145         choices.contains(FoerderklasseChoice.aukm_ohne_vns) ||
146         choices.contains(FoerderklasseChoice.twm_ziel)) &&
147       (!choices.contains(KategorieChoice.zf_us) ||
148         !choices.contains(KategorieChoice.anlage_pflege) ||
149         !choices.contains(KategorieChoice.flst) ||
150         !choices.contains(KategorieChoice.umwandlg)) &&
151       (!choices.contains(ZielflaecheChoice.ka) &&
152         !choices.contains(ZielflaecheChoice.contact)));
153   static final rgve = ZieleinheitChoice("rgve", "RGV",
154     condition: (choices) =>
155       (choices.contains(FoerderklasseChoice.aukm_nur_vns) ||
156         choices.contains(FoerderklasseChoice.aukm_ohne_vns) ||
157         choices.contains(FoerderklasseChoice.twm_ziel)) &&
158       (!choices.contains(KategorieChoice.zf_us) ||
159         !choices.contains(KategorieChoice.anlage_pflege) ||
160         !choices.contains(KategorieChoice.flst) ||
161         !choices.contains(KategorieChoice.umwandlg)) &&
162       (!choices.contains(ZielflaecheChoice.ka) &&
163         !choices.contains(ZielflaecheChoice.contact)));
164   static final ha = ZieleinheitChoice("ha", "ha",
165     condition: (choices) =>
166       !choices.contains(ZielflaecheChoice.ka) &&
167       !choices.contains(ZielflaecheChoice.contact));
168   static final contact = ZieleinheitChoice("contact", "bitte um Unterstützung");
169
170   ZieleinheitChoice(String abbreviation, String description,
171     {bool Function(Set<Choice> choices)? condition})
172     : super(abbreviation, description, condition: condition);
173 }

```

Listing 86: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-4/conditional_form/lib/choices/choices.dart](#)

6.3 Schritt 5

```
144 var saveMassnahmeButton = find.byTooltip(saveMassnahmeTooltip);
145 await tester.tap(saveMassnahmeButton);
146 await tester.pumpAndSettle(durationAfterEachStep);
147
148 var capturedJson =
149     verify(massnahmenJsonFileMock.saveMassnahmen(captureAny)).captured.last;
150
```

Listing 87: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/integration_test/app_test.dart](#)

```
162 ],
163 'foerderklasse': 'aukm_ohne_vns',
164 'kategorie': 'extens',
165 'zielflaeche': 'al',
166 'zieleinheit': 'ha',
167 'hauptzielsetzungLand': 'biodiv'
```

Listing 88: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/integration_test/app_test.dart](#)

```
219 final hauptzielsetzungLandChoices = Choices<ZielsetzungLandChoice>(
220     _zielsetzungLandChoices,
221     name: "Hauptzielsetzung Land");
222
223 final nebenzielsetzungLandChoices =
224     Choices<ZielsetzungLandChoice>(_zielsetzungLandChoices, name: "Nebenziele");
```

Listing 89: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/choices/choices.dart](#)

```

68 abstract class Massnahmencharakteristika
69     implements
70         Built<Massnahmencharakteristika, MassnahmencharakteristikaBuilder> {
71     String? get foerderklasse;
72     String? get kategorie;
73     String? get zielflaeche;
74     String? get zieleinheit;
75     String? get hauptzielsetzungLand;
76
77     BuiltSet<String> get nebenziele;

```

Listing 90: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/data_model/massnahme.dart](#)

```

133     required BehaviorSubject<dynamic> selectionViewModel,
134     OnSelect<ChoiceType>? onSelect,
135     OnDeselect<ChoiceType>? onDeselect)) {
136     Iterable<ChoiceType> computeChoices(dynamic viewModel) {
137         if (viewModel is BuiltSet) {
138             return viewModel.map((e) => e).cast<ChoiceType>();
139         } else if (viewModel is ChoiceType?) {
140             return {if (viewModel != null) viewModel};
141         } else {
142             throw ArgumentError.value(viewModel);
143         }
144     }
145
146     void _defaultOnSingleSelectStrategy(ChoiceType selectedValue) {
147         var svm = selectionViewModel as BehaviorSubject<ChoiceType?>;
148         svm.value = selectedValue;
149     }
150
151     void _defaultOnSingleDeselectStrategy(ChoiceType selectedValue) {
152         var svm = selectionViewModel as BehaviorSubject<ChoiceType?>;
153         svm.value = null;
154     }
155
156     void _defaultOnMultiSelectStrategy(ChoiceType selectedValue) {
157         var svm = selectionViewModel as BehaviorSubject<BuiltSet<ChoiceType>>;
158         svm.value = svm.value.rebuild((b) => b.add(selectedValue));
159     }
160
161     void _defaultOnMultiDeselectStrategy(ChoiceType deselectedValue) {
162         var svm = selectionViewModel as BehaviorSubject<BuiltSet<ChoiceType>>;
163         svm.value = svm.value.rebuild((b) => b.remove(deselectedValue));
164     }
165
166     OnSelect<ChoiceType> onSelectNonNull;
167     OnDeselect<ChoiceType> onDeselectNonNull;
168     if (multiSelection) {
169         onSelectNonNull = onSelect ?? _defaultOnMultiSelectStrategy;
170         onDeselectNonNull = onDeselect ?? _defaultOnMultiDeselectStrategy;
171     } else {
172         onSelectNonNull = onSelect ?? _defaultOnSingleSelectStrategy;
173         onDeselectNonNull = onDeselect ?? _defaultOnSingleDeselectStrategy;
174     }
175
176     return FormField(
177         validator: (_) {

```

Listing 91: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

187
188 bool atLeastOneValueInvalid =
189     choices.any((c) => !c.conditionMatches(vm.priorChoices.value));
190
191 if (atLeastOneValueInvalid) {

```

Listing 92: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

202         initialValue: computeChoices(selectionViewModel.value),
203         onSelect: onSelectNonNull,
204         onDeselect: onDeselectNonNull,
205         errorText: field.errorText,
206     ));
207 }
208
209 return Scaffold(
210     appBar: AppBar(
211         title: const Text('Maßnahmen Detail'),

```

Listing 93: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

251         allChoices: nebenzielsetzungLandChoices,
252         selectionViewModel: vm.nebenziele,
253         const SizedBox(height: 64)
254     ],
255 ),
256 ),
257 ),

```

Listing 94: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

18 final hauptzielsetzungLand =
19     BehaviorSubject<ZielsetzungLandChoice?>.seeded(null);
20 final nebenziele = BehaviorSubject<BuiltSet<ZielsetzungLandChoice>>.seeded(
21     BuiltSet<ZielsetzungLandChoice>());

```

Listing 95: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

62 hauptzielsetzungLand.value =
63     hauptzielsetzungLandChoices.fromAbbreviation(mc.hauptzielsetzungLand);
64 nebenziele.value = BuiltSet(mc.nebenziele
65     .map((n) => hauptzielsetzungLandChoices.fromAbbreviation(n)));

```

Listing 96: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

79 ..hauptzielsetzungLand = hauptzielsetzungLand.value?.abbreviation
80 ..nebenziele =
81     SetBuilder(nebenziele.value.map((n) => n.abbreviation).toList()));

```

Listing 97: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

27 _buildColumnHeader(const Text("Hauptzielsetzung Land")),
28 _buildColumnHeader(const Text("Nebenziele")),

```

Listing 98: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/widgets/massnahmen_table.dart](#)

```

42 _buildSelectableCell(m,
43   Text(m.massnahmenCharakteristika.hauptzielsetzungLand ?? "")),
44 _buildSelectableCell(
45   m,
46   Column(
47     children: m.massnahmenCharakteristika.nebenziele
48       .map((n) => Text(n))
49       .toList(),
50   )),

```

Listing 99: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/widgets/massnahmen_table.dart](#)

```

15 class SelectionCard<ChoiceType extends Choice> extends StatelessWidget {
16   final String title;
17   final bool multiSelection;
18   final BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel;
19   final Choices<ChoiceType> allChoices;
20   final BehaviorSubject<Set<Choice>> priorChoices;
21   final OnSelect<ChoiceType> onSelect;
22   final OnDeselect<ChoiceType> onDeselect;
23   final String? errorText;
24
25   SelectionCard(
26     {required this.title,
27     required this.multiSelection,

```

Listing 100: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/widgets/selection_card.dart](#)

```

121 onTileTab() {
122   if (multiSelection) {
123     selectionViewModel.value =
124       selectionViewModel.value.rebuild((b) {
125         if (selectionViewModel.value.contains(c)) {
126           return b.remove(c);
127         } else {
128           b.add(c);
129         }
130       });
131   } else {
132     selectionViewModel.value =
133       selectionViewModel.value.rebuild((b) {
134         b.replace(isSelected ? [] : [c]);
135       });
136   }
137   if (isSelected) {
138     onDeselect(c);
139   } else {
140     onSelect(c);
141   }
142 }

```

Listing 101: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-5/conditional_form/lib/widgets/selection_card.dart](#)

6.4 Schritt 6

```
133   required BehaviorSubject<dynamic> selectionViewModel,  
134   onSelect<ChoiceType>? onSelect,  
135   onDeselect<ChoiceType>? onDeselect,  
136   ChoiceMatcher<ChoiceType>? customChoiceMatcherStrategy}) {  
137   final ChoiceMatcher<ChoiceType> matcher =  
138     customChoiceMatcherStrategy ?? _defaultChoiceMatcherStrategy;  
139  
140   Iterable<ChoiceType> computeChoices(dynamic viewModel) {  
141     if (viewModel is BuiltSet) {
```

Listing 102: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```
193     .any((c) => !matcher(c as ChoiceType, vm.priorChoices.value));  
194  
195   if (atLeastOneValueInvalid) {  
196     return "Wenigstens ein Wert im Feld ${allChoices.name} enthält ist fehlerhaft!";  
197   }  
198 }
```

Listing 103: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```
206         initialValue: computeChoices(selectionViewModel.value),  
207         choiceMatcher: matcher,  
208         onSelect: onSelectNonNull,  
209         onDeselect: onDeselectNonNull,  
210         errorText: field.errorText,  
211       ));  
212   }  
213  
214   return Scaffold(  
215     appBar: AppBar(  
216       title: const Text('Maßnahmen Detail'),
```

Listing 104: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

260         ZielsetzungLandChoice.ka &&
261         vm.hauptzielsetzungLand.value !=
262         ZielsetzungLandChoice.contact) {
263     return choice != vm.hauptzielsetzungLand.value;
264 } else {
265     if (choice != ZielsetzungLandChoice.ka &&
266         choice != ZielsetzungLandChoice.contact) {
267         return false;
268     }
269     return true;
270 }
271 },
272 ),
273 const SizedBox(height: 64)
274 ],
275 ),
276 ),
277 ),

```

Listing 105: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

326 }

```

Listing 106: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

13 typedef ChoiceMatcher<ChoiceType extends Choice> = bool Function(
14     ChoiceType choice, Set<Choice> priorChoices);

```

Listing 107: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart](#)

```

18 class SelectionCard<ChoiceType extends Choice> extends StatelessWidget {
19     final String title;
20     final bool multiSelection;
21     final BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel;
22     final Choices<ChoiceType> allChoices;
23     final BehaviorSubject<Set<Choice>> priorChoices;
24     final OnSelect<ChoiceType> onSelect;
25     final OnDeselect<ChoiceType> onDeselect;
26     final String? errorText;
27     final ChoiceMatcher<ChoiceType> choiceMatcher;
28
29     SelectionCard(
30         {required this.title,
31         required this.multiSelection,
32         required Iterable<ChoiceType> initialValue,
33         required this.allChoices,
34         required this.priorChoices,
35         required this.onSelect,
36         required this.onDeselect,
37         required this.choiceMatcher,

```

Listing 108: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart](#)

```

58 return StreamBuilder(
59     stream: selectionViewModel,
60     builder: (context, snapshot) {
61         final selectedChoices = selectionViewModel.value;
62         final bool wrongSelection =
63             selectedChoices.any((c) => !choiceMatcher(c, priorChoices.value));

```

Listing 109: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart](#)

```
111 for (ChoiceType c in allChoices) {
112   if (selectedChoices.contains(c) ||
113       choiceMatcher(c, priorChoices.value)) {
114     selectedAndSelectableChoices.add(c);
115   } else {
116     unselectableChoices.add(c);
117   }
118 }
```

Listing 110: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart](#)

```
120 return ListView(children: [
121   ...selectedAndSelectableChoices.map((ChoiceType c) {
122     bool isSelected = selectedChoices.contains(c);
123     bool selectedButDoesNotMatch =
124       isSelected && !choiceMatcher(c, priorChoices.value);
```

Listing 111: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart](#)

```
166 ...unselectableChoices
167   .where((c) => !choiceMatcher(c, priorChoices.value))
168   .map((Choice c) {
169     return ListTile(
170       key: Key(
171         "invalid choice ${allChoices.name} - ${c.abbreviation}"),
172       title: Text(c.description),
173       leading: const Icon(Icons.close));
```

Listing 112: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart](#)

6.5 Schritt 7

```
148 await tabOption(DuengungArtChoice.dueng_org_n_miner, tabConfirm: true);
149 await tabConfirmButton();
150
151 var saveMassnahmeButton = find.byTooltip(saveMassnahmeTooltip);
152 await tester.tap(saveMassnahmeButton);
153 await tester.pumpAndSettle(durationAfterEachStep);
154
155 var capturedJson =
156     verify(massnahmenJsonFileMock.saveMassnahmen(captureAny)).captured.last;
157
```

Listing 113: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/integration_test/app_test.dart](#)

```
178     {'vorgabe': 'dueng_spez_art', 'art': 'dueng_org_n_miner'}
179   ],
180   };
181
182   expect(actualMassnahme, equals(expectedJson));
183   });
184 }
```

Listing 114: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/integration_test/app_test.dart](#)

```
14 const Choice(this.abbreviation, this.description, {Condition? condition})
15     : condition = condition ?? _conditionIsAlwaysMet;
16
17 @override
18 bool operator ==(Object other) =>
19     identical(this, other) ||
20     other is Choice &&
21         description == other.description &&
22         abbreviation == other.abbreviation;
23
24 @override
25 int get hashCode => description.hashCode ^ abbreviation.hashCode;
```

Listing 115: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/choices/base/choice.dart](#)

```

226 class DuengungChoice extends Choice {
227     static final o = DuengungChoice("o", "keine Angabe / ohne Einschränkung",
228         condition: (choices) => !choices.contains(contact));
229     static final dueng_keine = DuengungChoice("keine", "keine Düngung",
230         condition: (choices) =>
231             !choices.contains(o) && !choices.contains(contact));
232     static final ausbring_techn = DuengungChoice("ausbring_techn",
233         "besondere Ausbringungstechnik (z.B. Schleppschlauch) mit weiteren Angaben in
234         ↳ Technische Anforderungen",
235         condition: (choices) =>
236             !choices.contains(o) && !choices.contains(contact));
237     static final ausbring_t_kurz = DuengungChoice(
238         "ausbring_t_kurz", "verkürzte oder vorgegebene Ausbringungszeiten",
239         condition: (choices) =>
240             !choices.contains(o) && !choices.contains(contact));
241     static final dueng_red = DuengungChoice(
242         "dueng_red", "reduzierte Düngungsmenge",
243         condition: (choices) =>
244             !choices.contains(o) && !choices.contains(contact));
245     static final dueng_spez_art = DuengungChoice(
246         "dueng_spez_art", "Beschränkung auf spezifische Düngemittel",
247         condition: (choices) =>
248             !choices.contains(o) && !choices.contains(contact));
249     static final contact = DuengungChoice("contact", "bitte um Unterstützung",
250         condition: (choices) => !choices.contains(o));
251
252     bool get canSetArt => canSetArtCondition({this});
253     final Condition canSetArtCondition =
254         (choices) => choices.contains(dueng_spez_art);
255
256     DuengungChoice(String abbreviation, String description,
257         {bool Function(Set<Choice> choices)? condition})
258         : super(abbreviation, description, condition: condition);
259 }

```

Listing 116: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/choices/choices.dart](#)

```

260 final duengungChoices = Choices<DuengungChoice>({
261     DuengungChoice.o,
262     DuengungChoice.dueng_keine,
263     DuengungChoice.ausbring_techn,
264     DuengungChoice.ausbring_t_kurz,
265     DuengungChoice.dueng_red,
266     DuengungChoice.dueng_spez_art,
267     DuengungChoice.contact,
268 }, name: "Düngung");

```

Listing 117: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/choices/choices.dart](#)

```

270 class DuengungArtChoice extends Choice {
271   static final o = DuengungArtChoice("o", "keine Angabe");
272   static final dueng_org_tier_liqu = DuengungArtChoice(
273     "dueng_org_tier_liqu", "organisch, tierisch und flüssig: z.B. Gülle");
274   static final dueng_org_tier_fest = DuengungArtChoice(
275     "dueng_org_tier_fest", "organisch, tierisch, fest: Festmist");
276   static final dueng_org_pfl =
277     DuengungArtChoice("dueng_org_pfl", "organisch, pflanzlich: Gärrest");
278   static final dueng_org = DuengungArtChoice(
279     "dueng_org", "organisch: nicht differenziert oder mehreres betreffend");
280   static final dueng_miner = DuengungArtChoice("dueng_miner", "mineralisch");
281   static final dueng_org_n_miner =
282     DuengungArtChoice("dueng_org_n_miner", "organisch und mineralisch");
283   static final contact = DuengungArtChoice("contact", "bitte um Unterstützung");
284
285   DuengungArtChoice(String abbreviation, String description,
286     {bool Function(Set<Choice> choices)? condition})
287     : super(abbreviation, description, condition: condition);
288 }

```

Listing 118: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/choices/choices.dart](#)

```

290 final duengungArtChoices = Choices<DuengungArtChoice>({
291   DuengungArtChoice.o,
292   DuengungArtChoice.dueng_org_tier_liqu,
293   DuengungArtChoice.dueng_org_tier_fest,
294   DuengungArtChoice.dueng_org_pfl,
295   DuengungArtChoice.dueng_org,
296   DuengungArtChoice.dueng_miner,
297   DuengungArtChoice.dueng_org_n_miner,
298   DuengungArtChoice.contact
299 }, name: "Düngung Art");

```

Listing 119: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/choices/choices.dart](#)

```

16 Massnahmencharakteristika get massnahmenCharakteristika;
17
18 BuiltSet<Duengung> get duengung;

```

Listing 120: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/data_model/massnahme.dart](#)

```

91 abstract class Duengung implements Built<Duengung, DuengungBuilder> {
92   String get vorgabe;
93
94   String? get art;
95
96   Duengung._();
97
98   factory Duengung([void Function(DuengungBuilder) updates]) = _$Duengung;
99
100   static Serializer<Duengung> get serializer => _$duengungSerializer;
101 }

```

Listing 121: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/data_model/massnahme.dart](#)

```

133 {bool multiSelection = false,
134 required Choices<ChoiceType> allChoices,
135 required BehaviorSubject<dynamic> selectionViewModel,
136 ListTileTitleBuilder<ChoiceType>? listTileTitleBuilder,
137 OnSelect<ChoiceType>? onSelect,
138 OnDeselect<ChoiceType>? onDeselect,
139 ChoiceMatcher<ChoiceType>? customChoiceMatcherStrategy}} {
140 final ChoiceMatcher<ChoiceType> matcher =

```

Listing 122: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

207         allChoices: allChoices,
208         priorChoices: vm.priorChoices,
209         initialValue: computeChoices(selectionViewModel.value),
210         choiceMatcher: matcher,
211         listTileTitleBuilder: listTileTitleBuilder,
212         onSelect: onSelectNonNull,
213         onDeselect: onDeselectNonNull,
214         errorText: field.errorText,
215     ));
216 }
217
218 return Scaffold(

```

Listing 123: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

280         allChoices: Choices<DuengungViewModel>({
281             ...vm.duengung.value,
282             for (final c in duengungChoices)
283                 if (!vm.duengung.value.contains(c))
284                     DuengungViewModel(c)
285         }, name: duengungChoices.name),
286         selectionViewModel: vm.duengung,
287         listTileTitleBuilder:
288             (context, choice, isSelected) {
289             return Column(
290                 crossAxisAlignment: CrossAxisAlignment.start,
291                 children: [
292                     Text(choice.description),
293                     if (isSelected) ...[
294                         if (choice.canSetArt)
295                             buildSelectionCard(
296                                 allChoices: duengungArtChoices,
297                                 selectionViewModel: choice.art),
298                     ],
299                 ],
300             );
301         },
302         const SizedBox(height: 64)
303     ],
304 ),
305 ),

```

Listing 124: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart](#)

```

20 final nebenziele = BehaviorSubject<BuiltSet<ZielsetzungLandChoice>>.seeded(
21     BuiltSet<ZielsetzungLandChoice>());
22 final duengung = BehaviorSubject<BuiltSet<DuengungViewModel>>.seeded(
23     BuiltSet<DuengungViewModel>());

```

Listing 125: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

28 MassnahmenFormViewModel() {
29   Stream<Set<Choice>> choicesStream = Rx.combineLatest([
30     foerderklasse,
31     kategorie,
32     zielflaeche,
33     zieleinheit,
34     hauptzielsetzungLand,
35     duengung,
36   ], (_) {
37     return {
38       if (foerderklasse.value != null) foerderklasse.value!,
39       if (kategorie.value != null) kategorie.value!,
40       if (zielflaeche.value != null) zielflaeche.value!,
41       if (zieleinheit.value != null) zieleinheit.value!,
42       if (hauptzielsetzungLand.value != null) hauptzielsetzungLand.value!,
43       ...duengung.value,
44     };
45   });

```

Listing 126: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

68 nebenziele.value = BuiltSet(mc.nebenziele
69   .map((n) => hauptzielsetzungLandChoices.fromAbbreviation(n)));
70 }
71
72 duengung.value = BuiltSet(
73   {for (var d in model.duengung) DuengungViewModel.fromModel(d)});

```

Listing 127: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

87 ..nebenziele =
88   SetBuilder(nebenziele.value.map((n) => n.abbreviation).toList())
89   ..duengung.replace(duengung.value.map((e) => e.model));
90 }

```

Listing 128: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

92 class DuengungViewModel extends DuengungChoice {
93   final art = BehaviorSubject<DuengungArtChoice?>.seeded(null);
94
95   DuengungViewModel(DuengungChoice choice)
96     : super(choice.abbreviation, choice.description,
97       condition: choice.condition);
98
99   factory DuengungViewModel.fromModel(Duengung model) {
100     final vm =
101       DuengungViewModel(duengungChoices.fromAbbreviation(model.vorgabe!));
102     vm.model = model;
103     return vm;
104   }
105
106   set model(Duengung model) {
107     if (model.art != null) {
108       art.value = duengungArtChoices.fromAbbreviation(model.art);
109     }
110   }
111
112   Duengung get model => Duengung((b) {
113     b.vorgabe = abbreviation;
114
115     if (art.value != null) {
116       b.art = art.value!.abbreviation;
117     }
118   });
119 }

```

Listing 129: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart](#)

```

28 _buildColumnHeader(const Text("Nebenziele")),
29 _buildColumnHeader(const Text("Düngung"))

```

Listing 130: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/massnahmen_table.dart](#)

```

45 _buildSelectableCell(
46   m,
47   Column(
48     children: m.massnahmenCharakteristika.nebenziele
49       .map((n) => Text(n))
50       .toList(),
51   )),
52 _buildSelectableCell(m, buildDuengungTable(m.duengung), padding: 0)

```

Listing 131: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/massnahmen_table.dart](#)

```

59 Widget buildDuengungTable(Iterable<Duengung> duengung) {
60   return Table(
61     border: TableBorder.all(width: 0.5),
62     defaultColumnWidth: const IntrinsicColumnWidth(),
63     children: [
64       TableRow(children: [
65         _buildColumnHeader(const Text("Vorgabe")),
66         _buildColumnHeader(const Text("Art")),
67       ]),
68       ...duengung
69         .map((d) => TableRow(
70           children: [
71             _buildColumnHeader(Text(d.vorgabe)),
72             if (d.art != null)
73               _buildColumnHeader(Text(d.art ?? ""))
74             else
75               _buildColumnHeader(const Text("")),
76           ],
77         ))
78         .toList()
79     ],
80   );
81 }

```

Listing 132: XXXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/massnahmen_table.dart](#)

```

7 typedef ListTileTitleBuilder<ChoiceType extends Choice> = Widget Function(
8   BuildContext context, ChoiceType choice, bool isSelected);
9
10 Widget _defaultBuildListTileTitleStrategy(context, Choice c, isSelected) {
11   return Column(
12     crossAxisAlignment: CrossAxisAlignment.start,
13     children: [Text(c.description)],
14   );
15 }

```

Listing 133: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart](#)

```

28 class SelectionCard<ChoiceType extends Choice> extends StatelessWidget {
29   final String title;
30   final bool multiSelection;
31   final BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel;
32   final Choices<ChoiceType> allChoices;
33   final BehaviorSubject<Set<Choice>> priorChoices;
34   final OnSelect<ChoiceType> onSelect;
35   final OnDeselect<ChoiceType> onDeselect;
36   final String? errorText;
37   final ListTileTitleBuilder<ChoiceType> listTileTitleBuilder;
38   final ChoiceMatcher<ChoiceType> choiceMatcher;
39
40   SelectionCard(
41     {required this.title,
42     required this.multiSelection,
43     required Iterable<ChoiceType> initialValue,
44     required this.allChoices,
45     required this.priorChoices,
46     ListTileTitleBuilder<ChoiceType>? listTileTitleBuilder,
47     required this.onSelect,
48     required this.onDeselect,
49     required this.choiceMatcher,
50     this.errorText,
51     Key? key})
52     : selectionViewModel = BehaviorSubject<BuiltSet<ChoiceType>>.seeded(
53       BuiltSet.from(initialValue)),
54       listTileTitleBuilder =
55         listTileTitleBuilder ?? _defaultBuildListTileTitleStrategy,
56       super(key: key);
57

```

Listing 134: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart](#)

```

106 Widget createMultipleChoiceSelectionScreen(BuildContext context) {
107   final formKey = GlobalKey<FormState>();
108
109   Future<bool> goBackOnValidInput() {
110     if (formKey.currentState!.validate()) {
111       Navigator.of(context).pop();
112       return Future.value(true);
113     } else {
114       ScaffoldMessenger.of(context)
115         .showSnackBar(const SnackBar(content: Text('Fehler im Formular.')));
116       return Future.value(false);
117     }
118   }

```

Listing 135: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart](#)

```
120 return Scaffold(  
121   appBar: AppBar(  
122     title: Text(title),  
123   ),  
124   body: Builder(builder: (context) {  
125     return WillPopScope(  
126       onWillPop: goBackOnValidInput,  
127       child: Form(  
128         key: formKey,  
129         child: StreamBuilder(  
130           stream: selectionViewModel,  
131           builder: (context, snapshot) {  
132             final selectedChoices = selectionViewModel.value;  
133  
134             Set<ChoiceType> selectedAndSelectableChoices = {};  
135             Set<ChoiceType> unselectableChoices = {};
```

Listing 136: XXXX, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart](#)

```

1 class MassnahmenFormViewModel {
2     final letzterStatus = BehaviorSubject<LetzterStatus?>.seeded(null);
3     // ...
4 }

```

Listing 137: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

```

1 class ChoiceChangeNotifier extends ChangeNotifier {
2     BuiltSet<Choice> _choices = BuiltSet<Choice>();
3
4     BuiltSet<Choice> get choices => _choices;
5
6     set choices(BuiltSet<Choice> choices) {
7         _choices = choices;
8         notifyListeners();
9     }
10 }
11 class LetzterStatusViewModel extends ChoiceChangeNotifier {}

```

Listing 138: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

```

1 providers: [
2     Provider<MassnahmenFormViewModel>(
3         create: (_) => MassnahmenFormViewModel(),
4     Provider<MassnahmenJsonFile>(create: (_) => MassnahmenJsonFile()),
5     Provider(
6         create: (context) => MassnahmenPool(
7             Provider.of<MassnahmenJsonFile>(context, listen: false)),
8         ChangeNotifierProvider(create: (context) => LetzterStatusViewModel())
9 ],

```

Listing 139: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

```
1 Consumer<ConsumerType>(  
2     builder: (context, choiceChangeNotifier, child) {  
3         final selectedChoices = choiceChangeNotifier.choices;  
4         return Card(  
5             child: Column(  
6                 crossAxisAlignment: CrossAxisAlignment.start,  
7                 children: [  
8                     ListTile(  
9                         focusNode: focusNode,  
10                        title: Text(title),  
11                        subtitle:  
12                        Text(selectedChoices.map((c) => c.description).join(", ")),  
13                        trailing: const Icon(Icons.edit),  
14                        onTap: navigateToSelectionScreen,  
15                    )  
16                ],  
17            ),  
18        );  
19    },  
20 )
```

Listing 140: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

Teil III

Anhang

A Technologiewahl Anhang

A.1 Stimmen verwendeter Frameworks

Auf diesen Anhang verweisen!

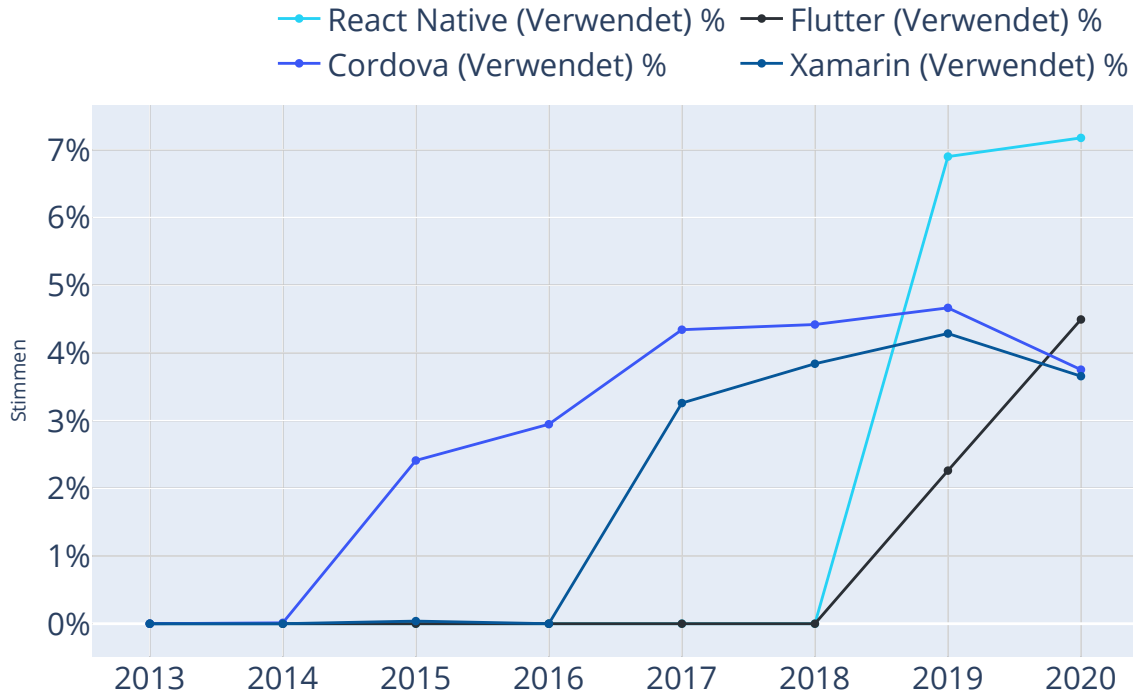


Abbildung 10: Stimmen verwendeter Frameworks, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

A.2 Stimmen gewünschter Frameworks

Auf diesen Anhang verweisen!

B Vergleich React Native und Flutter Anhang

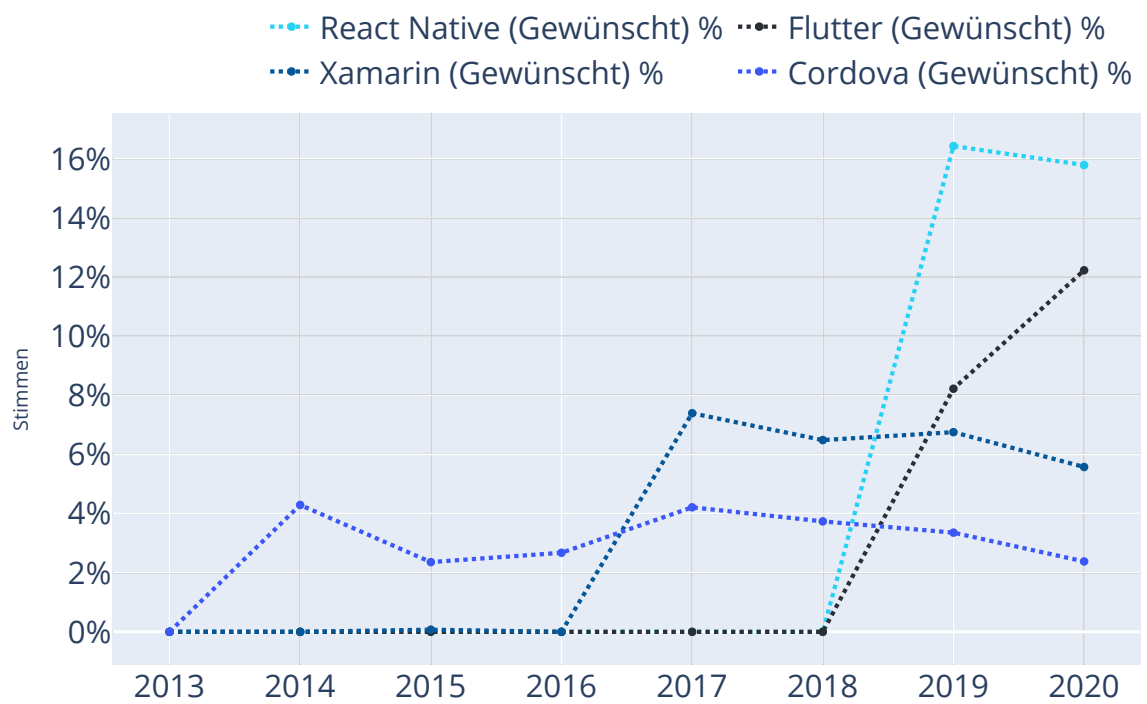


Abbildung 11: Stimmen gewünschter Frameworks, Quelle: Eigene Abbildung, Notebook: [Charts/StackOverflowUmfrage/StackOverflowUmfrage.ipynb](#), Daten-Quelle: **FEHLT!**

```

1 import 'package:flutter/material.dart';
2 import 'validation.dart';
3
4 void main() => runApp(MyApp());
5
6 class MyApp extends StatelessWidget {
7   @override
8   Widget build(BuildContext context) => MaterialApp(
9     home: Scaffold(
10      body: MyCustomForm(),
11    ),
12  );
13 }
14
15 class MyCustomForm extends StatefulWidget {
16   @override
17   MyCustomFormState createState() => MyCustomFormState();
18 }
19
20 class MyCustomFormState extends State<MyCustomForm> {
21   final _formKey = GlobalKey<FormState>();
22
23   @override
24   Widget build(BuildContext context) => Form(
25     key: _formKey,
26     child: Padding(
27       padding: const EdgeInsets.all(8.0),
28       child: Column(
29         children: [
30           TextFormField(
31             decoration: const InputDecoration(labelText: "Name"),
32             validator: (String? value) =>
33               validateNotEmpty("Name", value)),
34           TextFormField(
35             decoration: const InputDecoration(labelText: "Email"),
36             validator: (String? value) => validateEmail("Name", value)),
37           TextFormField(
38             decoration: const InputDecoration(labelText: "Password"),
39             validator: (String? value) =>
40               validateNotEmpty("Password", value)),
41           Padding(
42             padding: const EdgeInsets.symmetric(vertical: 16.0),
43             child: ElevatedButton(
44               onPressed: () {
45                 if (_formKey.currentState!.validate()) {
46                   ScaffoldMessenger.of(context).showSnackBar(
47                     SnackBar(content: Text('Processing Data')));
48                 }
49               },
50               child: Text('Submit'),
51             ),
52           ),
53         ],
54       ),
55     ),
56   );
57 }

```

Listing 141: , Quelle: Eigenes Listing, Datei: [Quellcode/Vergleich/form-in-flutter/lib/main.dart](#)

```

1 final emailPattern = RegExp(
2   r'^(((~<>()\[\]\.\.,;\s@")+(\.~<>()\[\]\.\.,;\s@")+)|(".*"))@((\[[0-9]{1,3}\. _]
   ↳  [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\)|((\[a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$');
3
4 String? validateEmail(String label, String? value) {
5   if (value == null || value.isEmpty) {
6     return '$label is required';
7   } else if (!emailPattern.hasMatch(value)) {
8     return 'Invalid Email Format';
9   } else {
10    return null;
11  }
12 }
13
14 String? validateNotEmpty(String label, String? value) {
15   if (value == null || value.isEmpty) {
16     return '$label is required';
17   } else {
18     return null;
19   }
20 }

```

Listing 142: , Quelle: Eigenes Listing, Datei: [Quellcode/Vergleich/form-in-flutter/lib/validation.dart](#)

```

1 import * as React from 'react';
2 import {
3   Text,
4   View,
5   StyleSheet,
6   Button,
7   Alert,
8   ScrollView,
9 } from 'react-native';
10 import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-view';
11
12 import Constants from 'expo-constants';
13 import { useForm } from 'react-hook-form';
14
15 // You can import from local files
16 import Input from '../components/Input';
17 import Form from '../components/Form';
18 import validation from '../validation';
19 import Hero from '../Hero';
20
21 type FormData = {
22   name: string;
23   email: string;
24   password: string;
25 };
26
27 export default () => {
28   const { handleSubmit, register, setValue, errors } = useForm<FormData>();
29
30   const onSubmit = (data: FormData) => {
31     Alert.alert('data', JSON.stringify(data));
32   };
33
34   return (
35     <KeyboardAwareScrollView
36       contentContainerStyle={styles.container}
37       style={{ backgroundColor: '#181e34' }}>
38       <Hero />
39       <View style={styles.formContainer}>
40         <Form {...{ register, setValue, validation, errors }}>
41           <Input name="name" label="Name" />
42           <Input name="email" label="Email" />
43           <Input name="password" label="Password" secureTextEntry={true} />
44           <Button title="Submit" onPress={handleSubmit(onSubmit)} />
45         </Form>
46       </View>
47     </KeyboardAwareScrollView>
48   );
49 };
50
51 const styles = StyleSheet.create({
52   container: {
53     flex: 1,
54     justifyContent: 'center',
55     paddingTop: Constants.statusBarHeight,
56     backgroundColor: '#181e34',
57   },
58   formContainer: {
59     padding: 8,
60     flex: 1,
61   },
62   button: {
63     backgroundColor: 'red',
64   },
65 });

```

Listing 143: , Quelle: <https://dev.to/elaziziyousseuf/forms-in-react-native-the-right-way-4d46>,
Datei: [Quellcode/Vergleich/form-in-react-native-the-right-way/App.tsx](#)

```

1 import * as React from 'react';
2 import { Text, View, StyleSheet, Image } from 'react-native';
3
4 export default () => {
5   return (
6     <View style={styles.container}>
7       <Image style={styles.logo} source={require('./assets/hero.jpg')} />
8       <Text style={styles.paragraph}>
9         Form in React Native, The right Way!
10      </Text>
11    </View>
12  );
13 }
14
15 const styles = StyleSheet.create({
16   container: {
17     justifyContent: 'center',
18     flex:1,
19   },
20   paragraph: {
21     margin: 24,
22     marginTop: 0,
23     fontSize: 34,
24     fontWeight: 'bold',
25     textAlign: 'center',
26     color: '#FFF'
27   },
28   logo: {
29     width: '100%',
30     height: 200
31   }
32 });

```

Listing 144: , Quelle: <https://dev.to/elaziziyoussouf/forms-in-react-native-the-right-way-4d46>,
 Datei: [Quellcode/Vergleich/form-in-react-native-the-right-way/Hero.tsx](#)

```

1 export default {
2   name: {required: {value: true, message: 'Name is required'}},
3   email: {
4     required: {value: true, message: 'Email is required'},
5     pattern: {
6       value: /^(?!(\s|[\s\.\;\s@"]+)|(\s|[\s\.\;\s@"]+)*|(".*"))@_|
7         ↪ ((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\)|((\[[a-zA-Z\ -0-9]+\.\ _]
8         ↪ )+[a-zA-Z]{2,}))$/,
9       message: 'Invalid Email Format',
10     },
11   },
12   password: {
13     required: {value: true, message: 'Password is required'},
14   },
15 };

```

Listing 145: , Quelle: <https://dev.to/elaziziyoussouf/forms-in-react-native-the-right-way-4d46>,
 Datei: [Quellcode/Vergleich/form-in-react-native-the-right-way/validation.tsx](#)

```

1 import * as React from 'react';
2 import { TextInput, KeyboardAvoidingView, findNodeHandle } from 'react-native';
3 import { ValidationOptions, FieldError } from 'react-hook-form';
4
5 interface ValidationMap {
6   [key: string]: ValidationOptions;
7 }
8
9 interface ErrorMap {
10  [key: string]: FieldError | undefined;
11 }
12
13 interface Props {
14   children: JSX.Element | JSX.Element[];
15   register: (
16     field: { name: string },
17     validation?: ValidationOptions
18   ) => void;
19   errors: ErrorMap;
20   validation: ValidationMap;
21   setValue: (name: string, value: string, validate?: boolean) => void;
22 }
23
24 export default ({
25   register,
26   errors,
27   setValue,
28   validation,
29   children,
30 }: Props) => {
31   const Inputs = React.useRef<Array<TextInput>>([]);
32
33   React.useEffect(() => {
34     (Array.isArray(children) ? [...children] : [children]).forEach((child) => {
35       if (child.props.name)
36         register({ name: child.props.name }, validation[child.props.name]);
37     });
38   }, [register]);
39
40   return (
41     <>
42       {(Array.isArray(children) ? [...children] : [children]).map(
43         (child, i) => {
44           return child.props.name
45             ? React.createElement(child.type, {
46               ...{
47                 ...child.props,
48                 ref: (e: TextInput) => {
49                   Inputs.current[i] = e;
50                 },
51                 onChangeText: (v: string) =>
52                   setValue(child.props.name, v, true),
53                 onSubmitEditing: () => {
54                   Inputs.current[i + 1]
55                     ? Inputs.current[i + 1].focus()
56                     : Inputs.current[i].blur();
57               },
58               //onBlur: () => triggerValidation(child.props.name),
59               blurOnSubmit: false,
60               //name: child.props.name,
61               error: errors[child.props.name],
62             },
63             )
64           : child;
65         }
66       )}
67     </>
68   );
69 };

```

```

1 import * as React from 'react';
2 import {
3   View,
4   TextInput,
5   Text,
6   StyleSheet,
7   ViewStyle,
8   TextStyle,
9   TextInputProps,
10 } from 'react-native';
11 import { FieldError } from 'react-hook-form';
12 interface Props extends TextInputProps {
13   name: string;
14   label?: string;
15   labelStyle?: TextStyle;
16   error?: FieldError | undefined;
17 }
18
19 export default React.forwardRef<any, Props>((
20   (props, ref): React.ReactElement => {
21     const { label, labelStyle, error, ...inputProps } = props;
22
23     return (
24       <View style={styles.container}>
25         {label && <Text style={[styles.label, labelStyle]}>{label}</Text>}
26         <TextInput
27           autoCapitalize="none"
28           ref={ref}
29           style={[styles.input, { borderColor: error ? '#fc6d47' : '#c0cbd3' }]}
30           {...inputProps}
31         />
32         <Text style={styles.textError}>{error && error.message}</Text>
33       </View>
34     );
35   }
36 );
37
38 const styles = StyleSheet.create({
39   container: {
40     marginVertical: 8,
41   },
42   input: {
43     borderStyle: 'solid',
44     borderWidth: 1,
45     borderRadius: 5,
46     paddingVertical: 5,
47     paddingLeft: 5,
48     fontSize: 16,
49     height: 40,
50     color: '#c0cbd3',
51   },
52   label: {
53     paddingVertical: 5,
54     fontSize: 16,
55     fontWeight: 'bold',
56     color: '#c0cbd3',
57   },
58   textError: {
59     color: '#fc6d47',
60     fontSize: 14,
61   },
62 });

```

Listing 147: , Quelle: <https://dev.to/elaziziyoussouf/forms-in-react-native-the-right-way-4d46>,
 Datei: [Quellcode/Vergleich/form-in-react-native-the-right-way/components/Input.tsx](#)

C Schritt 1 Anhang

```
48 test('Storage with one Massnahme deserialises without error', () {
49   var json = {
50     "massnahmen": [
51       {
52         "guid": "test massnahme id",
53         "letzteBearbeitung": {
54           "letztesBearbeitungsDatum": 0,
55           "letzterStatus": "bearb"
56         },
57         "identifikatoren": {"massnahmenTitel": "Massnahme 1"}
58       }
59     ]
60   };
61
62   var expectedStorage = Storage();
63   expectedStorage =
64     expectedStorage.rebuild((b) => b.massnahmen.add(Massnahme((b) => b
65       ..guid = "test massnahme id"
66       ..identifikatoren.massnahmenTitel = "Massnahme 1"
67       ..letzteBearbeitung.update((b) {
68         b.letztesBearbeitungsDatum =
69           DateTime.fromMillisecondsSinceEpoch(0, isUtc: true);
70       })))));
71
72   var actualStorage = serializers.deserializeWith(Storage.serializer, json);
73
74   expect(actualStorage, equals(expectedStorage));
```

Listing 148: Ein automatisierter Testfall überprüft, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-1/conditional_form/test/data_model/storage_test.dart](#)

D Schritt 2 Anhang

```

5 class FoerderklasseChoice extends Choice {
6   static final oelb = FoerderklasseChoice("oelb", "Ökolandbau");
7   static final azl = FoerderklasseChoice("azl", "Ausgleichszulage");
8   static final ea = FoerderklasseChoice("ea", "Erschwerenausgleich");
9   static final aukm_nur_vns = FoerderklasseChoice("aukm_nur_vns",
10     "Agrarumwelt-(und Klima)Maßnahme: nur Vertragsnaturschutz");
11   static final aukm_ohne_vns = FoerderklasseChoice("aukm_ohne_vns",
12     "Agrarumwelt-(und Klima)Maßnahmen, tw. auch mit Tierwohlaspekten, aber OHNE
13     ↳ Vertragsnaturschutz");
14   static final twm_ziel = FoerderklasseChoice(
15     "twm_ziel", "Tierschutz/Tierwohlmaßnahmen mit diesem als Hauptziel");
16   static final contact =
17     FoerderklasseChoice("contact", "bitte um Unterstützung");
18   FoerderklasseChoice(String abbreviation, String description,
19     {bool Function(Set<Choice> choices)? condition})
20     : super(abbreviation, description);
21 }
22
23 final foerderklasseChoices = Choices<FoerderklasseChoice>({
24   FoerderklasseChoice.oelb,
25   FoerderklasseChoice.azl,
26   FoerderklasseChoice.ea,
27   FoerderklasseChoice.aukm_nur_vns,
28   FoerderklasseChoice.aukm_ohne_vns,
29   FoerderklasseChoice.twm_ziel,
30   FoerderklasseChoice.contact
31 }, name: "Förderklasse");
32
33 class KategorieChoice extends Choice {
34   static final zf_us =
35     KategorieChoice("zf_us", "Anbau Zwischenfrucht/Untersaat");
36   static final anlage_pflege =
37     KategorieChoice("anlage_pflege", "Anlage/Pflege Struktur");
38   static final dungmang = KategorieChoice("dungmang", "Düngemanagement");
39   static final extens = KategorieChoice("extens", "Extensivierung");
40   static final flst = KategorieChoice("flst", "Flächenstilllegung/Brache");
41   static final umwandlg = KategorieChoice("umwandlg", "Nutzungsumwandlung");
42   static final bes_kult_rass = KategorieChoice(
43     "bes_kult_rass", "Förderung bestimmter Rassen / Sorten / Kulturen");
44   static final contact = KategorieChoice("contact", "bitte um Unterstützung");
45
46   KategorieChoice(String abbreviation, String description)
47     : super(abbreviation, description);
48 }
49
50 final kategorieChoices = Choices<KategorieChoice>({
51   KategorieChoice.zf_us,
52   KategorieChoice.anlage_pflege,
53   KategorieChoice.dungmang,
54   KategorieChoice.extens,
55   KategorieChoice.flst,
56   KategorieChoice.umwandlg,
57   KategorieChoice.bes_kult_rass,
58   KategorieChoice.contact
59 }, name: "Kategorie");

```

Listing 149: Die Klasse FoerderklasseChoice, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)

```

61 class ZielflaecheChoice extends Choice {
62   static final ka = ZielflaecheChoice("ka", "keine Angabe/Vorgabe");
63   static final al = ZielflaecheChoice("al", "AL");
64   static final gl = ZielflaecheChoice("gl", "GL");
65   static final lf = ZielflaecheChoice("lf", "LF");
66   static final dk_sk = ZielflaecheChoice("dk_sk", "DK/SK");
67   static final hff = ZielflaecheChoice("hff", "HFF");
68   static final biotop_le =
69     ZielflaecheChoice("biotop_le", "Landschaftselement/Biotop o.Ä.");
70   static final wald = ZielflaecheChoice("wald", "Wald/Forst");
71   static final contact = ZielflaecheChoice("contact", "bitte um Unterstützung");
72
73   ZielflaecheChoice(String abbreviation, String description)
74     : super(abbreviation, description);
75 }
76
77 final zielflaecheChoices = Choices<ZielflaecheChoice>({
78   ZielflaecheChoice.ka,
79   ZielflaecheChoice.al,
80   ZielflaecheChoice.gl,
81   ZielflaecheChoice.lf,
82   ZielflaecheChoice.dk_sk,
83   ZielflaecheChoice.hff,
84   ZielflaecheChoice.biotop_le,
85   ZielflaecheChoice.wald,
86   ZielflaecheChoice.contact
87 }, name: "Zielfläche");
88
89 class ZieleinheitChoice extends Choice {
90   static final ka = ZieleinheitChoice("ka", "keine Angabe/Vorgabe");
91   static final m3 = ZieleinheitChoice("m3", "m³ (z.B. Gülle)");
92   static final pieces =
93     ZieleinheitChoice("pieces", "Kopf/Stück (z.B. Tiere oder Bäume)");
94   static final gve = ZieleinheitChoice("gve", "GV/GVE");
95   static final rgve = ZieleinheitChoice("rgve", "RGV");
96   static final ha = ZieleinheitChoice("ha", "ha");
97   static final contact = ZieleinheitChoice("contact", "bitte um Unterstützung");
98
99   ZieleinheitChoice(String abbreviation, String description)
100     : super(abbreviation, description);
101 }
102
103 final zieleinheitChoices = Choices<ZieleinheitChoice>({
104   ZieleinheitChoice.ka,
105   ZieleinheitChoice.m3,
106   ZieleinheitChoice.pieces,
107   ZieleinheitChoice.gve,
108   ZieleinheitChoice.rgve,
109   ZieleinheitChoice.ha,
110   ZieleinheitChoice.contact
111 }, name: "Zieleinheit");

```

Listing 150: Die Menge foerderklasseChoices, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)

```

113 class ZielsetzungLandChoice extends Choice {
114     static final ka = ZielsetzungLandChoice("ka", "keine Angabe/Vorgabe");
115     static final bsch = ZielsetzungLandChoice("bsch", "Bodenschutz");
116     static final wsch = ZielsetzungLandChoice("wsch", "Gewässerschutz");
117     static final asch = ZielsetzungLandChoice("asch", "Spezieller Artenschutz");
118     static final biodiv = ZielsetzungLandChoice("biodiv", "Biodiversität");
119     static final struktkviel =
120         ZielsetzungLandChoice("struktkviel", "Erhöhung der Strukturvielfalt");
121     static final genet_res = ZielsetzungLandChoice("genet_res",
122         "Erhaltung genetischer Ressourcen (Pflanzen, z. B. im Grünland, und Tiere, z. B.
123         ↳ bedrohte Rassen)");
124     static final tsch = ZielsetzungLandChoice(
125         "tsch", "Tierschutz/Maßnahmen zum Tierwohl im Betrieb");
126     static final klima = ZielsetzungLandChoice("klima", "Klima");
127     static final contact =
128         ZielsetzungLandChoice("contact", "bitte um Unterstützung");
129     ZielsetzungLandChoice(String abbreviation, String description)
130         : super(abbreviation, description);
131 }
132
133 final _zielsetzungLandChoices = {
134     ZielsetzungLandChoice.ka,
135     ZielsetzungLandChoice.bsch,
136     ZielsetzungLandChoice.wsch,
137     ZielsetzungLandChoice.asch,
138     ZielsetzungLandChoice.biodiv,
139     ZielsetzungLandChoice.struktkviel,
140     ZielsetzungLandChoice.genet_res,
141     ZielsetzungLandChoice.tsch,
142     ZielsetzungLandChoice.klima,
143     ZielsetzungLandChoice.contact
144 };
145
146 final hauptzielsetzungLandChoices = Choices<ZielsetzungLandChoice>(
147     _zielsetzungLandChoices,
148     name: "Hauptzielsetzung Land");

```

Listing 151: Die Klasse KategorieChoice, Quelle: Eigenes Listing, Datei: [Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart](#)

Literatur

- <Formik /> | *Formik Docs API*. Juni 2021. URL: https://web.archive.org/web/20210409184616if_/https://formik.org/docs/api/formik (Zitiert auf der Seite 27).
- Adobe Inc. *FAQ | PhoneGap Docs*. Aug. 2016. URL: <https://web.archive.org/web/20200806024626/http://docs.phonegap.com/phonegap-build/faq/> (Zitiert auf der Seite 23).
- *Update for Customers Using PhoneGap and PhoneGap Build*. Aug. 2020. URL: <https://web.archive.org/web/20200811121213/https://blog.phonegap.com/update-for-customers-using-phonegap-and-phonegap-build-cc701c77502c?gi=df435eca31bb> (Zitiert auf der Seite 24).
- Borenkraout, Matan. *Native Testing Library Introduction | Testing Library Docs*. Nov. 2020. URL: <https://web.archive.org/web/20210128142719/https://testing-library.com/docs/react-native-testing-library/intro/> (Zitiert auf der Seite 28).
- Does redux-form work with React Native?* Juni 2021. URL: <https://web.archive.org/web/20210602234346/https://redux-form.com/7.3.0/docs/faq/reactnative.md/> (Zitiert auf der Seite 26).
- Facebook Inc. *The React Native Ecosystem*. Juni 2021. URL: <https://web.archive.org/web/20210602191504/https://github.com/facebook/react-native/blob/d48f7ba748a905818e8c64fe70fe5b24aa098b05/ECOSYSTEM.md> (Zitiert auf der Seite 28).
- Fowler, Martin. *Inversion of Control Containers and the Dependency Injection pattern*. Jan. 2004. URL: <http://web.archive.org/web/20210707041912/https://martinfowler.com/articles/injection.html> (Zitiert auf der Seite 43).
- *InversionOfControl*. Juni 2005. URL: <http://web.archive.org/web/20050628234825/https://martinfowler.com/bliki/InversionOfControl.html> (Zitiert auf der Seite 43).
- Gamma, Erich u. a. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Pearson Deutschland GmbH, 2009 (Zitiert auf der Seite 38).
- Google LLC. *Build a form with validation*. Juni 2021. URL: <https://web.archive.org/web/20210122020924/https://flutter.dev/docs/cookbook/forms/validation> (Zitiert auf der Seite 27).
- *Dart - Language tour - spread operator*. Juli 2021. URL: <https://web.archive.org/web/20210625070139/https://dart.dev/guides/language/language-tour#spread-operator> (besucht am 08.07.2021) (Zitiert auf der Seite 52).
 - *Dart Programming Language Specification 5th edition*. Apr. 2021. URL: <https://web.archive.org/web/20210702071617/https://dart.dev/guides/language/specifications/DartLangSpec-v2.10.pdf> (Zitiert auf den Seiten 31, 32, 36).
 - *Desktop support for Flutter*. Juni 2021. URL: <https://web.archive.org/web/20210531034514/http://flutter.dev/desktop/> (Zitiert auf der Seite 13).
 - *Flutter - Beautiful native apps in record time*. Juni 2021. URL: <https://web.archive.org/web/20210630233338/https://flutter.dev/> (Zitiert auf der Seite 14).
 - *Flutter - Introduction to widgets*. Juni 2021. URL: <http://web.archive.org/web/20210603081649/https://flutter.dev/docs/development/ui/widgets-intro> (Zitiert auf der Seite 14).
 - *Forms | Flutter Docs Cookbook*. Juni 2021. URL: <https://web.archive.org/web/20201102003629/https://flutter.dev/docs/cookbook/forms> (Zitiert auf der Seite 27).
 - *Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe*. Mai 2021. URL: <https://support.google.com/trends/answer/4365533> (Zitiert auf der Seite 22).
 - *Web support for Flutter*. Mai 2021. URL: <http://web.archive.org/web/20210506012158/https://flutter.dev/web> (Zitiert auf der Seite 14).

- Gosling, James u. a. *The Java® Language Specification Java SE 16 Edition*. Feb. 2021. URL: <https://web.archive.org/web/20210514051033/https://docs.oracle.com/javase/specs/jls/se16/jls16.pdf> (Zitiert auf den Seiten 31, 32).
- Gossman, John. *Introduction to Model/View/ViewModel pattern for building WPF apps*. Okt. 2005. URL: <https://web.archive.org/web/20101103111603/http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx> (Zitiert auf der Seite 42).
- Johnson, Ralph E und Brian Foote. „Designing reusable classes“. In: *Journal of object-oriented programming* 1.2 (1988), S. 22–35 (Zitiert auf der Seite 43).
- Lynch, Max. *The Last Word on Cordova and PhoneGap*. März 2014. URL: <https://web.archive.org/web/20210413012559/https://blog.ionicframework.com/what-is-cordova-phonegap/> (Zitiert auf der Seite 23).
- React Native* | *Formik Docs*. Juni 2021. URL: https://web.archive.org/web/20210507005917if_/https://formik.org/docs/guides/react-native (Zitiert auf der Seite 26).
- React Native* | *React Hook Form - Get Started*. Juni 2021. URL: https://web.archive.org/web/20210523042601if_/https://react-hook-form.com/get-started/ (Zitiert auf der Seite 26).
- reduxForm* | *Redux Form - API*. Juni 2021. URL: <https://web.archive.org/web/20210506221401/https://redux-form.com/7.4.2/docs/api/reduxform.md/#code-validate-values-object-props-object-gt-errors-object-code-optional> (Zitiert auf der Seite 27).
- register* | *React Hook Form - API*. Juni 2021. URL: <https://web.archive.org/web/20210406032209/https://react-hook-form.com/api/useform/register> (Zitiert auf der Seite 27).
- Spolsky, Joel. „How Hard Could It Be?: The Unproven Path“. In: *inc.com* (Nov. 2008). URL: <http://web.archive.org/web/20081108094045/http://www.inc.com/magazine/20081101/how-hard-could-it-be-the-unproven-path.html> (Zitiert auf der Seite 22).
- Stack Exchange, Inc. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*. Mai 2021. URL: <https://insights.stackoverflow.com/survey/> (Zitiert auf den Seiten 22, 23).

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig angefertigt und mich fremder Hilfe nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichtem oder nicht veröffentlichtem Schrifttum entnommen sind, habe ich als solche kenntlich gemacht.

Wernigerode, den 16.11.2020

Alexander Johr