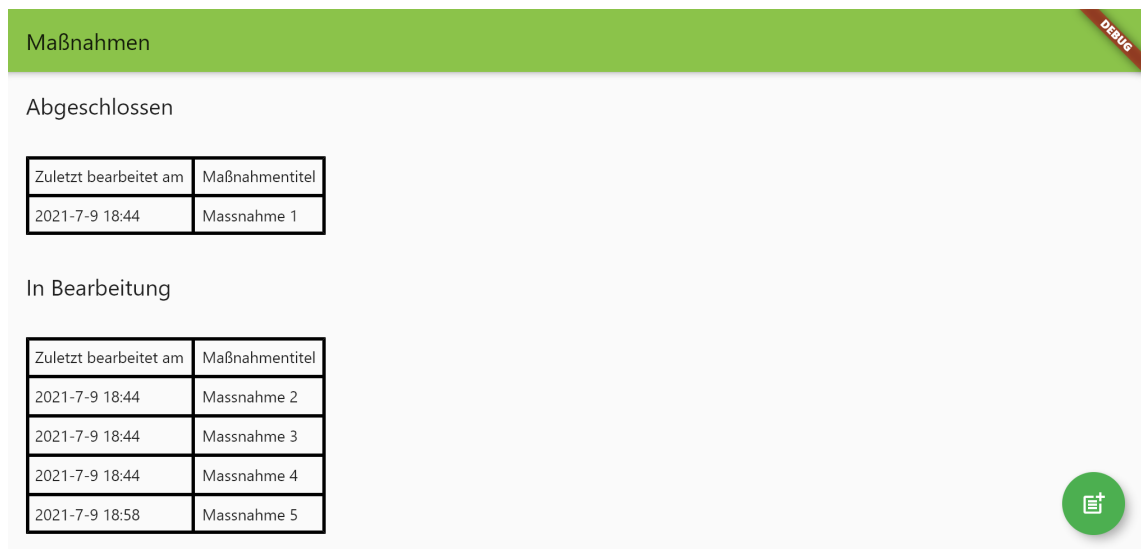


Im ersten Schritt soll die Formular-Anwendung in ihrer Grundstruktur entwickelt werden. Die erste Ansicht, welche der Benutzer sieht, soll die Übersicht der bereits eingetragenen Maßnahmen sein. Abbildung 1 zeigt diese Übersicht.



Maßnahmen

Abgeschlossen

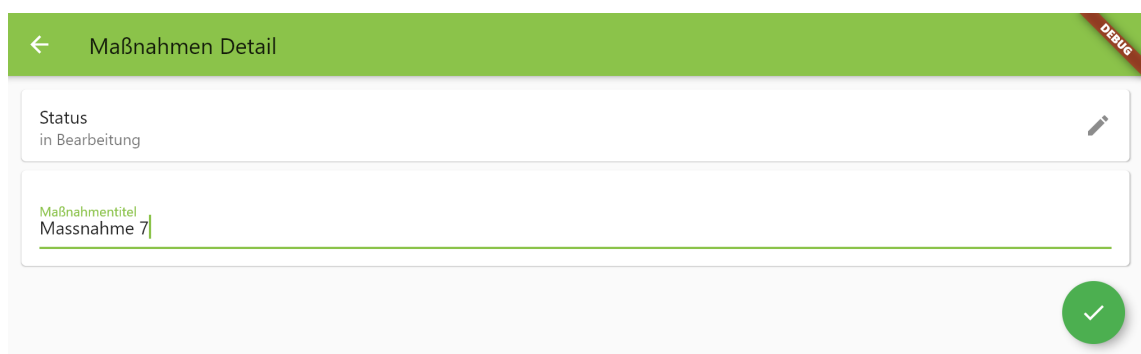
Zuletzt bearbeitet am	Maßnahmentitel
2021-7-9 18:44	Massnahme 1

In Bearbeitung

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 1: 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. Sie ist in Abbildung 2 zu sehen.



← **Maßnahmen Detail**

Status
in Bearbeitung

Maßnahmentitel
Massnahme 7

Abbildung 2: 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“, wie in Abbildung 3 zu sehen. 1



Abbildung 3: Der Selektions-Bildschirm für das Feld Status erlaubt die Auswahl der Optionen „in Bearbeitung“ und „abgeschlossen“. Quelle: Eigene Abbildung

1 HIER EINFÜGEN Status Choice

```

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 }

```

Code Snippet 1: Die Klasse LetzterStatus, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/lib/choices/choices.dart

```

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

```

Code Snippet 2: Die Klasse Choice, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/lib/choices/base/choice.dart

```

6 part '$file_name$.g.dart';
7
8 abstract class $ClassName$ implements Built<$ClassName$, $ClassName$Builder> {
9   $todo$
10
11   $ClassName$. _();
12   factory $ClassName$([void Function($ClassName$Builder) updates]) =
13     ↳ _$$$ClassName$;
14 }

```

Code Snippet 3: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

2 Ein Test soll verifizieren, dass die Daten korrekt abgelegt werden

Doch 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 XXX - 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. Dieser Boilerplate-Code kann durch das Live Template für Android Studio in Listing 3 generiert werden.¹

`$ClassName$` 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 Templates der Boilerplate Code generiert. In Listing 4 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“ - Kurzform von General Unique Identifier - eindeutig identifiziert werden können. 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 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. Die Attribute „letzteBearbeitung“ und „identifikatoren“ erhalten dagegen ganz automatisch Standardwerte in Form von Instanzen der dazugehörigen Klassen. Diese wiederum konfigurieren ihre eigenen Felder und deren Initialwerte.

Der Werte-Typ Identifikatoren ist in Listing 5 zu sehen. Er enthält das Attribut „massnahmenTitel“, welcher im Eingabeformular durch das Texteingabefeld gefüllt werden wird.

¹<https://web.archive.org/web/20210710140113/https://github.com/GiancarloCode/built-value-snippets/blob/master/intellij/src/main/resources/liveTemplates/snippets.xml>

```

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 }

```

Code Snippet 4: Der Werte-Typ Massnahme, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-1/conditional_form/lib/data_model/massnahme.dart

```

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

```

Code Snippet 5: 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 10 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 CC-CCCCC.

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“ (Zeilen 45-48). Es ist eine Hilfsmethode, die das letzte Bearbeitungsdatum in ein für Menschen lesbares Datumsformat umwandelt. In dem Übersichts-Bildschirm Abbildung 1 ist das Datumsformat sichtbar.

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

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.

Das Ergebnis der Serialisierung wird im dazugehörigen Unit-Test ersichtlich. Listing ZZZZZZZ zeigt den Unit Test für den Typ Maßnahme. In Zeile 8 wird ein Objekt der

```

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;

```

Code Snippet 6: Der Werte-Typ LetzteBearbeitung, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/lib/data_model/massnahme.dart

```

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

```

Code Snippet 7: Der Serialisierer für Massnahme und Storage, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/lib/data_model/serializers.dart

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 in Bild Vaio mithilfe des sogenannten Bilder Entwurfsmuster möglich. 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.

—

```

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

```

Code Snippet 8: Ein automatisierter Testfall überprüft, ob die Maßnahme korrekt serialisiert wird, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-1/conditional_form/test/data_model/massnahme_test.dart

```

37 test('Massnahme deserialises without error', () {
38   var json = {
39     'guid': "test massnahme id",
40     'letzteBearbeitung': {
41       'letztesBearbeitungsDatum': 0,
42       'letzterStatus': 'bearb'
43     },
44     'identifikatoren': {'massnahmenTitel': 'Massnahme 1'}
45   };
46
47   var expectedMassnahme = Massnahme((b) => b
48     ..guid = "test massnahme id"
49     ..identifikatoren.massnahmenTitel = "Massnahme 1"
50     ..letzteBearbeitung.update((b) {
51       b.letztesBearbeitungsDatum =
52         DateTime.fromMillisecondsSinceEpoch(0, isUtc: true);
53     }));
54   var actualMassnahme =
55     serializers.deserializeWith(Massnahme.serializer, json);
56
57   expect(actualMassnahme, equals(expectedMassnahme));
58 });

```

Code Snippet 9: Ein automatisierter Testfall überprüft, ob die Maßnahme korrekt deserialisiert wird, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-1/conditional_form/test/data_model/massnahme_test.dart

```

9 abstract class Storage implements Built<Storage, StorageBuilder> {
10   @BuiltValueField(wireName: 'massnahmen')
11   BuiltSet<Massnahme> get massnahmen;
12
13   static void _initializeBuilder(StorageBuilder b) =>
14     b..massnahmen = SetBuilder();
15
16   Storage._();
17
18   factory Storage([void Function(StorageBuilder) updates]) = _$Storage;
19
20   static Serializer<Storage> get serializer => _$storageSerializer;
21 }

```

Code Snippet 10: Der Werte-Typ Storage, Quelle: Eigenes Listing,
 Datei: Quellcode/Schritt-1/conditional_form/lib/data_model/storage.dart

```

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

```

Code Snippet 11: Ein automatisierter Testfall überprüft, ob die Liste der Maßnahmen korrekt serialisiert wird, Quelle: Eigenes Listing,
 Datei: Quellcode/Schritt-1/conditional_form/test/data_model/storage_test.dart

```

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));
75 });

```

Code Snippet 12: Ein automatisierter Testfall überprüft, ob die Liste der Maßnahmen korrekt deserialisiert wird, Quelle: Eigenes Listing,
 Datei: Quellcode/Schritt-1/conditional_form/test/data_model/storage_test.dart

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

Code Snippet 13: Die Klasse MassnahmenJsonFile, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-1/conditional_form/lib/persistence/massnahmen_json_file.dart

```

7
8 class MassnahmenPool {
9   final MassnahmenJsonFile jsonFile;
10
11   MassnahmenPool(this.jsonFile) {
12     init();
13   }
14
15   final storageSubject = BehaviorSubject<Storage>.seeded(
16     Storage((b) => b..massnahmen = SetBuilder()));
17
18   Storage get storage => storageSubject.value;
19
20   set storage(Storage m) {
21     storageSubject.value = m;
22   }
23
24   init() async {
25     refresh();
26   }
27
28   refresh() async {
29     try {
30       final massnahmenAsJson = await jsonFile.readMassnahmen();
31
32       final massnahmen =
33         serializers.deserializeWith(Storage.serializer, massnahmenAsJson)!;
34
35       storage = massnahmen;
36     } on MassnahmenFileDoesNotExistException {
37       storage = Storage();
38     }
39   }
40
41   putMassnahmeIfAbsent(Massnahme massnahme) async {
42     var rebuild = storage.rebuild((b) => b.massnahmen
43       ..removeWhere((m) => m.guid == massnahme.guid)
44       ..add(massnahme));
45
46     var serializedMassnahmen =
47       serializers.serializeWith(Storage.serializer, rebuild);
48
49     await jsonFile.saveMassnahmen(serializedMassnahmen as Map<String, dynamic>);
50
51     storage = rebuild;
52   }
53 }

```

Code Snippet 14: Die Klasse MassnahmenPool, Quelle: Eigenes Listing,
 Datei: Quellcode/Schritt-1/conditional_form/lib/data_access/massnahmen_pool.dart

```

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

```

Code Snippet 15: Die Struktur der Klasse MassnahmenMasterScreen, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_master.dart

```

31 return SingleChildScrollView(
32   child: Column(
33     crossAxisAlignment: CrossAxisAlignment.start,
34     children: [
35       const Padding(
36         padding: EdgeInsets.all(16.0),
37         child: Text(
38           "Abgeschlossen",
39           style: TextStyle(fontSize: 20),
40         ),
41     ),
42     SingleChildScrollView(
43       scrollDirection: Axis.horizontal,
44       child: Padding(
45         padding: const EdgeInsets.all(16.0),
46         child: MassnahmenTable(
47           massnahmenPool.storage.massnahmen
48             .where((m) =>
49               m.letzteBearbeitung.letzterStatus ==
50               LetzterStatus.fertig.abbreviation)
51             .toSet(), onSelect: (selectedMassnahme) {
52               final vm = Provider.of<MassnahmenFormViewModel>(
53                 context,
54                 listen: false);
55               vm.model = selectedMassnahme.rebuild((m) => m
56                 ..letzteBearbeitung.letztesBearbeitungsDatum =
57                 DateTime.now().toUtc());
58               Navigator.of(context)
59                 .pushNamed(MassnahmenDetailScreen.routeName);
60             })),
61   ),

```

Code Snippet 16: Die Ausgabe der finalen Maßnahmen, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_master.dart

```

48 .where((m) =>
49   m.letzteBearbeitung.letzterStatus == LetzterStatus.bearb.abbreviation)

```

Code Snippet 17: Die Bedingung der Entwurf-Maßnahmen, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-1/conditional_form/lib/screens/massnahmen_master.dart

```

5 class MassnahmenTable extends StatefulWidget {
6   final Set<Massnahme> _massnahmenToDisplay;
7   final OnSelectCallback? onSelect;
8
9   const MassnahmenTable(this._massnahmenToDisplay, {this.onSelect, Key? key})
10     : super(key: key);
11
12   @override
13   _MassnahmenTableState createState() => _MassnahmenTableState();
14 }
15
16 class _MassnahmenTableState extends State<MassnahmenTable> {
17   @override
18   Widget build(BuildContext context) {
19     return Table(
20       border: TableBorder.all(width: 3),
21       defaultColumnWidth: const IntrinsicColumnWidth(),
22       defaultVerticalAlignment: TableCellVerticalAlignment.middle,
23       children: [
24         TableRow(children: [
25           _buildColumnHeader(const Text("Zuletzt bearbeitet am")),
26           _buildColumnHeader(const Text("Maßnahmentitel"))
27         ]),
28         ...widget._massnahmenToDisplay.map((m) {
29           return TableRow(children: [
30             _buildSelectableCell(m, Text(m.letzteBearbeitung.formattedDate)),
31             _buildSelectableCell(m, Text(m.identifikatoren.massnahmenTitel)),
32           ]);
33         }).toList(),
34       ],
35     );
36   }
37
38   Widget _buildColumnHeader(Widget child) => Padding(
39     padding: const EdgeInsets.all(8.0),
40     child: child,
41   );
42
43   Widget _buildSelectableCell(Massnahme m, Widget child,
44     {double padding = 8.0}) =>
45     TableRowInkWell(
46       onTap: () {
47         if (widget.onSelect != null) {
48           widget.onSelect!(m);
49         }
50       },
51       child: Padding(
52         padding: EdgeInsets.all(padding),
53         child: child,
54       ),
55     );
56 }
57
58 typedef OnSelectCallback = void Function(Massnahme selectedMassnahme);

```

Code Snippet 18: Die Klasse MassnahmenTable, Quelle: Eigenes Listing,
 Datei: Quellcode/Schritt-1/conditional_form/lib/widgets/massnahmen_table.dart

Wunsch

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

3 Model View ViewModel

3.1 Model

3.2 View

3.3 ViewModel

4 Selection Card

4.1 Strategie Entwurfsmuster

4.1.1 On Select On Deselect Strategie

4.1.2 Compute Choice Strategie

4.2 Single Selection

4.2.1 View Model

Choice

4.2.2 View

4.2.3 Model

4.3 Multi Selection Selection

4.3.1 View Model

BuiltSet

4.3.2 View

4.3.3 Model

4.4 Nested Selection

4.4.1 View Model

BuiltMap

Nested View Model

Convertierung View Model und Model

Nested View Strategie

4.4.2 Extensions Methods VS Compute Choice from Selection Model

4.4.3 View

4.4.4 Model

5 Doktorarbeit

5.1 Condtions

5.1.1 And

Check

Finde invalide oder fehlende Auswahl

5.1.2 Or

Check

Finde invalide oder fehlende Auswahl

5.1.3 In

Check

Finde invalide oder fehlende Auswahl

5.1.4 NotIn

5.1.5 Exclusive

5.1.6 Always Met

5.2 Fluent API

5.3 Chart

5.3.1 Generiere Chart automatisch und Rekursiv aus Datenstruktur

5.3.2 Zeichne invalide oder fehlende Auswahl

5.3.3 conditions chart dart

Gespräch mit Daniel über Doktorarbeit math bibliotheken

ki

viel detailliertere beschreibungen ausführungen nötig mehr recherche nötig

fachwissen anwenden fachwissen abstrahieren transfer

weg der transferleistung beschreiben können

wenn übertragung der abstrakten in ein neues feld passiert die methoden neu das was dabei passiert neu ist die methoden neu ist

// TODO: Research Question: Which VM Package for nested views, Consumers of same type (Cobid anschauen?)

Anhang