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

Teil I

Implementierung

0.1 Schritt 6

In diesem Schritt soll das Formular um Mehrfachauswahlfelder erweitert werden. Im Speziellen handelt es sich um das Auswahlfeld Nebenziele. Es beinhaltet die gleichen Auswahloptionen wie das Auswahlfeld Hauptzielsetzung.

0.1.1 Integrationstest erweitern

Zunächst wird der Integrationstest um die Auswahl der Nebenziele erweitert (Listing 1).

```
118 await tabSelectionCard(hauptzielsetzungLandChoices);
119 await tabOption(ZielsetzungLandChoice.biodiv, tabConfirm: true);
120
121 await tabSelectionCard(nebenzielsetzungLandChoices);
122 await tabOption(ZielsetzungLandChoice.bsch);
123 await tabOption(ZielsetzungLandChoice.klima, tabConfirm: true);
124
125 var saveMassnahmeButton = find.byTooltip(saveMassnahmeTooltip);
126 await tester.tap(saveMassnahmeButton);
127 await tester.pumpAndSettle(durationAfterEachStep);
```

Listing 1: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/integration_test/app_test.dart

Zu diesem Zweck löst der Test nach der Auswahl der Hauptzielsetzung (Z. 118-119) nun einen Klick auf die Selektionskarte für die Nebenzielsetzung aus (Z. 121). Dadurch öffnet sich der Auswahlbildschirm, in welchem die Option *„Bodenschutz“* (Z. 122) und anschließend die Option *„Klima“* (Z. 123) gewählt wird. Mit Auswahl der letzten Option und durch die damit verbundene Übergabe des Arguments `true` für den optionalen Parameter `tabConfirm` wird der Auswahlbildschirm umgehend wieder geschlossen. Anschließend erfolgt erneut das Speichern der Maßnahme (Z. 125-126).

Anders als bei den bisherigen Schlüssel-Werte-Paaren innerhalb des Objektes `'massnahmenCharakteristika` kann der Wert der Nebenziele nicht als einzelner String gespeichert werden (Listing 2).

```
136 var expectedJson = {
137   'letzteBearbeitung': {'letzterStatus': 'fertig'},
138   'identifikatoren': {'massnahmenTitel': massnahmeTitle},
139   'massnahmenCharakteristika': {
140     'nebenziele': [
141       'bsch',
142       'klima',
143     ],
144     'foerderklasse': 'aukm_ohne_vns',
145     'kategorie': 'extens',
146     'zielflaeche': 'al',
147     'zieleinheit': 'ha',
148     'hauptzielsetzungLand': 'biodiv'
149   },
150 };
```

Listing 2: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/integration_test/app_test.dart

Bei dem Inhalt der Mehrfachauswahlfelder handelt es sich schließlich um eine Auflistung mehrerer Werte. Sie wird im erwarteten JSON-Dokument als Array-Literal codiert (Z. 140-143).

0.1.2 Hinzufügen der Menge der Nebenziele

Für die Menge der Nebenziele müssen keine weiteren Auswahloptionen hinzugefügt werden. Es werden die gleichen Optionen verwendet, die auch bei der Menge mit dem Namen *„Hauptzielsetzung Land“* zum Einsatz kommen (Listing 3, Z. 123-124).

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

Listing 3: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/choices/choices.dart

0.1.3 Aktualisierung des Models

Um die Liste der Nebenziele im Wertetyp `MassnahmenCharakteristika` einzufügen, kann der Datentyp `BuiltSet` verwendet werden (Listing 4, Z. 77). Die Getter-Methode `nebenziele`

```
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 4: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/data_model/massnahme.dart

bedarf keiner Null-Zulässigkeit, da das Nicht-Vorhandensein von Werten darüber erreicht werden kann, dass die Menge leer ist.

0.1.4 Aktualisierung der Übersichtstabelle

Für das Einfügen der Überschrift in der Übersichtstabelle gibt es keine Unterschiede zum bisherigen Vorgehen. Die Überschrift wird nach der Spaltenüberschrift für die *„Hauptzielsetzung“* eingefügt (Listing 5, Z. 28).

Die Anzeige der Werte in den TabellenZellen ist dagegen unterschiedlich (Listing 6). Dieses Mal handelt es sich um die Aufzählung von mehreren Werten, weshalb ein `Column`-Widget

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

Listing 5: XXXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/widgets/massnahmen_table.dart

die einzelnen Einträge untereinander auflistet (Z. 46-49). Jedes Element des `BuiltSet` `nebenziele` (Z. 47) wird über die Methode `map` jeweils in ein Element des Widgets `Text` konvertiert (Z. 48).

```
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 6: XXXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/widgets/massnahmen_table.dart

0.1.5 Aktualisierung des ViewModels

Die Nebenziele werden – erneut Mit dem Datentyp BuiltSet – im ViewModel hinzugefügt (Listing 7).

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

Listing 7: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart

Der benannte Konstruktor `seeded` initialisiert die Instanzvariable mit einer leeren Menge (Z. 20). Dafür wird der parameterlose Konstruktor von `BuiltSet` aufgerufen (Z. 21). Dadurch unterscheidet sich das `BehaviorSubject` von den anderen im ViewModel und muss dementsprechend bei der Konvertierung zwischen Model in ViewModel gesondert behandelt werden.

Bei Konvertierung von Model in ViewModel sind für alle Auswahloptionen – genau wie in den Schritten zuvor – jeweils nur die Abkürzungen verfügbar. Die Liste der gespeicherten Abkürzungen der Nebenziele muss dementsprechend zuerst in eine Menge von Auswahloptionen konvertiert werden, bevor sie dem `BuiltSet` übergeben werden kann (Listing 8). Die Methode `map` löst das Problem, indem sie die ihr als Argument übergebene Funktion für jede Abkürzung in der Menge *„Nebenziele“* aufruft (Z. 65). Die übergebene anonyme Funktion konvertiert die Abkürzung in die zugehörige Auswahloption. Die resultierende Menge kann dem Konstruktor von `BuiltSet` übergeben werden (Z. 64-65).

```
46 set model(Massnahme model) {
47   guid.value = model.guid;
48
49   letzterStatus.value = letzterStatusChoices
50       .fromAbbreviation(model.letzteBearbeitung.letzterStatus);
51   massnahmenTitel.value = model.identifikatoren.massnahmenTitel;
52
53   {
54     final mc = model.massnahmenCharakteristika;
55
56     foerderklasse.value =
57         foerderklasseChoices.fromAbbreviation(mc.foerderklasse);
58     kategorie.value = kategorieChoices.fromAbbreviation(mc.kategorie);
59
60     zielflaeche.value = zielflaecheChoices.fromAbbreviation(mc.zielflaeche);
61     zieleinheit.value = zieleinheitChoices.fromAbbreviation(mc.zieleinheit);
62     hauptzielsetzungLand.value =
63         hauptzielsetzungLandChoices.fromAbbreviation(mc.hauptzielsetzungLand);
64     nebenziele.value = BuiltSet(mc.nebenziele
65         .map((n) => hauptzielsetzungLandChoices.fromAbbreviation(n)));
66   }
67 }
```

Listing 8: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart

Ähnlich verhält es sich bei der Umwandlung des ViewModels in das Model (Listing 9).

```
69 Massnahme get model => Massnahme((b) => b
70   ..guid = guid.value
71   ..letzteBearbeitung.letzterStatus = letzterStatus.value?.abbreviation
72   ..letzteBearbeitung.letztesBearbeitungsDatum = DateTime.now().toUtc()
73   ..identifikatoren.update((b) => b..massnahmenTitel = massnahmenTitel.value)
74   ..massnahmenCharakteristika.update((b) => b
75     ..foerderklasse = foerderklasse.value?.abbreviation
76     ..kategorie = kategorie.value?.abbreviation
77     ..zielflaeche = zielflaeche.value?.abbreviation
78     ..zieleinheit = zieleinheit.value?.abbreviation
79     ..hauptzielsetzungLand = hauptzielsetzungLand.value?.abbreviation
80     ..nebenziele =
81         SetBuilder(nebenziele.value.map((n) => n.abbreviation).toList())));
```

Listing 9: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_form_view_model.dart

In diesem Fall muss die Menge der Auswahloptionen der *„Nebenziele“* in die entsprechenden Abkürzungen umgewandelt werden, bevor sie im *„Model“* gespeichert wird. Die Methode `map` er hält zu diesem Zweck erneut eine anonyme Funktion, welche die Abkürzung der Auswahloptionen abfragt (Z. 81). Die resultierende Menge wird als Parameter dem Konstruktor `SetBuilder` übergeben (Z. 80-81). Der `SetBuilder` wiederum kümmert sich um das Bauen des `BuiltSet`, sobald ein Objekt des Typs Massnahme gebaut wird.

0.1.6 Aktualisierung der Eingabemaske

Unterhalb des Auswahlfeldes für das Hauptziel wird die Selektionkarte für die Nebenziele eingefügt (Listing 10).

```
212 buildSelectionCard<ZielsetzungLandChoice>(
213     allChoices: hauptzielsetzungLandChoices,
214     selectionViewModel: vm.hauptzielsetzungLand),
215 buildMultiSelectionCard(
216     allChoices: nebenzielsetzungLandChoices,
217     selectionViewModel: vm.nebenziele),
```

Listing 10: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Allerdings handelt es sich dieses Mal um ein Mehrfachauswahlfeld, weshalb eine neue Methode namens `buildMultiSelectionCard` aufgerufen wird (Z. 215-217).

Da nun zwei Methoden zum Erstellen von Elementen des Widgets `SelectionCard` existieren, ist es sinnvoll, den Quellcode zu refaktorisieren, um redundanten Code zu vermeiden.

Innerhalb der bereits vorhandenen Methode `buildSelectionCard` wird die Routine, welche für die Validierung das Formulares genutzt wird, in eine neue Methode namens `validateChoices` (Listing 11, Z. 123-128) ausgelagert.

```
119 Widget buildSelectionCard<ChoiceType extends Choice>(
120     {required Choices<ChoiceType> allChoices,
121     required BehaviorSubject<ChoiceType?> selectionViewModel}) {
122   return FormField(
123       validator: (_) => validateChoices(
124           name: allChoices.name,
125           choices: {
126             if (selectionViewModel.value != null) selectionViewModel.value!
127           },
128           priorChoices: vm.priorChoices.value),
129       builder: (field) => SelectionCard<ChoiceType>(
```

Listing 11: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Sie bekommt die Attribute für den Namen der Menge (Z. 124), die zu validierenden Optionen (Z. 125-127) und schließlich die bisher ausgewählten Optionen aller Auswahlfelder (Z. 128) übergeben. Die ausgelagerte Funktion ist in Anhang A in Listing 21 auf Seite 14 zu finden.

Für die Erstellung der Mehrfachauswahlfelder ist die Methode `buildMultiSelectionCard` zuständig (Listing 12).

Das übergebene `selectionViewModel` unterstützt mit dem Typometer `BuiltSet` die Auswahl von mehreren Auswahloption (Z. 146). Bei `selectionViewModel` handelt es sich bereits um eine Menge. Für die Validierung `150` sowie für die Übergabe des initialen Wertes an den Konstruktor der `SelectionCard` (Z. 157) ist eine Umwandlung in eine Menge daher nicht mehr nötig. Dem Konstruktor `SelectionCard` wird weiterhin über den Parameter `multiSelection` mitgeteilt, dass mehr als eine Auswahl zum gewählt werden darf (Z. 154). Die Methoden `onSelect` und `onDeselect` ersetzen nun nicht mehr den aktuell gespeicherten Wert über eine einfache Zuweisung. Sie nutzen stattdessen die Methode `rebuild` des `BuiltSet` um ein

```
144 Widget buildMultiSelectionCard<ChoiceType extends Choice>(
145     {required Choices<ChoiceType> allChoices,
146     required BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel}) {
147   return FormField(
148       validator: (_) => validateChoices(
149           name: allChoices.name,
150           choices: selectionViewModel.value,
151           priorChoices: vm.priorChoices.value),
152       builder: (field) => SelectionCard<ChoiceType>(
153             title: allChoices.name,
154             multiSelection: true,
155             allChoices: allChoices,
156             priorChoices: vm.priorChoices,
157             initialValue: selectionViewModel.value,
158             onSelect: (selectedChoice) => selectionViewModel.value =
159                 selectionViewModel.value
160                     .rebuild((b) => b.add(selectedChoice)),
161             onDeselect: (selectedChoice) => selectionViewModel.value =
162                 selectionViewModel.value
163                     .rebuild((b) => b.remove(selectedChoice)),
164             errorText: field.errorText,
165           ));
166 }
```

Listing 12: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Element mit Hilfe von `add` hinzuzufügen (Z. 160) bzw. mit `remove` Elemente zu entfernen (Z. 163). Der Methodenaufruf `rebuild` sorgt jedoch nicht für das Hinzufügen oder Löschen am Original-Objekt, sondern erstellt eine Kopie der Liste mit der gewünschten Änderungen. Deshalb erfolgt eine Zuweisung der Kopie zum Wert des `BehaviorSubject`-Objekts, was wiederum das Auslösen eines neuen Ereignisses bewirkt (Z. 158,161).

0.1.7 Aktualisierung der Selektionskarte

Diese Selektionskarte wird um die Instanzvariable `multiSelection` erweitert (Listing 13, Z. 17), dessen Wert im Konstruktor übergeben wird (Z. 27) aber auch ausgelassen werden kann, da der Standardwert `false` angegeben ist.

```
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       this.multiSelection = false,
```

Listing 13: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart

Die Rückruffunktion `onChanged` des CheckboxListTile unterscheidet schließlich zwischen Mehrfach- und Einzel-Selektion. Sollte `multiSelection` mit `true` gesetzt sein (Z. 133), so erstellt die Methode `rebuild` von `BuiltSet` eine Kopie des aktuellen ViewModels der Selektionen. In der anonymen Funktion, welche für die Manipulationen an der Kopie genutzt wird, wird in einer Fallunterscheidung überprüft, ob das angewählte Element bereits selektiert ist (Z. 136). Sollte das der Fall sein, so wird diese bereits selektierte Option, die nun erneut angewählt wurde, mit der Methode `remove` des Builder-Objekts aus dem `BuiltSet` entfernt (Z. 137). Anderenfalls war die Option nicht selektiert, weshalb sie mit der Methode `add` hinzugefügt wird.

```
124 return CheckboxListTile(
125     key: Key(
126         "valid choice ${allChoices.name} - ${c.abbreviation}"),
127     controlAffinity: ListTileControlAffinity.leading,
128     title: Text(c.description),
129     tileColor: selectedButDoesNotMatch ? Colors.red : null,
130     value: isSelected,
131     onChanged: (selected) {
132       if (selected != null) {
133         if (multiSelection) {
134           selectionViewModel.value =
135               selectionViewModel.value.rebuild((b) {
136             if (selectionViewModel.value.contains(c)) {
137               return b.remove(c);
138             } else {
139               b.add(c);
140             }
141           });
142         } else {
143           selectionViewModel.value =
144               selectionViewModel.value.rebuild((b) {
145             b.replace(isSelected ? [] : [c]);
146           });
147         }
148         if (selected) {
149           onSelect(c);
150         } else {
151           onDeselect(c);
152         }
153       }
154     });
```

Listing 14: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/widgets/selection_card.dart

0.2 Schritt 7

```
Widget buildSelectionCard<ChoiceType extends Choice>(
    {required Choices<ChoiceType> allChoices,
    required BehaviorSubject<ChoiceType?> selectionViewModel,
    ChoiceMatcher<ChoiceType> choiceMatcher =
        defaultChoiceMatcherStrategy}) {
  return FormField(
      validator: (_) => validateChoices(
          name: allChoices.name,
          choices: {
            if (selectionViewModel.value != null) selectionViewModel.value!
          },
          priorChoices: vm.priorChoices.value,
```

Listing 15: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```
            errorText: field.errorText,
          ));
}

return Scaffold(
  appBar: AppBar(
    title: const Text('Maßnahmen Detail'),
  ),
  body: WillPopScope(
    onWillPop: () {
      if (inputsAreValidOrNotMarkedFinal()) {
        saveRecord();
        return Future.value(true);
      } else {
        showValidationError();
        return Future.value(false);
      }
    },
    child: Form(
      key: formKey,
      child: Stack(
        children: [
          SingleChildScrollView(
            child: Center(
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
```

Listing 16: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```
child: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Column(
    mainAxisSize: MainAxisSize.min,
    children: [
      FloatingActionButton(
        mini: true,
        heroTag: 'save_draft_floating_action_button',
        child: const Icon(Icons.paste, color: Colors.white),
        backgroundColor: Colors.orange,
        onPressed: saveDraftAndGoBackToOverviewScreen,
      ),
      const SizedBox(
        height: 10,
      ),
      FloatingActionButton(
        tooltip: saveMassnahmeTooltip,
        heroTag: 'save_floating_action_button',
        child: const Icon(Icons.check, color: Colors.white),
```

Listing 17: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-7/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

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

const confirmButtonTooltip = 'Auswahl übernehmen';

class SelectionCard<ChoiceType extends Choice> extends StatelessWidget {
  final String title;
  final bool multiSelection;
  final BehaviorSubject<BuiltSet<ChoiceType>> selectionViewModel;
  final Choices<ChoiceType> allChoices;
  final BehaviorSubject<Set<Choice>> priorChoices;
  final OnSelect<ChoiceType> onSelect;
  final OnDeselect<ChoiceType> onDeselect;
  final String? errorText;
  final ChoiceMatcher<ChoiceType> choiceMatcher;

  SelectionCard(
      {required this.title,
      this.multiSelection = false,
      required Iterable<ChoiceType> initialValue,
      required this.allChoices,
      required this.priorChoices,
      required this.onSelect,
      required this.onDeselect,
      required this.choiceMatcher,
      this.errorText,
      Key? key})
```

Listing 18: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart

```
return StreamBuilder(
    stream: needsRepaint,
    builder: (context, snapshot) {
      final selectedChoices = selectionViewModel.value;
      final bool wrongSelection =
          selectedChoices.any((c) => !choiceMatcher(c, priorChoices.value));

      return Card(
        child: Column(
```

Listing 19: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart

```
body: StreamBuilder(
    stream: selectionViewModel,
    builder: (context, snapshot) {
      final selectedChoices = selectionViewModel.value;

      Set<ChoiceType> selectedAndSelectableChoices = {};
      Set<ChoiceType> unselectableChoices = {};

      for (ChoiceType c in allChoices) {
        if (selectedChoices.contains(c) ||
            choiceMatcher(c, priorChoices.value)) {
          selectedAndSelectableChoices.add(c);
        } else {
          unselectableChoices.add(c);
        }
      }

      return ListView(children: [
        ...selectedAndSelectableChoices.map((ChoiceType c) {
          bool isSelected = selectedChoices.contains(c);
          bool selectedButDoesNotMatch =
              !choiceMatcher(c, priorChoices.value);

          return CheckboxListTile(
```

Listing 20: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-7/conditional_form/lib/widgets/selection_card.dart

Teil II

Anhang

Teil III

Implementierung

A Schritt 6 Anhang

```
276 String? validateChoices(
277     {required String name,
278     required Iterable<Choice> choices,
279     required Set<Choice> priorChoices}) {
280   if (choices.isEmpty) {
281     return "Feld ${name} enthält keinen Wert!";
282   }
283
284   bool atLeastOneValueInvalid =
285       choices.any((c) => !c.conditionMatches(priorChoices));
286
287   if (atLeastOneValueInvalid) {
288     return "Wenigstens ein Wert im Feld ${name} enthält ist fehlerhaft!";
289   }
290
291   return null;
292 }
```

Listing 21: XXXX, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-6/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart