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 3

In diesem Schritt soll die grundlegende Validierungsfunktion hinzugefügt werden. Maßnahmen, die als abgeschlossen markiert sind, dürfen keine leeren Eingabefelder enthalten und der Maßnahmentitel darf nicht doppelt belegt sein. `Flutter` stellt das Widget `Form` für die Validierung von Eingabefeldern bereit.

0.2 Einfügen des Form-Widgets

Das Widget `Form` ist ein Container, welcher die Validierung für alle Kinderelemente des Typs `FormField` ausführt. Damit es alle Eingabefelder im Formular umgibt, wird es zwischen dem `WillPopScope` und dem `Stack` eingefügt (Listing. 1, Z. 168). Darüber hinaus wird die Funktion `saveRecord` durch `validateAndSave` ersetzt (Z. 167), welche ebenfalls in diesem Schritt implementiert wird. Das `Form`-Widget muss über einen `key` registriert werden (Z. 169), damit auf die Validierungsfunktionen zurückgegriffen werden kann.

```
158 return Scaffold(
159   appBar: AppBar(
160     title: const Text('Maßnahmen Detail'),
161   ),
162   body: WillPopScope(
163     onWillPop: validateAndSave,
164     child: Form(
165       key: formKey,
166       child: Stack(
```

Listing 1: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Die Erstellung des `formKey` findet zu Beginn der `build`-Methode des Eingabeformulars statt (Listing. 2, Z. 20). Der `GlobalKey` identifiziert ein Element, welches durch ein Widget gebaut wurde, über die gesamte Applikation hinweg. Es erlaubt darüber hinaus auf das `State`-Objekt zuzugreifen, welches mit dem `StatefulWidget` verknüpft ist. Ohne Angabe eines Typparameters kann nur Zugriff auf Funktionen des Typs `State` gewährt werden. Doch die gewünschte Methode `validate` ist nur Teil des Typs `FormState`. Damit das Element, welches über den `GlobalKey` registriert wurde, auch den `FormState` liefert, kann der entsprechende Typparameter `<FormState>` bei der Erstellung des GlobalKey übergeben werden.

```
17 Widget build(BuildContext context) {
18   final vm = AppState.of(context).viewModel;
19   final model = AppState.of(context).model;
20   final formKey = GlobalKey<FormState>();
```

Listing 2: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

0.3 Validierung des Maßnahmentitels

Das Eingabefeld für den Maßnahmen-Titel ist ein `TextFormField` (Listing. 3, Z. 99). Es erbt vom Typ `FormField` und wird daher mit dem Vaterelement `Form` verknüpft. Es beinhaltet bereits einen Parameter für die Validierungsfunktion namens `validator` (Z. 104). Die übergebene Funktion erhält im ersten Parameter den für das Textfeld eingetragenen Wert. Die Funktion soll `null` zurückgeben, wenn keine Fehler in der Validierung geschehen sind. In jedem anderen Fall soll der Text zurückgegeben werden, der als Fehlermeldung

angezeigt werden soll. Sollte der Parameter `null` sein oder aber ein leerer String (Z. 105), so wird die entsprechende Fehlermeldung `'Bitte Text eingeben'` angezeigt (Z. 107). Damit der Benutzer direkt zu dem fehlerhaften Eingabefeld geführt wird, kann ein Objekt der Klasse `FocusNode` verwendet werden. Er wird vor der Konstruktion der Karte erstellt (Z. 95) und dem Parameter `focusNode` des `TextFormField` übergeben (Z. 100). Sollte ein Fehler bei der Validierung gefunden werden, kann mit der Methode `requestFocus` angeordnet werden, den Cursor in das betreffende Feld zu setzen (Z. 106). Das sorgt auch dafür, dass das Eingabefeld in den sichtbaren Bereich gerückt wird.

Sollte das Textfeld nicht leer sein, so soll noch überprüft werden, ob der Maßnahmen-Titel bereits vergeben ist. Über das Model kann die Liste der Maßnahmen angefordert werden (Z. 110). Die Funktion `any` akzeptiert als Argument eine Funktion, die für alle Elemente der Liste ausgeführt wird (Z. 110-113). Wenn die Rückgabe der Funktion auch nur in einem Fall `true` ist, so evaluiert auch `any` mit `true`. Andernfalls ist die Rückgabe `false`. Die anonyme Funktion schließt zunächst den Vergleich mit derselben Maßnahme aus, welche sich gerade in Bearbeitung befindet. Der Vergleich der guid ist dafür ausreichend. Sollte es eine andere Maßnahme geben, welche den gleichen Titel hat (Z. 112-113), so wird Die lokale Variable `massnahmeTitleDoesAlreadyExists` auf `true` gesetzt. Der Benutzer bekommt die entsprechende Fehlermeldung `'Dieser Maßnahmentitel ist bereits vergeben'` zu lesen `117`. Wenn keine der beiden Fallunterscheidungen das `return`-Statement (Z. 107, 117) auslöst, so erfolgt schließlich die Rückgabe von `null`. In dem Kontext der `validator`-Funktion bedeutet die Rückgabe von `null`, dass die Validierung erfolgreich war.

```
94  Widget createMassnahmenTitelTextFormField() {
95    final focusNode = FocusNode();
96    return Card(
97      child: Padding(
98        padding: const EdgeInsets.all(16.0),
99        child: TextFormField(
100         focusNode: focusNode,
101         initialValue: vm.massnahmenTitel.value,
102         decoration: const InputDecoration(
103             hintText: 'Maßnahmentitel', labelText: 'Maßnahmentitel'),
104         validator: (title) {
105           if (title == null || title.isEmpty) {
106             focusNode.requestFocus();
107             return 'Bitte Text eingeben';
108           }
109           var massnahmeTitleDoesAlreadyExists =
110               model.storage.value.massnahmen.any((m) =>
111                   m.guid != vm.guid.value &&
112                   m.identifikatoren.massnahmenTitel ==
113                       vm.massnahmenTitel.value);
114
115           if (massnahmeTitleDoesAlreadyExists) {
116             focusNode.requestFocus();
117             return 'Dieser Maßnahmentitel ist bereits vergeben';
118           }
119           return null;
120         },
121         onChanged: (value) {
122           vm.massnahmenTitel.value = value;
123         },
124       ),
125     ),
126   );
127 }
```

Listing 3: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Das `Form`-Widget validiert lediglich Kindelemente vom Typ `FormField`. Dementsprechend wird das Widget `SelectionCard` nicht in die Validierung miteinbezogen. Es erbt nicht von `FormField`. Es wäre möglich, eine weitere Klasse zu erstellen, die von `FormField` erbt und alle Parameter für die Erstellung einer Selektions-Karte wiederverwendet. Doch das würde bedeuten, dass für alle folgenden Schritte jeder weitere Parameter in beiden Konstruktoren der Klassen gepflegt werden müsste. Um der Arbeit leichter folgen zu können, wurde sich für einen anderen, simpleren Weg entschieden: Die Selektionskarte kann ebenso von einem `FormField` umgeben werden (Listing. 4, Z. 132-159), welches die Selektionskarte in der `builder`-Funktion erstellt und an den Parametern nichts ändert, außer einen weiteren hinzuzufügen: der Text für die Fehlermeldung (Z. 158). Der erste Parameterder `builder`-Funktion ist das `State`-Objekt das `FormField`. Es enthält die Getter-Methode `errorText`, die bei gegebenenfalls fehlgeschlagener Validierung die zurückgegebene Fehlermeldung enthält.

Die anonyme Funktion, die als Argument dem Parameter `validator` übergeben wird (Z. 133-143), erstellt eine temporäre Menge, die den Wert des `selectionViewModel` enthält, wenn dieser nicht `null` ist, andernfalls ist sie eine leere Menge (Z. 134-136). Die `validator`-Funktion gibt eine Fehlermeldung zurück, sollte die Menge leer sein (Z. 138-140). Ist die Menge dagegen gefüllt, so gibt sie `null` zurück, um mitzuteilen, dass die Validierung erfolgreich war (Z. 142).

```
129 Widget buildSelectionCard<ChoiceType extends Choice>(
130     {required Choices<ChoiceType> allChoices,
131     required BehaviorSubject<ChoiceType?> selectionViewModel}) {
132   return FormField(
133       validator: (_) {
134         Iterable<Choice> choices = {
135           if (selectionViewModel.value != null) selectionViewModel.value!
136         };
137 
138         if (choices.isEmpty) {
139           return "Feld ${allChoices.name} enthält keinen Wert!";
140         }
141 
142         return null;
143       },
144       builder: (field) => SelectionCard<ChoiceType>(
145             title: allChoices.name,
146             allChoices: allChoices,
147             initialValue: {
148               if (selectionViewModel.value != null)
149                 selectionViewModel.value!
150             },
151             onSelect: (selectedChoice) =>
152                 selectionViewModel.value = selectedChoice,
153             onDeselect: (selectedChoice) => selectionViewModel.value = null,
154             errorText: field.errorText,
155           ));
156 }
```

Listing 4: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Der `errorText` wird im Konstruktor der Klasse `SelectionCard` übergeben (Listing. 5, Z. 29). Da er `null` sein darf, ist er mit dem Suffix `?` als Typ mit Null-Zulässigkeit gekennzeichnet (Z. 21).

Durch Einfügen einer `Column` zwischen der `Card` (Listing. 6, Z. 53) und dem ListTile (Z. 57) kann die visuelle Repräsentation der Selektionskarte in der Höhe erweitert werden. Sollte der `errorText` gesetzt sein (Z. 65), so erscheint unter dem Titel und dem Untertitel

```dart
final OnSelect<ChoiceType> onSelect;
final OnDeselect<ChoiceType> onDeselect;
final String? errorText;

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

Listing 5: errorText wird der SelectionCard hinzugefügt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart

eine entsprechende Fehlermeldung (Z. 66-71).

```dart
return Card(
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      ListTile(
        focusNode: focusNode,
        title: Text(title),
        subtitle: Text(
            selectedChoices.map((c) => c.description).join(", ")),
        trailing: const Icon(Icons.edit),
        onTap: navigateToSelectionScreen,
      ),
      if (errorText != null)
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(errorText!,
              style:
                  const TextStyle(fontSize: 12.0, color: Colors.red)),
        )
    ],
  ),
);
```

Listing 6: errorText wird ausgegeben, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart

Oberhalb des vorhandenen `FloatingActionButton` wird nun ein weiterer eingefügt, der zum Speichern des Entwurfs mit der Funktion `saveDraft` genutzt werden soll (Listing. 7, Z. 210-216). Der ursprüngliche `FloatingActionButton` versucht die Validierung und die anschließende Speicherung der Maßnahme mithilfe der neuen Funktion `validateAndSave` (Z. 224).

```
children: [
  FloatingActionButton(
    mini: true,
    heroTag: 'save_draft_floating_action_button',
    child: const Icon(Icons.paste, color: Colors.white),
    backgroundColor: Colors.orange,
    onPressed: saveDraft,
  ),
  const SizedBox(
    height: 10,
  ),
  FloatingActionButton(
    tooltip: saveMassnahmeTooltip,
    heroTag: 'save_floating_action_button',
    child: const Icon(Icons.check, color: Colors.white),
    onPressed: validateAndSave,
  )
],
```

Listing 7: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```
void saveDraft() {
  ScaffoldMessenger.of(context)
    ..hideCurrentSnackBar()
    ..showSnackBar(
        const SnackBar(content: Text('Entwurf wird gespeichert ...')));

  var draft = vm.model.rebuild((b) =>
      b.letzteBearbeitung.letzterStatus = LetzterStatus.bearb.abbreviation);

  model.putMassnahmeIfAbsent(draft);
  Navigator.of(context).pop();
}
```

Listing 8: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```
Future<bool> validateAndSave() {
  if (inputsAreValidOrNotMarkedFinal()) {
    saveRecordAndGoBackToOverviewScreen();
    return Future.value(true);
  } else {
    showValidationError();
    return Future.value(false);
  }
}
```

Listing 9: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```
bool inputsAreValidOrNotMarkedFinal() {
  if (vm.letzterStatus.value != LetzterStatus.fertig) {
    return true;
  }

  if (formKey.currentState!.validate()) {
    return true;
  }
```

Listing 10: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

```
45 void showValidationError() {
46   ScaffoldMessenger.of(context).showSnackBar(SnackBar(
47       content: Row(
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 }
```

Listing 11: Die Maßnahmencharakteristika Selektionskarten werden ergänzt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-3/conditional_form/lib/screens/massnahmen_detail/massnahmen_detail.dart

Teil II

Anhang

A Schritt 2 Anhang

```dart
class FoerderklasseChoice extends Choice {
  static final oelb = FoerderklasseChoice("oelb", "Ökolandbau");
  static final azl = FoerderklasseChoice("azl", "Ausgleichszulage");
  static final ea = FoerderklasseChoice("ea", "Erschwernisausgleich");
  static final aukm_nur_vns = FoerderklasseChoice("aukm_nur_vns",
      "Agrarumwelt-(und Klima)Maßnahme: nur Vertragsnaturschutz");
  static final aukm_ohne_vns = FoerderklasseChoice("aukm_ohne_vns",
      "Agrarumwelt-(und Klima)Maßnahmen, tw. auch mit Tierwohlaspekten, aber OHNE
      ↪  Vertragsnaturschutz");
  static final twm_ziel = FoerderklasseChoice(
      "twm_ziel", "Tierschutz/Tierwohlmaßnahmen mit diesem als Hauptziel");
  static final contact =
      FoerderklasseChoice("contact", "bitte um Unterstützung");

  FoerderklasseChoice(String abbreviation, String description,
      {bool Function(Set<Choice> choices)? condition})
      : super(abbreviation, description);
}

final foerderklasseChoices = Choices<FoerderklasseChoice>({
  FoerderklasseChoice.oelb,
  FoerderklasseChoice.azl,
  FoerderklasseChoice.ea,
  FoerderklasseChoice.aukm_nur_vns,
  FoerderklasseChoice.aukm_ohne_vns,
  FoerderklasseChoice.twm_ziel,
  FoerderklasseChoice.contact
}, name: "Förderklasse");

class KategorieChoice extends Choice {
  static final zf_us =
      KategorieChoice("zf_us", "Anbau Zwischenfrucht/Untersaat");
  static final anlage_pflege =
      KategorieChoice("anlage_pflege", "Anlage/Pflege Struktur");
  static final dungmang = KategorieChoice("dungmang", "Düngemanagement");
  static final extens = KategorieChoice("extens", "Extensivierung");
  static final flst = KategorieChoice("flst", "Flächenstilllegung/Brache");
  static final umwandlg = KategorieChoice("umwandlg", "Nutzungsumwandlung");
  static final bes_kult_rass = KategorieChoice(
      "bes_kult_rass", "Förderung bestimmter Rassen / Sorten / Kulturen");
  static final contact = KategorieChoice("contact", "bitte um Unterstützung");

  KategorieChoice(String abbreviation, String description)
      : super(abbreviation, description);
}

final kategorieChoices = Choices<KategorieChoice>({
  KategorieChoice.zf_us,
  KategorieChoice.anlage_pflege,
  KategorieChoice.dungmang,
  KategorieChoice.extens,
  KategorieChoice.flst,
  KategorieChoice.umwandlg,
  KategorieChoice.bes_kult_rass,
  KategorieChoice.contact
}, name: "Kategorie");
```

Listing 12: Die Mengen foerderklasseChoices und kategorieChoices , Quelle: Eigenes Listing, Datei: Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart

```
class ZielflaecheChoice extends Choice {
  static final ka = ZielflaecheChoice("ka", "keine Angabe/Vorgabe");
  static final al = ZielflaecheChoice("al", "AL");
  static final gl = ZielflaecheChoice("gl", "GL");
  static final lf = ZielflaecheChoice("lf", "LF");
  static final dk_sk = ZielflaecheChoice("dk_sk", "DK/SK");
  static final hff = ZielflaecheChoice("hff", "HFF");
  static final biotop_le =
      ZielflaecheChoice("biotop_le", "Landschaftselement/Biotop o.Ä.");
  static final wald = ZielflaecheChoice("wald", "Wald/Forst");
  static final contact = ZielflaecheChoice("contact", "bitte um Unterstützung");

  ZielflaecheChoice(String abbreviation, String description)
      : super(abbreviation, description);
}

final zielflaecheChoices = Choices<ZielflaecheChoice>({
  ZielflaecheChoice.ka,
  ZielflaecheChoice.al,
  ZielflaecheChoice.gl,
  ZielflaecheChoice.lf,
  ZielflaecheChoice.dk_sk,
  ZielflaecheChoice.hff,
  ZielflaecheChoice.biotop_le,
  ZielflaecheChoice.wald,
  ZielflaecheChoice.contact
}, name: "Zielfläche");

class ZieleinheitChoice extends Choice {
  static final ka = ZieleinheitChoice("ka", "keine Angabe/Vorgabe");
  static final m3 = ZieleinheitChoice("m3", "m³ (z.B. Gülle)");
  static final pieces =
      ZieleinheitChoice("pieces", "Kopf/Stück (z.B. Tiere oder Bäume)");
  static final gve = ZieleinheitChoice("gve", "GV/GVE");
  static final rgve = ZieleinheitChoice("rgve", "RGV");
  static final ha = ZieleinheitChoice("ha", "ha");
  static final contact = ZieleinheitChoice("contact", "bitte um Unterstützung");

  ZieleinheitChoice(String abbreviation, String description)
      : super(abbreviation, description);
}

final zieleinheitChoices = Choices<ZieleinheitChoice>({
  ZieleinheitChoice.ka,
  ZieleinheitChoice.m3,
  ZieleinheitChoice.pieces,
  ZieleinheitChoice.gve,
  ZieleinheitChoice.rgve,
  ZieleinheitChoice.ha,
  ZieleinheitChoice.contact
}, name: "Zieleinheit");
```

Listing 13: Die Mengen zielflaecheChoices und zieleinheitChoices, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart

```dart
class ZielsetzungLandChoice extends Choice {
  static final ka = ZielsetzungLandChoice("ka", "keine Angabe/Vorgabe");
  static final bsch = ZielsetzungLandChoice("bsch", "Bodenschutz");
  static final wsch = ZielsetzungLandChoice("wsch", "Gewässerschutz");
  static final asch = ZielsetzungLandChoice("asch", "Spezieller Artenschutz");
  static final biodiv = ZielsetzungLandChoice("biodiv", "Biodiversität");
  static final strutktviel =
      ZielsetzungLandChoice("strutktviel", "Erhöhung der Strukturvielfalt");
  static final genet_res = ZielsetzungLandChoice("genet_res",
      "Erhaltung genetischer Ressourcen (Pflanzen, z. B. im Grünland, und Tiere, z. B.
      ↪  bedrohte Rassen)");
  static final tsch = ZielsetzungLandChoice(
      "tsch", "Tierschutz/Maßnahmen zum Tierwohl im Betrieb");
  static final klima = ZielsetzungLandChoice("klima", "Klima");
  static final contact =
      ZielsetzungLandChoice("contact", "bitte um Unterstützung");

  ZielsetzungLandChoice(String abbreviation, String description)
      : super(abbreviation, description);
}

final _zielsetzungLandChoices = {
  ZielsetzungLandChoice.ka,
  ZielsetzungLandChoice.bsch,
  ZielsetzungLandChoice.wsch,
  ZielsetzungLandChoice.asch,
  ZielsetzungLandChoice.biodiv,
  ZielsetzungLandChoice.strutktviel,
  ZielsetzungLandChoice.genet_res,
  ZielsetzungLandChoice.tsch,
  ZielsetzungLandChoice.klima,
  ZielsetzungLandChoice.contact
};

final hauptzielsetzungLandChoices = Choices<ZielsetzungLandChoice>(
    _zielsetzungLandChoices,
    name: "Hauptzielsetzung Land");
```

Listing 14: Die Menge hauptzielsetzungLandChoices, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-2/conditional_form/lib/choices/choices.dart