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

Hochschule Harz
Fachbereich Automatisierung und Informatik

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

Teil I

Implementierung

1 Schritt 1 - Formular in Grundstruktur erstellen

1.1 Integrations-Test zum Test der Oberfläche

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

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

Listing 1: Der Integration Test Driver, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/driver.dart

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

```
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 2: Initialisierung des Integrations Tests, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie

breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muß keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

```
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 3: Initialisierung des Widgets für den Integrations Tests, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

```dart
Future<void> tabSelectionCard(Choices choices, {Finder? ancestor}) async {
  final Finder textLabel;
  if (ancestor != null) {
    textLabel =
        find.descendant(of: ancestor, matching: find.text(choices.name));
  } else {
    textLabel = find.text(choices.name);
  }

  expect(textLabel, findsWidgets);

  final card = find.ancestor(of: textLabel, matching: find.byType(Card));
  expect(card, findsOneWidget);

  await tester.ensureVisible(card);
  await tester.tap(card);
  await tester.pumpAndSettle(durationAfterEachStep);
}

Future<void> tabConfirmButton() async {
  var confirmChoiceButton = find.byTooltip(confirmButtonTooltip);
  await tester.tap(confirmChoiceButton);
  await tester.pumpAndSettle(durationAfterEachStep);
}

Future<Finder> tabOption(Choice choice, {bool tabConfirm = false}) async {
  final choiceLabel = find.text(choice.description);
  expect(choiceLabel, findsOneWidget);

  var listTileKey = tester
      .element(choiceLabel)
      .findAncestorWidgetOfExactType<CheckboxListTile>()!
      .key!;
  var listTile = find.byKey(listTileKey);

  expect(listTile, findsOneWidget);

  await tester.ensureVisible(choiceLabel);
  await tester.tap(choiceLabel);
  await tester.pumpAndSettle(durationAfterEachStep);

  if (tabConfirm) {
    await tabConfirmButton();
  }

  return listTile;
}

Future<void> fillTextFormField(
    {required String title, required String text}) async {
  final textFormField = find
      .ancestor(of: find.text(title), matching: find.byType(TextFormField))
      .first;
  expect(textFormField, findsOneWidget);

  await tester.ensureVisible(textFormField);
  await tester.enterText(textFormField, text);
  await tester.pumpAndSettle(durationAfterEachStep);
}
```

Listing 4: Die Hilfsmethode tabSelectionCard, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

```dart
await tester.pumpAndSettle(durationAfterEachStep);

var createNewMassnahmeButton = find.byKey(createNewMassnahmeButtonKey);
final gesture = await tester.press(createNewMassnahmeButton);
await tester.pumpAndSettle(durationAfterEachStep);
await gesture.up();
await tester.pumpAndSettle(durationAfterEachStep);
```

Listing 5: Der Button zum Kreieren einer Maßnahme wird ausgelöst, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

```dart
await tabSelectionCard(letzterStatusChoices);
await tabOption(LetzterStatus.fertig, tabConfirm: true);
```

Listing 6: Der letzte Status wird ausgewählt, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

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

Listing 7: Der Maßnahmentitel wird eingegeben, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

```dart
var saveMassnahmeButton = find.byTooltip(saveMassnahmeTooltip);
await tester.tap(saveMassnahmeButton);
await tester.pumpAndSettle(durationAfterEachStep);
```

Listing 8: Der Button zum Speichern wird ausgelöst, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

```dart
var capturedJson =
    verify(massnahmenJsonFileMock.saveMassnahmen(captureAny)).captured.last;

var actualMassnahme = capturedJson['massnahmen'][0] as Map;
actualMassnahme.remove("guid");
actualMassnahme["letzteBearbeitung"].remove("letztesBearbeitungsDatum");

var expectedJson = {
  'letzteBearbeitung': {'letzterStatus': 'fertig'},
  'identifikatoren': {'massnahmenTitel': massnahmeTitle},
};

expect(actualMassnahme, equals(expectedJson));
```

Listing 9: Der Button zum Speichern wird ausgelöst, Quelle: Eigenes Listing, Datei: Quellcode/Schritt-1/conditional_form/integration_test/app_test.dart

Teil II

Anhang