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

Prof. Jürgen Singer Ph.D.
1. Prüfer

Prof. Daniel Ackermann
2. Prüfer

Inhaltsverzeichnis

Listingsverzeichnis

1 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 einem detaillierter untersucht werden. In Hinblick auf die Anforderungen an die Formular-Anwendung soll dabei die angemessenste Frontend-Technologie ausgewählt werden.

1.1 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 [1]

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 beinhaltet 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. [2]

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. [3]

1. Spolsky, „How Hard Could It Be?: The Unproven Path“
2. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*
3. Vgl. *Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe*

> Google Trends ist keine wissenschaftliche Umfrage und sollte nicht mit Umfragedaten verwechselt werden. Es spiegelt lediglich das Suchinteresse an bestimmten Themen wider. [4]

Genau aus diesem Grund wird Google Trends im folgenden lediglich zum Abgleich der Ergebnisse der Stackoverflow Umfrage eingesetzt.

1.1.1 Eingestellte Projekte

1.1.2 Stack Overflow Umfrage

4. *Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe*

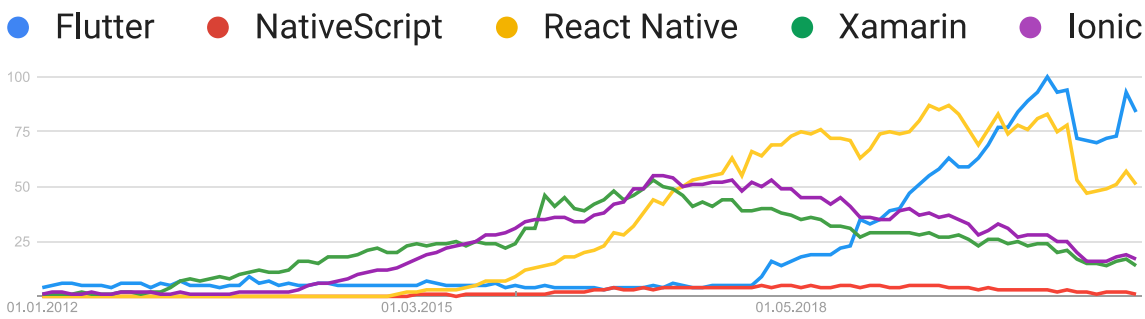

Abbildung 1: Ein QlikView Sheet mit diversen Sheet Objects, Quelle: Eigene Abbildung

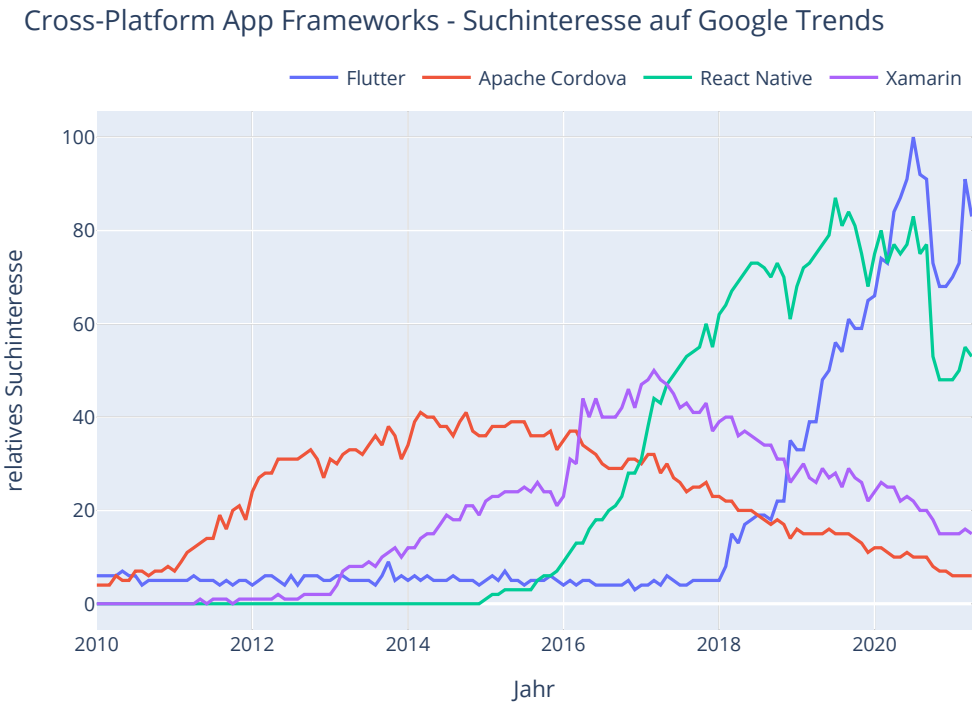

Abbildung 2: Ein QlikView Sheet mit diversen Sheet Objects, Quelle: Eigene Abbildung

Anhang

A Quelltexte

```
import 'package:flutter/material.dart';
import 'input.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) => MaterialApp(
        home: Scaffold(
          body: MyCustomForm(),
        ),
      );
}

final emailPattern = new RegExp(
    r'^(([^<>()\[\]\\.,;:\s@\enquote{]+(\.[^<>()\[\]\\.,;:\s@}
    ↪ ]+)*)|(\enquote{.+}
    ↪ ))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|
    ↪ (([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$');

class MyCustomForm extends StatefulWidget {
  @override
  MyCustomFormState createState() => MyCustomFormState();
}

class MyCustomFormState extends State<MyCustomForm> {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) => Form(
        key: _formKey,
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Column(
            children: [
              Input(label: \enquote{Name} ),
              Input(
                  label: \enquote{Email} ,
                  validator: (value) {
                    if (value == null || !emailPattern.hasMatch(value)) {
                      return 'Invalid Email Format';
                    }
                    return null;
                  }),
              Input(label: \enquote{Password} ),
              Padding(
                padding: const EdgeInsets.symmetric(vertical: 16.0),
                child: ElevatedButton(
                  onPressed: () {
                    if (_formKey.currentState!.validate()) {
                      ScaffoldMessenger.of(context).showSnackBar(
                          SnackBar(content: Text('Processing Data')));
                    }
                  },
                  child: Text('Submit'),
                ),
              ),
            ],
          ),
        ),
```

```
export default {
    name: {required: {value: true, message: 'Name is required'}},
    email: {
      required: {value: true, message: 'Email is required'},
      pattern: {
        value: /^(([^<>()\[\]\\.,;:\s@\enquote{]+(\.[^<>()\[\]\\.,;:\s@}
        ↪  ]+)*)|(\enquote{.+}
        ↪  ))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|
        ↪  (([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/,
        message: 'Invalid Email Format',
      },
    },
    password: {
      required: {value: true, message: 'Password is required'},
    },
  };
```

Listing 2: validation.tsx Datei

```
final emailPattern = new RegExp(
    r'^(([^<>()\[\]\\.,;:\s@\enquote{]+(\.[^<>()\[\]\\.,;:\s@}
    ↪  ]+)*)|(\enquote{.+}
    ↪  ))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|
    ↪  (([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$');

validateEmail(value) {
  if (value == null || !emailPattern.hasMatch(value))
    return 'Invalid Email Format';
  else
    return null;
}

validateNotEmpty(String label, value) {
  if (value == null || value.isEmpty)
    return '$label is required';
  else
    return null;
}
```

Listing 3: validation.dart Datei

```dart
import 'package:example_form_app/validation.dart';
import 'package:flutter/material.dart';

class Input extends StatelessWidget {
  final String label;
  final FormFieldValidator<String>? validator;

  const Input({required this.label, this.validator});

  @override
  Widget build(BuildContext context) => TextFormField(
      decoration: InputDecoration(labelText: label),
      validator: validator ?? (value) => validateNotEmpty(label, value));
}
```

Listing 4: input.dart Datei

```tsx
import * as React from 'react';
import { TextInput, KeyboardAvoidingView, findNodeHandle } from 'react-native';
import { ValidationOptions, FieldError } from 'react-hook-form';

interface ValidationMap {
  [key: string]: ValidationOptions;
}

interface ErrorMap {
  [key: string]: FieldError | undefined;
}

interface Props {
  children: JSX.Element | JSX.Element[];
  register: (
    field: { name: string },
    validation?: ValidationOptions
  ) => void;
  errors: ErrorMap;
  validation: ValidationMap;
  setValue: (name: string, value: string, validate?: boolean) => void;
}
```

Listing 5: Form.tsx Datei

```tsx
import * as React from 'react';
import { TextInput, KeyboardAvoidingView, findNodeHandle } from 'react-native';
import { ValidationOptions, FieldError } from 'react-hook-form';

interface ValidationMap {
  [key: string]: ValidationOptions;
}

interface ErrorMap {
  [key: string]: FieldError | undefined;
}

interface Props {
  children: JSX.Element | JSX.Element[];
  register: (
    field: { name: string },
    validation?: ValidationOptions
  ) => void;
  errors: ErrorMap;
  validation: ValidationMap;
  setValue: (name: string, value: string, validate?: boolean) => void;
}

```

Listing 6: Form.tsx Datei

```tsx
export default ({
  register,
  errors,
  setValue,
  validation,
  children,
}: Props) => {
  const Inputs = React.useRef<Array<TextInput>>([]);

  React.useEffect(() => {
    (Array.isArray(children) ? [...children] : [children]).forEach((child) => {
      if (child.props.name)
        register({ name: child.props.name }, validation[child.props.name]);
    });
  }, [register]);

  return (
    <>
      {(Array.isArray(children) ? [...children] : [children]).map(
        (child, i) => {
          return child.props.name
            ? React.createElement(child.type, {
                ...{
                  ...child.props,
                  ref: (e: TextInput) => {
                    Inputs.current[i] = e;
                  },
                  onChangeText: (v: string) =>
                    setValue(child.props.name, v, true),
                  onSubmitEditing: () => {
                    Inputs.current[i + 1]
                      ? Inputs.current[i + 1].focus()
                      : Inputs.current[i].blur();
                  },
                  //onBlur: () => triggerValidation(child.props.name),
                  blurOnSubmit: false,
                  //name: child.props.name,
                  error: errors[child.props.name],
                },
              })
            : child;
        }
      )}
    </>
  );
};
```

Listing 7: Form.tsx Datei

```tsx
import * as React from 'react';
import {
  View,
  TextInput,
  Text,
  StyleSheet,
  ViewStyle,
  TextStyle,
  TextInputProps,
} from 'react-native';
import { FieldError } from 'react-hook-form';
interface Props extends TextInputProps {
  name: string;
  label?: string;
  labelStyle?: TextStyle;
  error?: FieldError | undefined;
}

export default React.forwardRef<any, Props>(
  (props, ref): React.ReactElement => {
    const { label, labelStyle, error, ...inputProps } = props;

    return (
      <View style={styles.container}>
        {label && <Text style={[styles.label, labelStyle]}>{label}</Text>}
        <TextInput
          autoCapitalize=\enquote{none}
          ref={ref}
          style={[styles.input, { borderColor: error ? '#fc6d47' : '#c0cbd3' }]}
          {...inputProps}
        />
        <Text style={styles.textError}>{error && error.message}</Text>
      </View>
    );
  }
);

const styles = StyleSheet.create({
  container: {
    marginVertical: 8,
  },
  input: {
    borderWidth: 1,
    paddingLeft: 5,
  },
  label: {
    paddingVertical: 5,
  },
  textError: {
    color: '#fc6d47',
  },
});
```

Listing 8: Input.tsx Datei

```
import * as React from 'react';
import {
  Text,
  View,
  StyleSheet,
  Button,
  Alert,
  ScrollView,
} from 'react-native';

import { useForm } from 'react-hook-form';

import Input from './components/Input';
import Form from './components/Form';
import validation from './validation';

type FormData = {
  name: string;
  email: string;
  password: string;
};

export default () => {
  const { handleSubmit, register, setValue, errors } = useForm<FormData>();

  const onSubmit = (data: FormData) => {
    Alert.alert('data', JSON.stringify(data));
  };

  return (
      <View style={styles.formContainer}>
        <Form {...{ register, setValue, validation, errors }}>
          <Input name=\enquote{name} label=\enquote{Name } />
          <Input name=\enquote{email} label=\enquote{Email} />
          <Input name=\enquote{password} label=\enquote{Password}
          ↪  secureTextEntry={true} />
          <Button title=\enquote{Submit} onPress={handleSubmit(onSubmit)} />
        </Form>
      </View>
  );
};

const styles = StyleSheet.create({
  formContainer: {
    padding: 8,
    flex: 1,
  }
});
```

Listing 9: App.tsx Datei

Literatur

Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe. Mai 2021. URL: https://support.google.com/trends/answer/4365533 (Zitiert auf den Seiten 6, 7).

Spolsky, Joel. „How Hard Could It Be?: The Unproven Path“. In: *inc.com* (Nov. 2008). URL: https://www.inc.com/magazine/20081101/how-hard-could-it-be-the-unproven-path.html (Zitiert auf der Seite 6).

Stack Overflow Insights - Developer Hiring, Marketing, and User Research. Mai 2021. URL: https://insights.stackoverflow.com/survey/ (Zitiert auf der Seite 6).