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

1. Spolsky, „How Hard Could It Be?: The Unproven Path“
2. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*

das relative Such-Interesse abzubilden werden die Ergebnisse normalisiert um die Ergebnisse auf einer Skala von 0 bis 100 Darstellen zu können. [3]

> 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 Stack Overflow Umfrage eingesetzt.

1.1.1 Frameworks mit geringer Relevanz

NativeScript, Sencha (bzw. Sencha Touch) und Appcelerator spielen in den Umfrageergebnisse eine Untergeordnete Rolle. Dies ist in den aufsummierten Stimmen von 2013 bis 2020 für alle in der Umfrage auftauchenden Frontend-Technologien zu sehen (Abb. 1).

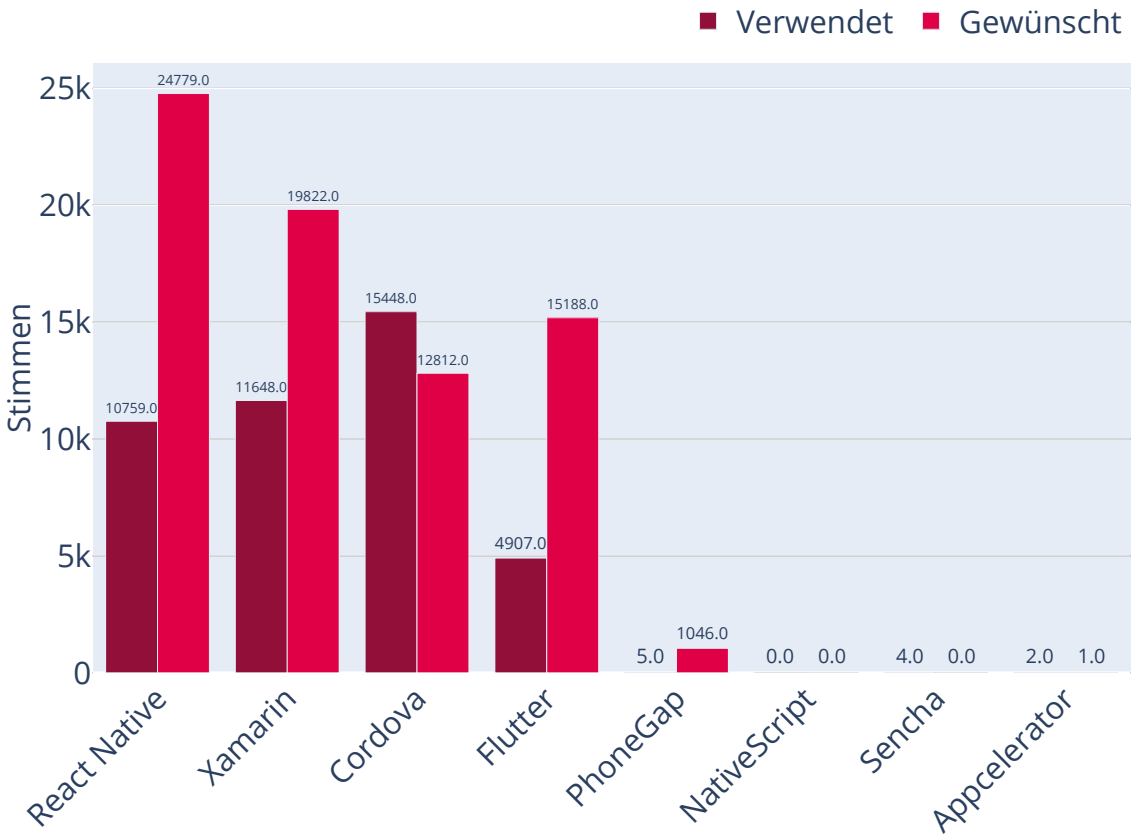

Abbildung 1: Summe der Stimmen der Stack Overflow Umfrage von 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

Auch das Suchinteresse auf Google ist für diese Frameworks äußerst gering. In Abbildung 2 werden NativeScript, Sencha, Appcelerator und auch Adobe PhoneGap mit Apache Cordova für das relative Suchinteresse verglichen.

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

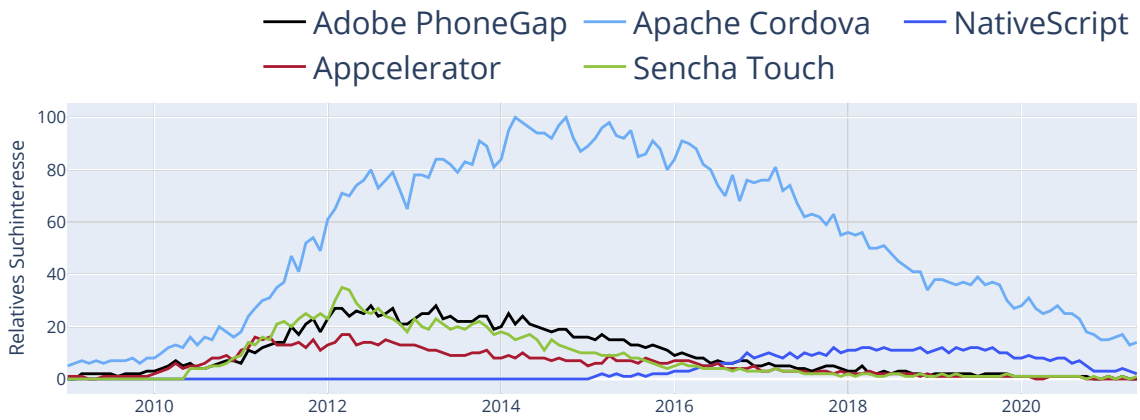

Abbildung 2: , Quelle: Eigene Abbildung, Notebook: Charts/Google Trends/Google Trends.ipynb, Daten-Quelle: Google Trends[5]

Verwandte Technologien zu Apache Cordova Das Ionic Framework taucht in den Ergebnissen der Stack Overflow Umfragen nicht auf. Ein Grund dafür könnte sein, dass es auf Apache Cordova aufbaut[6], welches bereits in den Ergebnissen vorkommt. Adobe PhoneGap taucht zwar in den Ergebnissen von 2013 mit 1043 Stimmen auf (Siehe Abbildung 3), verliert jedoch in den Folgejahren mit weniger als 10 Stimmen abrubt an Relevanz. Das stimmt nicht mit dem Suchinteresse auf Google überein, da es dort ab 2013 sogar steigt, wie in Abbildung 2 zu sehen. 2013 existierte PhoneGap noch als extra Mehrfachauswahlfeld in den Daten, während es ab 2014 nur noch in dem Feld für die sonstigen Freitext angaben auftaucht [7]. Auch Adobe PhoneGap baut auf Apache Cordova auf[8]. Für diese Auswertung spielen diese verwandten Technologien eine untergeordnete Rolle, da sie auch in den Google Trends weit hinter Apache Cordova zurückbleiben (Abb. 2).

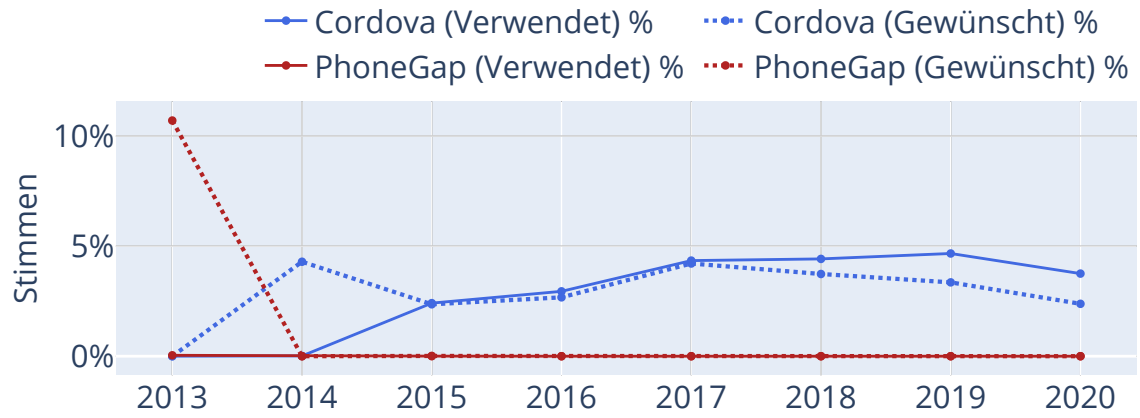

Abbildung 3: Stimmen für Cordova und PhoneGap 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

Am Beispiel von Adobe PhoneGap wird deutlich, wie wichtig es ist, auf eine Technologie zu setzen, die weit verbreitet ist. Im schlimmsten Fall wird die Technologie sogar vom Betreiber aufgrund zu geringer Nutzung komplett eingestellt, wie es be-

6. Lynch, *The Last Word on Cordova and PhoneGap*
7. Vgl. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*
8. Vgl. *FAQ | PhoneGap Docs*

reits bei PhoneGap geschehen ist. Adobe gab am 11. August 2020 bekannt, dass die Entwicklung an PhoneGap eingestellt wird und empfiehlt die Migration hin zu Apache Cordova.[9]

1.1.2 Frameworks mit sinkender Relevanz

Die Technologien Xamarin und Cordova zeigen bereits einen abfallenden Trend wie in Abbildung 4 ersichtlich ist. Im Fall von Xamarin gibt es immerhin mehr Entwickler, die sich wünschen, mit dem Framework zu arbeiten, als Entwickler, die tatsächlich mit Xamarin arbeiten. Cordova scheint in diesem Hinblick dagegen eher unbeliebt: es gibt mehr Entwickler, die mit Cordova arbeiten, als tatsächlich damit arbeiten wollen.

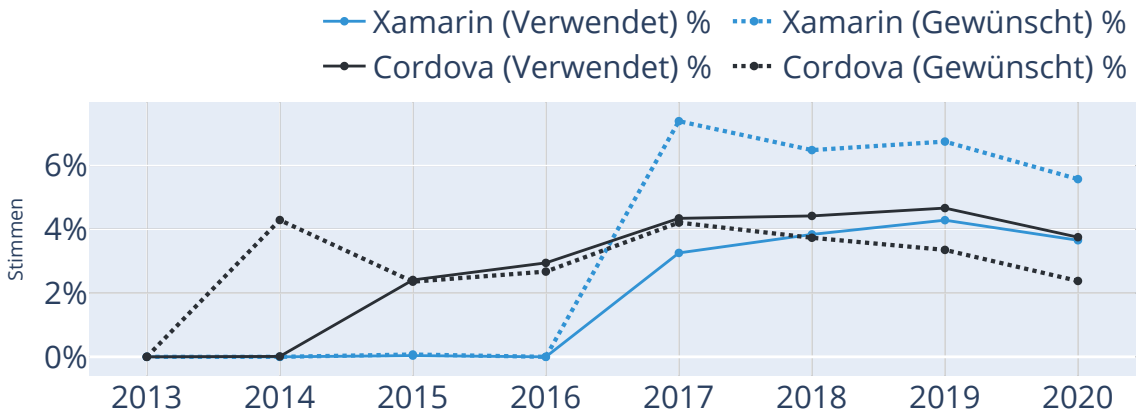

Abbildung 4: Stimmen für Xamarin und Cordova 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

In Abbildung 5 ist noch einmal zu sehen, das Google Trends die Erkenntnisse aus der Stack Overflow Umfrage reflektiert und es wird auch sichtbar, welche beiden Technologien möglicherweise der Grund für den Rückgang von Xamarin und Cordova sind.

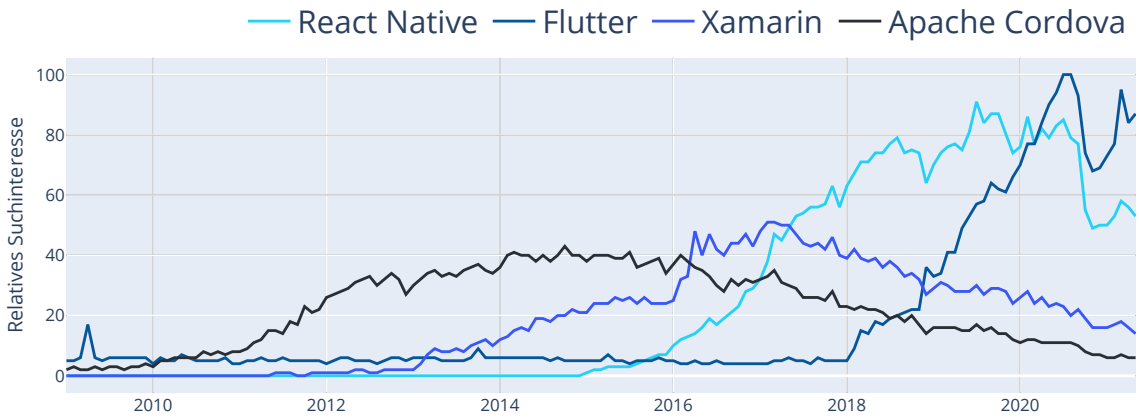

Abbildung 5: Suchinteresse sinkende und steigende Relevanz, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

9. Vgl. *Update for Customers Using PhoneGap and PhoneGap Build*

1.1.3 Frameworks mit steigender Relevanz

Besser ist es, auf Technologien zu setzen, die noch einen steigenden Trend der Verbreitung und Beliebtheit zeigen. In Abbildung 6 wird sichtbar, dass es sich dabei um Flutter und immerhin im Hinblick auf die Verbreitung auch für React Native handelt. Ungünstigerweise wird React Native in der Stack Overflow Umfrage erst seit 2018 als tatsächliches Framework abgefragt. Vorher erschien lediglich das Framework React, welches nicht für den Vergleich der Cross-PlatformFrameworks eignet, da es such um ein reines Web-Framework handelt. Doch auch die Ergebnisse von Google Trends zeigen einen ähnlichen Verlauf für die Jahre 2019 und 2020 (Abb. 5).

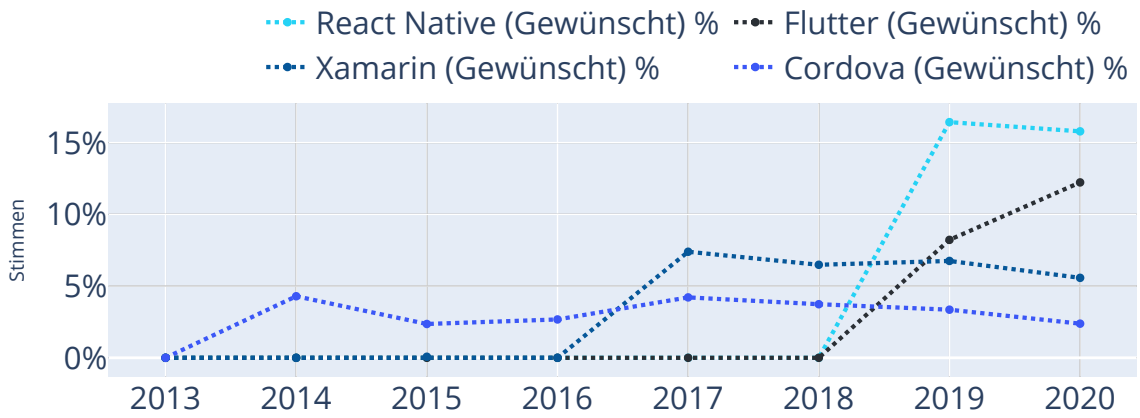

Abbildung 6: Stimmen für React Native und Flutter 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

Im Vergleich von dem Jahr 2019 mit 2020 wird sichbar, dass die Zahl der Entwickler, die sich wünschen, mit React Native zu arbeiten, gesunken ist. Dennoch ist die Anzahl der Entwickler, die mit React Native arbeiten möchten noch weit höher, als die der Entwickler, die tatsächlich mit React Native arbeiten.

Es ist möglich, dass der abfallende Trend daran liegt, dass die Zahl der Entwickler, die mit Flutter arbeiten möchten im selben Jahr gestiegen ist. React Native hat im Vergleich zu Flutter jedoch noch immer mehr aktive Entwickler und die Tendenz ist steigend. Doch die Anzahl der aktiven Flutter Entwickler zeigt einen noch stärker steigenden Trend. So könnte es sein, dass die Zahl der Flutter Entwickler die der React Native Entwickler in einem der nächsten Jahre überholt. Im Such-Interesse hat sich diese Entwicklung bereits vollzogen (Abb. 5).

Nichtsdestotrotz scheinen beide Technologien als Kandidaten für einen detaillierteren Vergleich für dieses Projekt in Frage zu kommen. Im nächsten Kapitel soll evaluiert werden, welches Framework für die Entwicklung der Formular-Anwendung angemessener ist.

1.2 Vergleich React Native und Flutter

1.2.1 Automatisiertes Testen

Automatisierte Test in React Native Die React Native Dokumentation führt genau eine Seite mit einem Überblick über die unterschiedlichen Testarten. Dabei wird das Konzept von Unit Tests, Mocking, Integrations Tests, Komponenten Tests und Snapshot Tests kurz erläutert, jedoch ohne ein Beispiel zu geben oder zu verlinken. Vier Quellcodeschnipsel sind auf der Seite zu finden: Ein Schnipsel zeigt den minimalen Aufbau eines Tests, zwei weitere Schnipsel veranschaulichen Beispielhaft, wie Nutzerinteraktionen getestet werden können und Letzteres zeigt die textuelle Representation der Ausgabe einer Komponente, die für einen Snapshottest verwendet wird. Weiterhin wird auf die Jest API Dokumentation verwiesen, sowie auf ein Beispiel für einen Snapshot Test in der Jest Dokumentation. [https://reactnative.dev/docs/testing-overview]

Um die notwendigen Anleitungen für das Erstellen der jeweiligen Tests ausfindig zu machen, ist es notwendig, die Dokumentation von React Native zu verlassen.

Die Dokumentation von Jest enthält mehr Details zum Einsatz der Testbibliothek, welches für mehrere auf Javascript basierende Frontend Framework kompatibel ist. [https://jestjs.io/docs/getting-started]. Somit muss zum Erstellen der Unit-Tests immerhin nur dieses Framework studiert werden.

Zum Entwickeln von Test von React Native Komponenten wird unter anderem auf die Bibliothek React Native Testing Library verwiesen. Anders als der Name vermuten lässt, handelt es sich nicht um eine von React Native bereitgestellte Bibliothek. Stattdessen wird sie vom Drittanbieter Callstack angeboten. [https://callstack.github.io/react-native-testing-library/docs/getting-started/] Sie verwendet im Hintergrund den React Test Renderer, welcher wiederum vom React Team angeboten wird und auch zum Testen von react.js Anwendungen geeignet ist. [https://reactjs.org/docs/test-renderer.html]. Der React Test Renderer Wird ebenfalls empfohlen, um Komponententest zu kreieren, die keine React Native spezifischen Funktionalitäten nutzen.

Um Integrationstest so entwickeln, welche die Applikation auf einem physischen Gerät oder auf einem Emulator testen, wird auf zwei weitere Drittanbieter Bibliotheken verlinkt: Appium [http://appium.io/] und Detox [https://github.com/wix/detox/]. Es wird darauf hingewiesen, dass Detox speziell für die Entwicklung von React Native Integrationstest entwickelt wurde. Appium wird lediglich als ein weiteres bekanntes Werkzeug erwähnt.

Es lässt sich damit zusammenfassen, dass der Aufwand der Einarbeitung für automatisiertes Testen in React Native vergleichsweise hoch ist. Die Dokumentation ist auf die Seiten der jeweiligen Anbieter verteilt. Der Entwickler muss sich den Überblick selbst verschaffen und zusätzlich die für das Framework React Native relevanten Inhalte identifizieren. Notwendig ist auch das Erlernen von mehreren APIs um alle Testarten abzudecken. Für einen Anfänger kommt erschwerend hinzu, dass eine Entscheidung für die eine oder andere Bibliothek notwendig wird. Um diese Entscheidung treffen zu können, ist eine Auseinandersetzung mit den Vor- und Nachteile der Technologien im Vorfeld vom Entwickler zu leisten.

Automatisierte Test in Flutter Die Flutter Dokumentation erklärt sehr umfangreich auf 11 Unterseiten die unterschiedlichen Testarten mit Quellcodebeispielen und verlinkt für jede Testart eine bis mehrere detaillierte Schritt-für-Schritt-Anleitungen, wie ein solcher Test erstellt wird.

Einer Seite erklärt den Unterschied zwischen Unit Test, Widget Test und Integrationstest [https://flutter.dev/docs/testing]. Eine weitere Seite erklärt Integrationstests in mehr Detail [https://flutter.dev/docs/testing/integration-tests].

Ein sogenanntes Codelab führt durch die Erstellung einer minimalistischen App und zwei Unit-, fünf Widget- und zwei Integrationstest für diese App [https://codelabs.developers.go
app-testing]

Im sogenannten Kochbuch tauchen folgende Rezepte auf:

- 2 Rezepte für Unit Tests
  - eine grundlegende Anleitung zum Erstellen von Unit-Tests [https://flutter.dev/docs/
  - Eine weitere Anleitung zum Nutzen von mocks in Unit Test mithilfe der Bibliothek mockito [https://flutter.dev/docs/cookbook/testing/unit/mocking]
- 3 Rezepte für Widget Tests
  - Eine grundlegende Anleitung zum Erstellen von Widget Test [https://flutter.dev/doc
  - Ein Rezept mit detaillierteren Beispielen zum Finden von widgets zur Laufzeit eines Widget Tests [https://flutter.dev/docs/cookbook/testing/widget/find
  - Ein Rezept zum Testen von Nutzerverhalten wie dem Tab, dem Drag und dem eingeben von Text [https://flutter.dev/docs/cookbook/testing/widget/tap-drag]
- 3 Rezepte für Integrationstests
  - Eine grundlegende Anleitung zum Erstellen eines Integrationstest [https://flutter.dev/docs/cookbook/testing/integration/introduction]
  - eine Anleitung zum simulieren von scrollen in der Anwendung während der Laufzeit eines Integrationstest [https://flutter.dev/docs/cookbook/testing/integr
  - eine Anleitung zum Performance Profiling [https://flutter.dev/docs/cookbook/testin

Zusammengefasst: Der Aufwand der Einarbeitung in das Testen in Flutter ist gering. Alle Werkzeuge werden vom Dart- und Flutter-Team bereitgestellt. Die Dokumentation ist umfangreich, folgt jedoch einem roten Faden. Eine Übersichtsseite fasst die Kerninformationen zusammen und verweist auf die jeweiligen Seiten für detailliertere Informationen und Übungen.

Anhang

A Technologiewahl Anhang

A.1 Stimmen verwendeter Frameworks

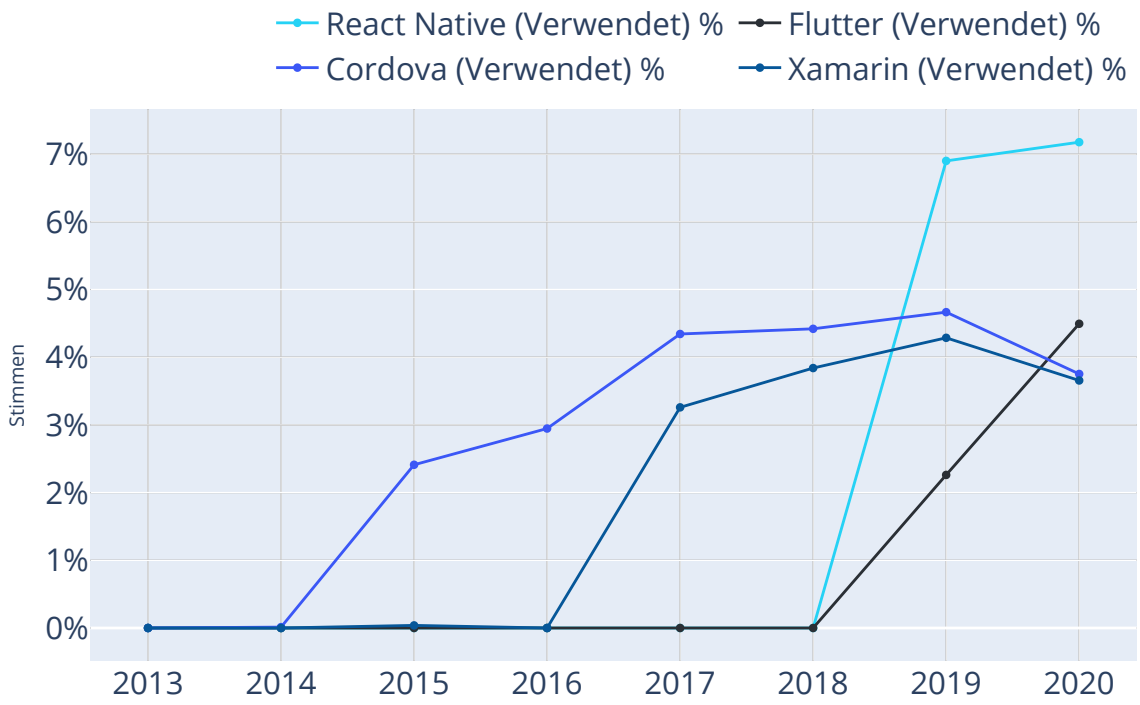

Abbildung 7: Stimmen verwendeter Frameworks, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

A.2 Stimmen gewünschter Frameworks

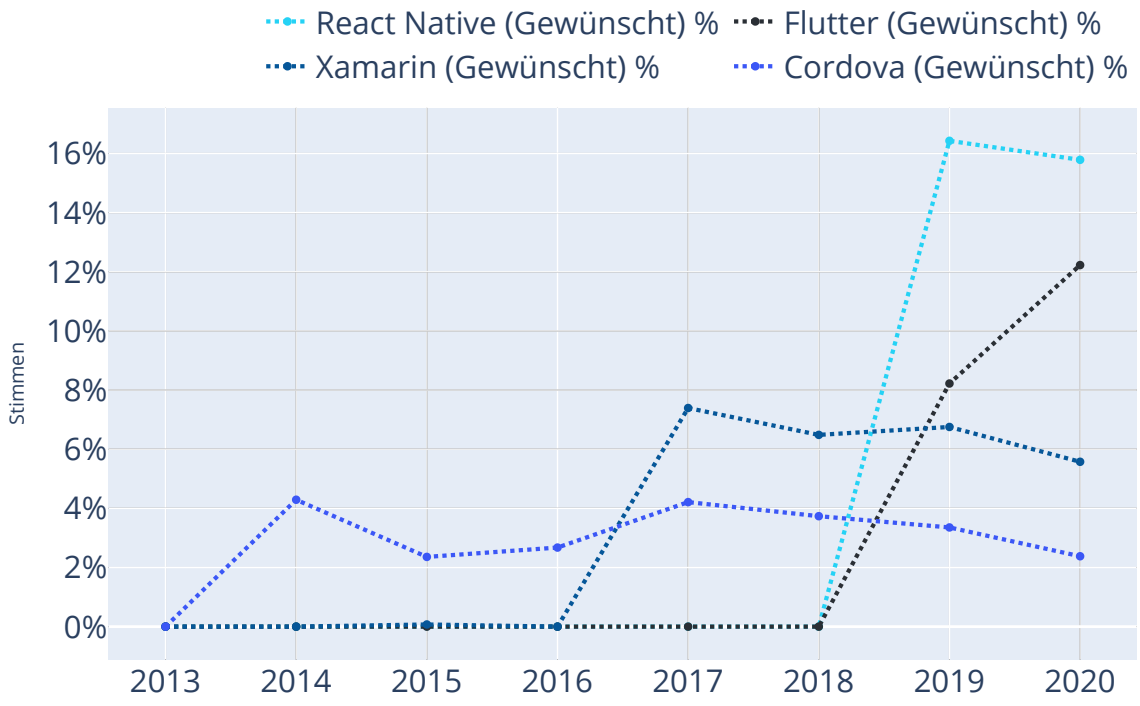

Abbildung 8: Stimmen gewünschter Frameworks, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

B Vergleich React Native und Flutter Anhang

```dart
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{.+}
    ↪  ]+)*)|(\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

```tsx
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

FAQ | PhoneGap Docs. Aug. 2016. URL: https://web.archive.org/web/20200806024626/http://docs.phonegap.com/phonegap-build/faq/ (Zitiert auf der Seite 9).

Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe. Mai 2021. URL: https://support.google.com/trends/answer/4365533 (Zitiert auf der Seite 8).

Lynch, Max. *The Last Word on Cordova and PhoneGap*. März 2014. URL: https://web.archive.org/web/20210413012559/https://blog.ionicframework.com/what-is-cordova-phonegap/ (Zitiert auf der Seite 9).

Spolsky, Joel. „How Hard Could It Be?: The Unproven Path“. In: *inc.com* (Nov. 2008). URL: http://web.archive.org/web/20081108094045/http://www.inc.com/magazine/20081101/how-hard-could-it-be-the-unproven-path.html (Zitiert auf der Seite 7).

Stack Overflow Insights - Developer Hiring, Marketing, and User Research. Mai 2021. URL: https://insights.stackoverflow.com/survey/ (Zitiert auf den Seiten 7, 9).

Update for Customers Using PhoneGap and PhoneGap Build. Aug. 2020. URL: https://web.archive.org/web/20200811121213/https://blog.phonegap.com/update-for-customers-using-phonegap-and-phonegap-build-cc701c77502c?gi=df435eca31bb (Zitiert auf der Seite 10).