

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

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 Auswahlmöglichkeiten mitunter zahlreich sein. Dem Nutzer sollen daher nur solche Auswahlmöglichkeiten angeboten werden, die zusammen mit der zuvor getroffenen Auswahl sinnvoll sind.

Im Wesentlichen ergibt sich die Kompatibilität der Auswahlmöglichkeiten 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	6
1 Anforderungen	7
2 Technologie Auswahl	8
2.1 Trendanalyse	8
2.1.1 Frameworks mit geringer Relevanz	9
2.1.2 Frameworks mit sinkender Relevanz	10
2.1.3 Frameworks mit steigender Relevanz	11
2.2 Vergleich React Native und Flutter	12
2.2.1 Vergleich zweier minimaler Beispiele für Formulare und Validierung .	12
2.2.2 Automatisiertes Testen	13
Anhang	19
A Technologiewahl Anhang	20
A.1 Stimmen verwendeter Frameworks	20
A.2 Stimmen gewünschter Frameworks	20
B Vergleich React Native und Flutter Anhang	21
Eidesstattliche Erklärung	29

Abbildungsverzeichnis

1	Stimmen der Stack Overflow Umfrage von 2013 bis 2020	9
3	Stimmen für Cordova und PhoneGap 2013 bis 2020	10
4	Stimmen für Xamarin und Cordova	11
5	Suchinteresse sinkende und steigende Relevanz	11
6	Stimmen für React Native und Flutter	12
7	Stimmen verwendeter Frameworks	20
8	Stimmen gewünschter Frameworks	20

Listingsverzeichnis

1	Schritt 3 errorText wird der SelectionCard hinzugefügt	16
2	Schritt 3 errorText wird ausgegeben	16
3	built_value Live Template	17
4	built_value Live Template	17
5	built_value Live Template	17
6	built_value Live Template	18

1 Anforderungen

Dieses Kapitel behandelt die Anforderungen

- Performance: Hohe Anzahl Eingabefelder
- nested formulars

Wunsch

- Alle Komponenten wiederverwendbar wie etwa Selection Caard nicht nur für Choices
-
- Strategy Entwurfsmuster für compute choice von viewModelType weil viewmodel nötig ist für nested formular
-

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

2.1 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 beinhalten 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

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

¹Spolsky, „How Hard Could It Be?: The Unproven Path“

²Stack Exchange, Inc., *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*

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

⁴Google LLC, *Häufig gestellte Fragen zu Google Trends-Daten - Google Trends-Hilfe*

Genau aus diesem Grund wird Google Trends im Folgenden lediglich zum Abgleich der Ergebnisse der Stack Overflow Umfrage eingesetzt.

2.1.1 Frameworks mit geringer Relevanz

NativeScript, Sencha (bzw. Sencha Touch) und Appcelerator spielen in den Umfrageergebnissen 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).

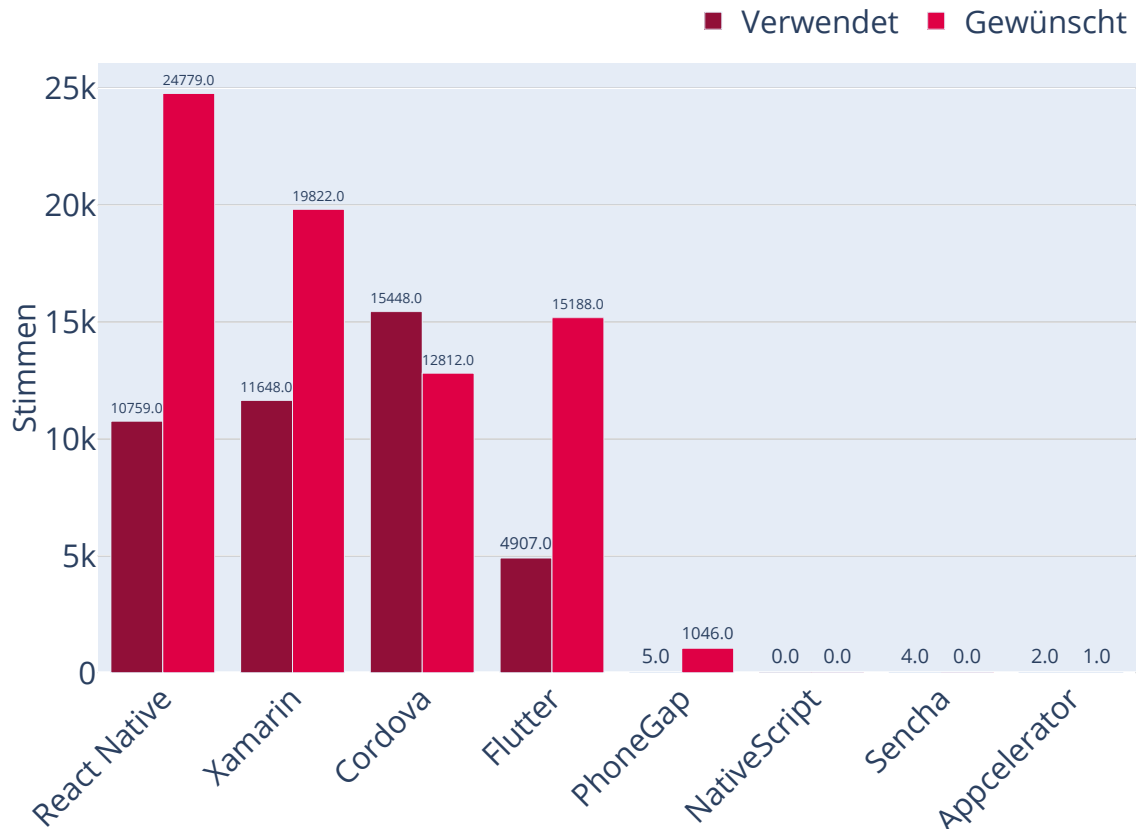


Abbildung 1: Summe der Stimmen der Stack Overflow Umfrage von 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

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.

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 aufbaut⁶, 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 abruht an Relevanz. Das stimmt nicht mit dem Suchinteresse auf Google überein, da es dort ab 2013 sogar steigt, wie in Abbildung 2 zu sehen ist. 2013 existierte PhoneGap noch als extra Mehrfachauswahlfeld in den Daten, während es ab 2014 nur noch in dem Feld für die sonstigen Freitext Angaben auftaucht⁷. Auch Adobe PhoneGap baut auf Apache Cordova auf⁸. Für diese

⁶Lynch, *The Last Word on Cordova and PhoneGap*

⁷Vgl. Stack Exchange, Inc., *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*

⁸Vgl. Adobe Inc., *FAQ / PhoneGap Docs*

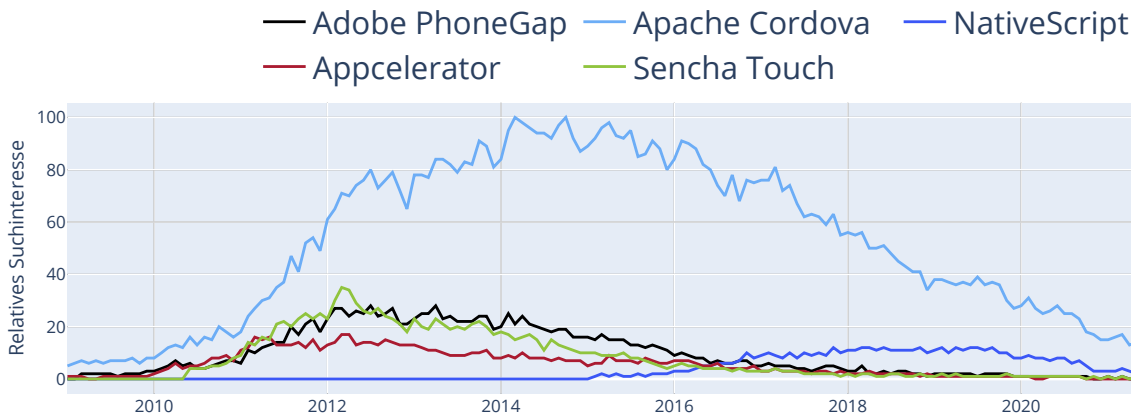


Abbildung 2: , Quelle: Eigene Abbildung, Notebook: Charts/Google Trends/Google Trends.ipynb, Daten-Quelle: Google Trends⁵

Auswertung spielen diese verwandten Technologien eine untergeordnete Rolle, da sie auch in den Google Trends weit hinter Apache Cordova zurückbleiben (Abb. 2).

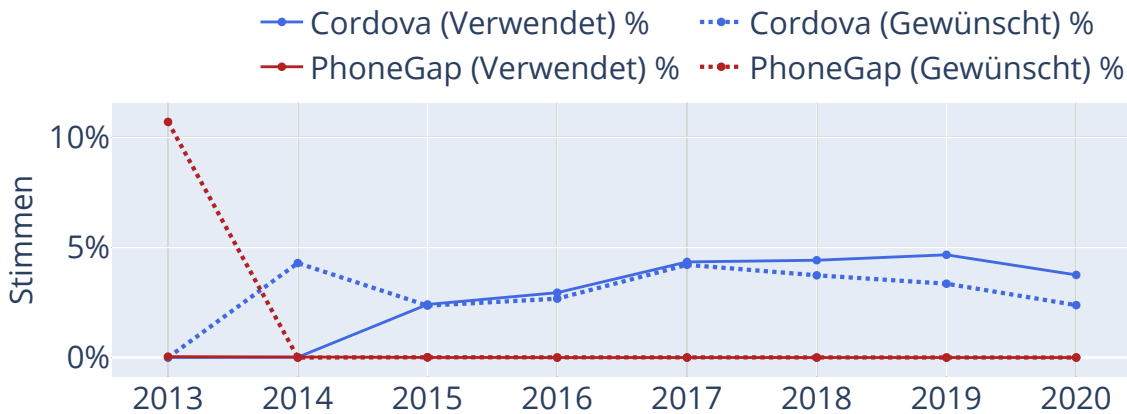


Abbildung 3: Stimmen für Cordova und PhoneGap 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

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 bereits 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

2.1.2 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.

In Abbildung 5 ist noch einmal zu sehen, dass 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.

⁹Vgl. Adobe Inc., *Update for Customers Using PhoneGap and PhoneGap Build*

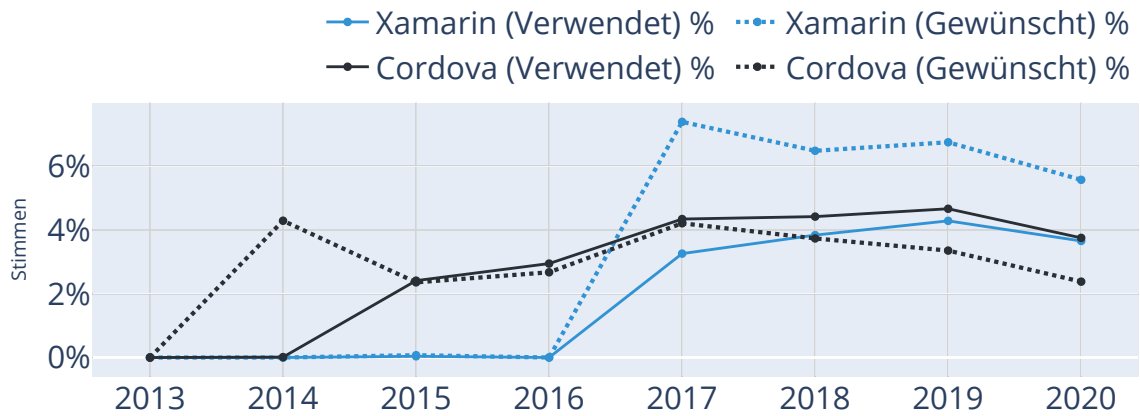


Abbildung 4: Stimmen für Xamarin und Cordova 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

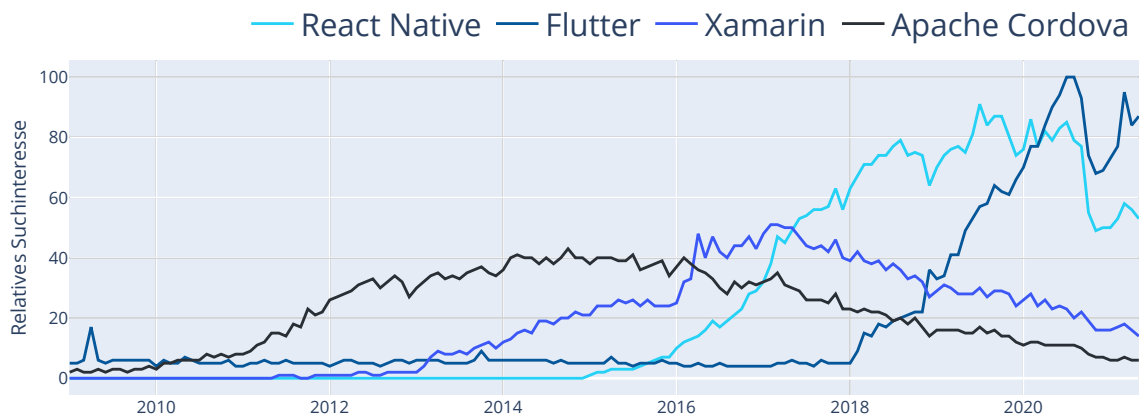


Abbildung 5: Suchinteresse sinkende und steigende Relevanz, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

2.1.3 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 sich 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).

Im Vergleich von dem Jahr 2019 mit 2020 wird sichtbar, 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).

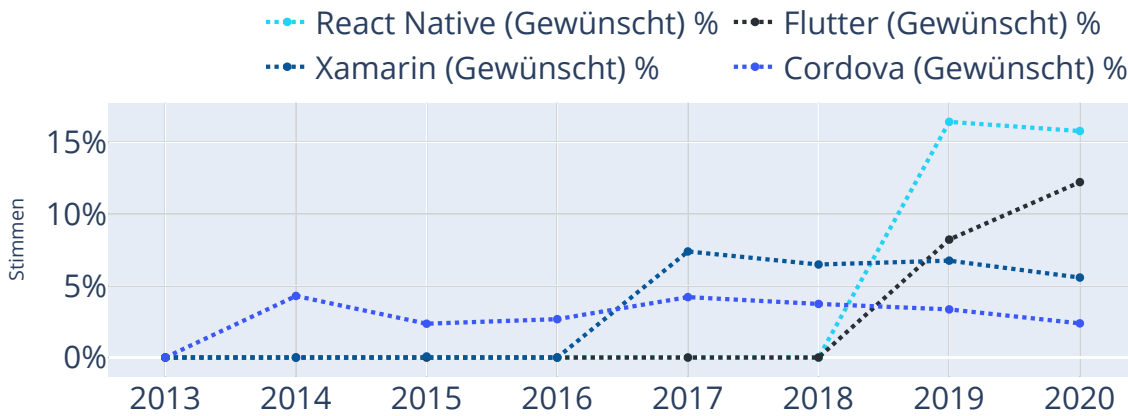


Abbildung 6: Stimmen für React Native und Flutter 2013 bis 2020, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

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.

2.2 Vergleich React Native und Flutter

2.2.1 Vergleich zweier minimaler Beispiele für Formulare und Validierung

Es soll eine Formularanwendung mit komplexer Validierung im Rahmen dieser These erstellt werden. Es ist durchaus sinnvoll, die beiden Technologien anhand von Beispielanwendungen, welche Formulare und die Validierung dieser beinhalten, zu vergleichen. Deshalb sollen nachfolgend jeweils eine solche Beispielanwendung der jeweiligen Technologie gefunden werden. Die Anwendung werden sich stark voneinander unterscheiden, weshalb sie im nächsten Schritt vereinfacht und aneinander angeglichen werden. Anschließend wird ersichtlich werden, nach welchen Kriterien sich die Technologien im Hinblick auf die Entwicklung der Formularanwendung vergleichen lassen.

React Native React native stellt nur eine vergleichsweise geringe Anzahl von eigenen Komponenten zur Verfügung und zu diesen gehören keine, welche die Validierung von Formularen ermöglichen. Doch die im react.js Raum sehr bekannten Bibliotheken Formic, Redux Forms und React Hook Form sind alle drei kompatibel mit React Native.^{10,11,12}

Für die Formular-Anwendung ist die Validierung komplexer Bedingungen nötig. Die Formular-Validierungs-Bibliotheken bieten in der Regel Funktionen an, welche überprüfen, ob ein Feld gefüllt ist oder der Inhalt einem speziellen Muster entspricht – wie etwa einem regulären Ausdruck. Doch solche mitgelieferten Validierungs-Funktionen reichen nicht aus um die Komplexität der Bedingungen abzubilden. Stattdessen müssen benutzerdefinierte Funktionen zum Einsatz kommen.

Keiner der drei oben genannten Validierungs-Bibliotheken ist in dieser Hinsicht limitiert. Sie alle bieten die Möglichkeit, eine JavaScript Funktion für die Validierung zu übergeben. Diese Funktion gibt einen Wahrheitswert zurück – wahr, wenn das Feld oder die Felder valide sind, falsch, falls nicht. In React Hook Form ist es die Funktion register,

¹⁰Vgl. *React Native / Formik Docs*

¹¹Vgl. *Does redux-form work with React Native?*

¹²Vgl. *React Native / React Hook Form - Get Started*

die ein Parameter-Object namens Register Options erhält, dessen Eigenschaft validate die JavaScript Funktion zugewiesen werden kann.¹³ In Redux Form ist es die initialisierungsfunktion reduxForm, die ein Konfigurations-Objekt mit dem Namen config erhält, in welchem die Eigenschaft ebenfalls validate heißt.¹⁴ Auch in Formik ist der Bezeichner validate, und ist als Attribut in der Formik Komponente zu finden.¹⁵

Es ist also absehbar, dass die Formular-Anwendung in React Native entwickelt werden kann. Die nötigen Funktionen werden von den Bibliotheken bereitgestellt. Einziger Nachteil hierbei ist, dass es sich um Drittanbieter Bibliotheken handelt, welche im Verlauf der Zeit an Beliebtheit gewinnen und verlieren können. Möglicherweise geht die Beliebtheit einer der Bibliothek mit der Zeit zurück, weshalb es weniger Kontributionen wie etwa neue Funktionalitäten oder Fehlerbehebungen, sowie Fragen und Antworten und Anleitungen zu diesen Bibliotheken geben wird, da die Entwickler sich für andere Bibliotheken entscheiden. Die Wahl der Bibliothek kann also schwerwiegende Folgen wie Mangel an Dokumentation oder Limitationen im Vergleich zu anderen Bibliotheken mit sich bringen. Eine Migration von der einen Bibliothek zu einer anderen könnte in Zukunft notwendig werden, wenn diese Limitationen während der Entwicklung auffallen. Aus dem Grund ist es in der Regel von Vorteil, wenn solche Funktionalitäten bereits im Kern der Frontend-Technologie integriert sind. Der Fall, dass die Kern-Komponenten an Relevanz verlieren und empfohlen wird, auf externe Bibliotheken zuzugreifen, ist zwar nicht ausgeschlossen, geschied aber im wesentlichen seltener.

Flutter Die Flutter Dokumentation stellt in ihrer cookbook Sektion ein Beispiel einer minimalistischen Formularanwendung mit Validierung bereit.¹⁶ Das Rezept ist Teil einer Serie von insgesamt fünf Anleitungen, welche Formulare in Flutter behandeln.¹⁷

2.2.2 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.^I

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^{II}. Somit muss zum Erstellen der Unit-Tests immerhin nur dieses Framework studiert werden.

Zum Entwickeln von Test von React Testing Library React Native Komponenten wird

¹³Vgl. *register* / *React Hook Form* - API

¹⁴Vgl. *reduxForm* / *Redux Form* - API

¹⁵Vgl. *<Formik />* / *Formik Docs* API

¹⁶Vgl. Google LLC, *Build a form with validation*

¹⁷Vgl. Google LLC, *Forms* / *Flutter Docs Cookbook*

^I<https://jestjs.io/docs/snapshot-testing>

^{II}<https://jestjs.io/docs/getting-started>

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. Im Unterschied zur React Testing Library, von der sie inspiriert ist, läuft sie ebenso wie React Native selbst nicht in einer Browser-Umgebung.¹⁸ Herausgegeben wird die react native testing Library vom Drittanbieter Callstack - ein Partner im React Native Ökosystem.¹⁹

Sie verwendet im Hintergrund den React Test Renderer^{III}, welcher wiederum vom React Team angeboten wird und auch zum Testen von react.js Anwendungen geeignet ist. 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^{IV} und Detox^V. 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^{VI}. Eine weitere Seite erklärt Integrationstests in mehr Detail^{VII}.

Ein sogenanntes Codelab führt durch die Erstellung einer minimalistischen App und zwei Unit-, fünf Widget- und zwei Integrationstest für diese App^{VIII}

Im sogenannten Kochbuch tauchen folgende Rezepte auf:

- 2 Rezepte für Unit Tests

– eine grundlegende Anleitung zum Erstellen von Unit-Tests^{IX}

¹⁸Vgl. Borenkraout, *Native Testing Library Introduction | Testing Library Docs*

¹⁹Vgl. Facebook Inc., *The React Native Ecosystem*

^{III}<https://reactjs.org/docs/test-renderer.html>

^{IV}<http://appium.io/>

^V<https://github.com/wix/detox/>

^{VI}<https://flutter.dev/docs/testing>

^{VII}<https://flutter.dev/docs/testing/integration-tests>

^{VIII}<https://codelabs.developers.google.com/codelabs/flutter-app-testing>

^{IX}<https://flutter.dev/docs/cookbook/testing/unit/introduction>

- Eine weitere Anleitung zum Nutzen von mocks in Unit Test mithilfe der Bibliothek mockito ^X
- 3 Rezepte für Widget Tests
 - Eine grundlegende Anleitung zum Erstellen von Widget Test ^{XI}
 - Ein Rezept mit detaillierteren Beispielen zum Finden von widgets zur Laufzeit eines Widget Tests ^{XII}
 - Ein Rezept zum Testen von Nutzerverhalten wie dem Tab, dem Drag und dem eingeben von Text ^{XIII}
- 3 Rezepte für Integrationstests
 - Eine grundlegende Anleitung zum Erstellen eines Integrationstest ^{XIV}
 - eine Anleitung zum simulieren von scrollen in der Anwendung während der Laufzeit eines Integrationstest ^{XV}
 - eine Anleitung zum Performance Profiling ^{XVI}

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.

^X<https://flutter.dev/docs/cookbook/testing/unit/mocking>
^{XI}<https://flutter.dev/docs/cookbook/testing/widget/introduction>
^{XII}<https://flutter.dev/docs/cookbook/testing/widget/finding>
^{XIII}<https://flutter.dev/docs/cookbook/testing/widget/tap-drag>
^{XIV}<https://flutter.dev/docs/cookbook/testing/integration/introduction>
^{XV}<https://flutter.dev/docs/cookbook/testing/integration/scrolling>
^{XVI}<https://flutter.dev/docs/cookbook/testing/integration/profiling>

```
19 final OnSelect<ChoiceType> onSelect;
20 final OnDeselect<ChoiceType> onDeselect;
21 final String? errorText;
22
23 SelectionCard(
24   {required this.title,
25     required Iterable<ChoiceType> initialValue,
26     required this.allChoices,
27     required this.onSelect,
28     required this.onDeselect,
29     this.errorText,
30     Key? key})
```

Code Snippet 1: errorText wird der SelectionCard hinzugefügt, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart

```
63 onTap: navigateToSelectionScreen,
64 ),
65 if (errorText != null)
66   Padding(
67     padding: const EdgeInsets.all(8.0),
68     child: Text(errorText!,
69       style:
70         const TextStyle(fontSize: 12.0, color: Colors.red)),
71   )
```

Code Snippet 2: errorText wird ausgegeben, Quelle: Eigenes Listing,
Datei: Quellcode/Schritt-3/conditional_form/lib/widgets/selection_card.dart

```
1 class MassnahmenFormViewModel {
2     final letzterStatus = BehaviorSubject<LetzterStatus?>.seeded(null);
3     // ...
4 }
```

Code Snippet 3: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

```
1 class ChoiceChangeNotifier extends ChangeNotifier {
2     BuiltSet<Choice> _choices = BuiltSet<Choice>();
3
4     BuiltSet<Choice> get choices => _choices;
5
6     set choices(BuiltSet<Choice> choices) {
7         _choices = choices;
8         notifyListeners();
9     }
10 }
11 class LetzterStatusViewModel extends ChoiceChangeNotifier {}
```

Code Snippet 4: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

```
1 providers: [
2     Provider<MassnahmenFormViewModel>(
3         create: (_) => MassnahmenFormViewModel(),
4     Provider<MassnahmenJsonFile>(create: (_) => MassnahmenJsonFile()),
5     Provider(
6         create: (context) => MassnahmenPool(
7             Provider.of<MassnahmenJsonFile>(context, listen: false))),
8     ChangeNotifierProvider(create: (context) => LetzterStatusViewModel()))
9 ],
```

Code Snippet 5: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio, Quelle: JetBrains Marketplace Built Value Snippets Plugin

```
1 Consumer<ConsumerType>(  
2     builder: (context, choiceChangeNotifier, child) {  
3         final selectedChoices = choiceChangeNotifier.choices;  
4         return Card(  
5             child: Column(  
6                 crossAxisAlignment: CrossAxisAlignment.start,  
7                 children: [  
8                     ListTile(  
9                         focusNode: focusNode,  
10                        title: Text(title),  
11                        subtitle:  
12                        Text(selectedChoices.map((c) => c.description).join(", ")),  
13                        trailing: const Icon(Icons.edit),  
14                        onTap: navigateToSelectionScreen,  
15                    )  
16                ],  
17            ),  
18        );  
19    },  
20 )
```

Code Snippet 6: Live Template für die Erstellung von built_value Boilerplate-Code in Android Studio,
Quelle: JetBrains Marketplace Built Value Snippets Plugin

Anhang

A Technologiewahl Anhang

A.1 Stimmen verwendeter Frameworks

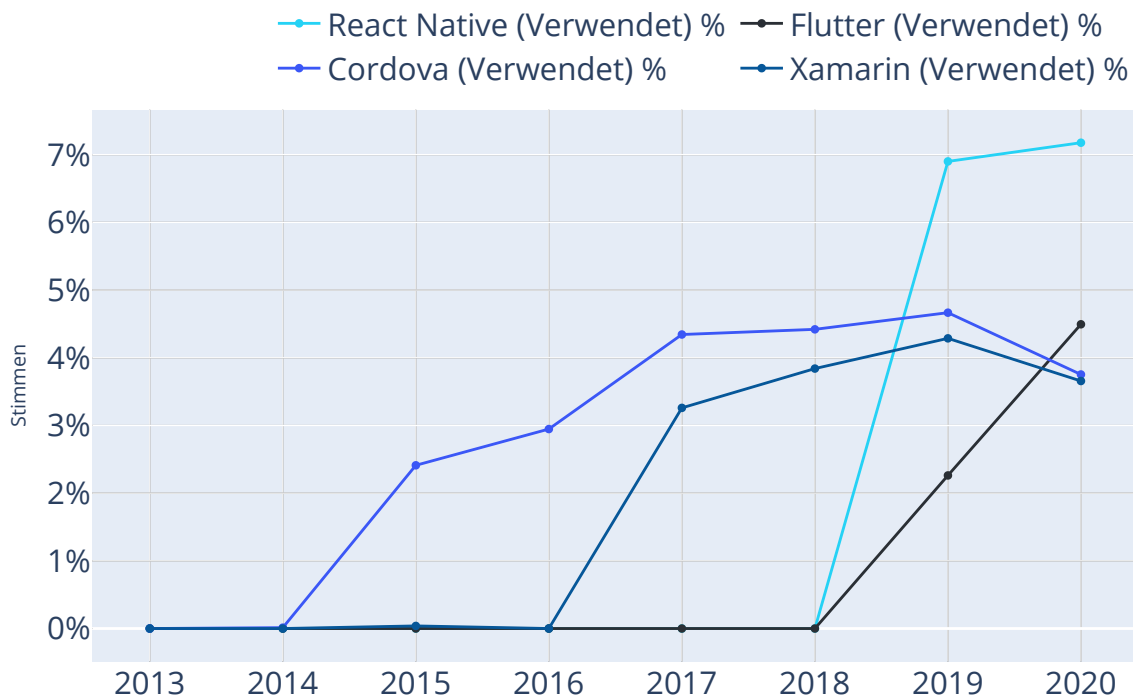


Abbildung 7: Stimmen verwendeter Frameworks, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

A.2 Stimmen gewünschter Frameworks

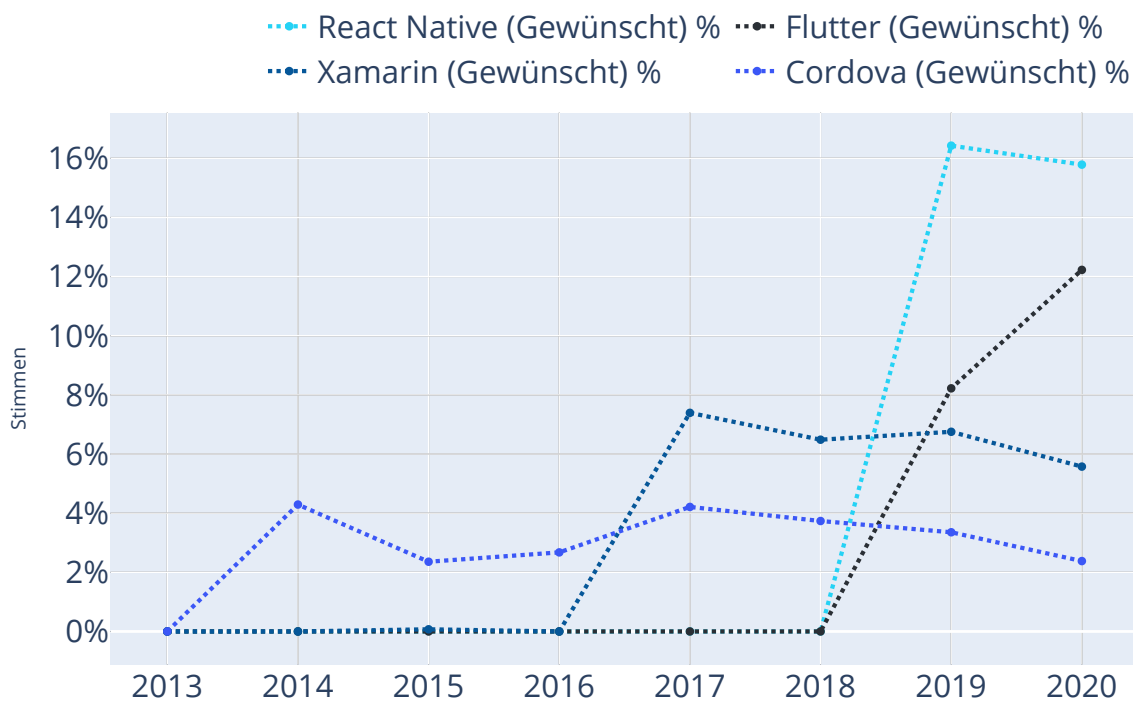


Abbildung 8: Stimmen gewünschter Frameworks, Quelle: Eigene Abbildung, Notebook: Charts/Stack Overflow Umfrage/Stack Overflow Umfrage.ipynb

B Vergleich React Native und Flutter Anhang

```

1 import 'package:flutter/material.dart';
2 import 'validation.dart';
3
4 void main() => runApp(MyApp());
5
6 class MyApp extends StatelessWidget {
7   @override
8   Widget build(BuildContext context) => MaterialApp(
9     home: Scaffold(
10       body: MyCustomForm(),
11     ),
12   );
13 }
14
15 class MyCustomForm extends StatefulWidget {
16   @override
17   MyCustomFormState createState() => MyCustomFormState();
18 }
19
20 class MyCustomFormState extends State<MyCustomForm> {
21   final _formKey = GlobalKey<FormState>();
22
23   @override
24   Widget build(BuildContext context) => Form(
25     key: _formKey,
26     child: Padding(
27       padding: const EdgeInsets.all(8.0),
28       child: Column(
29         children: [
30           TextFormField(
31             decoration: const InputDecoration(labelText: "Name"),
32             validator: (String? value) =>
33               validateNotEmpty("Name", value)),
34           TextFormField(
35             decoration: const InputDecoration(labelText: "Email"),
36             validator: (String? value) => validateEmail("Name", value)),
37           TextFormField(
38             decoration: const InputDecoration(labelText: "Password"),
39             validator: (String? value) =>
40               validateNotEmpty("Password", value)),
41           Padding(
42             padding: const EdgeInsets.symmetric(vertical: 16.0),
43             child: ElevatedButton(
44               onPressed: () {
45                 if (_formKey.currentState!.validate()) {
46                   ScaffoldMessenger.of(context).showSnackBar(
47                     SnackBar(content: Text('Processing Data')));
48                 }
49               },
50             child: Text('Submit'),
51           ),
52         ],
53       ),
54     ),
55   );
56 }
57 }

```

Code Snippet 7: , Quelle: Eigenes Listing,
 Datei: Quellcode/Vergleich/form-in-flutter/lib/main.dart

```

1 final emailPattern = RegExp(
2   r'^(([^<>()\[\]\\\.,;:\s@"]+(\.[^<>()\[\]\\\.,;:\s@"]+)*|(".*"))@((\[[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,}))$')';
3
4 String? validateEmail(String label, String? value) {
5   if (value == null || value.isEmpty) {
6     return '$label is required';
7   } else if (!emailPattern.hasMatch(value)) {
8     return 'Invalid Email Format';
9   } else {
10    return null;
11  }
12 }
13
14 String? validateNotEmpty(String label, String? value) {
15   if (value == null || value.isEmpty) {
16     return '$label is required';
17   } else {
18     return null;
19   }
20 }

```

Code Snippet 8: , Quelle: Eigenes Listing,

Datei: Quellcode/Vergleich/form-in-flutter/lib/validation.dart

```

1 import * as React from 'react';
2 import {
3   Text,
4   View,
5   StyleSheet,
6   Button,
7   Alert,
8   ScrollView,
9 } from 'react-native';
10 import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-view';
11
12 import Constants from 'expo-constants';
13 import { useForm } from 'react-hook-form';
14
15 // You can import from local files
16 import Input from './components/Input';
17 import Form from './components/Form';
18 import validation from './validation';
19 import Hero from './Hero';
20
21 type FormData = {
22   name: string;
23   email: string;
24   password: string;
25 };
26
27 export default () => {
28   const { handleSubmit, register, setValue, errors } = useForm<FormData>();
29
30   const onSubmit = (data: FormData) => {
31     Alert.alert('data', JSON.stringify(data));
32   };
33
34   return (
35     <KeyboardAwareScrollView
36       contentContainerStyle={styles.container}
37       style={{ backgroundColor: '#181e34' }}>
38       <Hero />
39       <View style={styles.formContainer}>
40         <Form {...{ register, setValue, validation, errors }}>
41           <Input name="name" label="Name " />
42           <Input name="email" label="Email" />
43           <Input name="password" label="Password" secureTextEntry={true} />
44           <Button title="Submit" onPress={handleSubmit(onSubmit)} />
45         </Form>
46       </View>
47     </KeyboardAwareScrollView>
48   );
49 };
50
51 const styles = StyleSheet.create({
52   container: {
53     flex: 1,
54     justifyContent: 'center',
55     paddingTop: Constants.statusBarHeight,
56     backgroundColor: '#181e34',
57   },
58   formContainer: {
59     padding: 8,
60     flex: 1,
61   },
62   button: {
63     backgroundColor: 'red',
64   },
65 });

```

Code Snippet 9: , Quelle: XXXXX todo HIGH PRIO: FÜLLEN XXXXX,
 Datei: Quellcode/Vergleich/form-in-react-native-the-right-way/App.tsx

```

1 import * as React from 'react';
2 import { Text, View, StyleSheet, Image } from 'react-native';
3
4 export default () => {
5   return (
6     <View style={styles.container}>
7       <Image style={styles.logo} source={require('./assets/hero.jpg')} />
8       <Text style={styles.paragraph}>
9         Form in React Native, The right Way!
10      </Text>
11    </View>
12  );
13 }
14
15 const styles = StyleSheet.create({
16   container: {
17     justifyContent: 'center',
18     flex:1,
19   },
20   paragraph: {
21     margin: 24,
22     marginTop: 0,
23     fontSize: 34,
24     fontWeight: 'bold',
25     textAlign: 'center',
26     color: '#FFF'
27   },
28   logo: {
29     width: '100%',
30     height: 200
31   }
32 });

```

Code Snippet 10: , Quelle: XXXXX todo HIGH PRIO: FÜLLEN XXXXX,
 Datei: Quellcode/Vergleich/form-in-react-native-the-right-way/Hero.tsx

```

1 export default {
2   name: {required: {value: true, message: 'Name is required'}},
3   email: {
4     required: {value: true, message: 'Email is required'},
5     pattern: {
6       value: /^[^<>()\[\]\\\.,;:\s@"]+(\.[^<>()\[\]\\\.,;:\s@"]+)*|(".*")@[_]
7         ↪  ((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\)|((([a-zA-Z\-0-9]+\._)
8         ↪  )+[a-zA-Z]{2,}))$/,
9       message: 'Invalid Email Format',
10    },
11  },
12  password: {
13    required: {value: true, message: 'Password is required'},
14  },
15 };

```

Code Snippet 11: , Quelle: XXXXX todo HIGH PRIO: FÜLLEN XXXXX,
 Datei: Quellcode/Vergleich/form-in-react-native-the-right-way/validation.tsx

```

1 import * as React from 'react';
2 import { TextInput, KeyboardAvoidingView, findNodeHandle } from 'react-native';
3 import { ValidationOptions, FieldError } from 'react-hook-form';
4
5 interface ValidationMap {
6   [key: string]: ValidationOptions;
7 }
8
9 interface ErrorMap {
10  [key: string]: FieldError | undefined;
11 }
12
13 interface Props {
14   children: JSX.Element | JSX.Element[];
15   register: (
16     field: { name: string },
17     validation?: ValidationOptions
18   ) => void;
19   errors: ErrorMap;
20   validation: ValidationMap;
21   setValue: (name: string, value: string, validate?: boolean) => void;
22 }
23
24 export default ({
25   register,
26   errors,
27   setValue,
28   validation,
29   children,
30 }: Props) => {
31   const Inputs = React.useRef<Array<TextInput>>([]);
32
33   React.useEffect(() => {
34     (Array.isArray(children) ? [...children] : [children]).forEach((child) => {
35       if (child.props.name)
36         register({ name: child.props.name }, validation[child.props.name]);
37     });
38   }, [register]);
39
40   return (
41     <>
42       {(Array.isArray(children) ? [...children] : [children]).map(
43         (child, i) => {
44           return child.props.name
45             ? React.createElement(child.type, {
46               ...{
47                 ...child.props,
48                 ref: (e: TextInput) => {
49                   Inputs.current[i] = e;
50                 },
51                 onChangeText: (v: string) =>
52                   setValue(child.props.name, v, true),
53                 onSubmitEditing: () => {
54                   Inputs.current[i + 1]
55                     ? Inputs.current[i + 1].focus()
56                     : Inputs.current[i].blur();
57               },
58               //onBlur: () => triggerValidation(child.props.name),
59               blurOnSubmit: false,
60               //name: child.props.name,
61               error: errors[child.props.name],
62             },
63             )
64             : child;
65         }
66       )}
67     </>
68   );
69 };

```

Code Snippet 12: , Quelle: XXXXX todo HIGH PRIO: FÜLLEN XXXXX,
Datei: Quellcode/Vergleich/form-in-react-native-the-right-way/components/Form.tsx

```

1 import * as React from 'react';
2 import {
3   View,
4   TextInput,
5   Text,
6   StyleSheet,
7   ViewStyle,
8   TextStyle,
9   TextInputProps,
10 } from 'react-native';
11 import { FieldError } from 'react-hook-form';
12 interface Props extends TextInputProps {
13   name: string;
14   label?: string;
15   labelStyle?: TextStyle;
16   error?: FieldError | undefined;
17 }
18
19 export default React.forwardRef<any, Props>(  

20   (props, ref): React.ReactElement => {  

21     const { label, labelStyle, error, ...inputProps } = props;  

22
23     return (  

24       <View style={styles.container}>  

25         {label && <Text style={[styles.label, labelStyle]}>{label}</Text>}  

26         <TextInput  

27           autoCapitalize="none"  

28           ref={ref}  

29           style={[styles.input, { borderColor: error ? '#fc6d47' : '#c0cbd3' }]}  

30           {...inputProps}  

31         />  

32         <Text style={styles.textError}>{error && error.message}</Text>  

33       </View>  

34     );  

35   }  

36 );  

37
38 const styles = StyleSheet.create({  

39   container: {  

40     marginVertical: 8,  

41   },  

42   input: {  

43     borderStyle: 'solid',  

44     borderWidth: 1,  

45     borderRadius: 5,  

46     paddingVertical: 5,  

47     paddingLeft: 5,  

48     fontSize: 16,  

49     height: 40,  

50     color: '#c0cbd3',  

51   },  

52   label: {  

53     paddingVertical: 5,  

54     fontSize: 16,  

55     fontWeight: 'bold',  

56     color: '#c0cbd3',  

57   },  

58   textError: {  

59     color: '#fc6d47',  

60     fontSize: 14,  

61   },  

62 });

```

Code Snippet 13: , Quelle: XXXXX todo HIGH PRIO: FÜLLEN XXXXX,
Datei: Quellcode/Vergleich/form-in-react-native-the-right-way/components/Input.tsx

Literatur

- <Formik /> | *Formik Docs API*. Juni 2021. URL: https://web.archive.org/web/20210409184616if_/https://formik.org/docs/api/formik (Zitiert auf der Seite 13).
- Adobe Inc. *FAQ | PhoneGap Docs*. Aug. 2016. URL: <https://web.archive.org/web/20200806024626/http://docs.phonegap.com/phonegap-build/faq/> (Zitiert auf der Seite 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).
- Borenkraout, Matan. *Native Testing Library Introduction | Testing Library Docs*. Nov. 2020. URL: <https://web.archive.org/web/20210128142719/https://testing-library.com/docs/react-native-testing-library/intro/> (Zitiert auf der Seite 14).
- Does redux-form work with React Native?* Juni 2021. URL: <https://web.archive.org/web/20210602234346/https://redux-form.com/7.3.0/docs/faq/reactnative.md/> (Zitiert auf der Seite 12).
- Facebook Inc. *The React Native Ecosystem*. Juni 2021. URL: <https://web.archive.org/web/20210602191504/https://github.com/facebook/react-native/blob/d48f7ba748a905818e8c64fe70fe5b24aa098b05/ECOSYSTEM.md> (Zitiert auf der Seite 14).
- Google LLC. *Build a form with validation*. Juni 2021. URL: <https://web.archive.org/web/20210122020924/https://flutter.dev/docs/cookbook/forms/validation> (Zitiert auf der Seite 13).
- *Forms | Flutter Docs Cookbook*. Juni 2021. URL: <https://web.archive.org/web/20201102003629/https://flutter.dev/docs/cookbook/forms> (Zitiert auf der Seite 13).
 - *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).
- React Native | Formik Docs*. Juni 2021. URL: https://web.archive.org/web/20210507005917if_/https://formik.org/docs/guides/react-native (Zitiert auf der Seite 12).
- React Native | React Hook Form - Get Started*. Juni 2021. URL: https://web.archive.org/web/20210523042601if_/https://react-hook-form.com/get-started/ (Zitiert auf der Seite 12).
- reduxForm | Redux Form - API*. Juni 2021. URL: <https://web.archive.org/web/20210506221401/https://redux-form.com/7.4.2/docs/api/reduxform.md/#-code-validate-values-object-props-object-gt-errors-object-code-optional-> (Zitiert auf der Seite 13).
- register | React Hook Form - API*. Juni 2021. URL: <https://web.archive.org/web/20210406032209/https://react-hook-form.com/api/useform/register> (Zitiert auf der Seite 13).
- 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 8).
- Stack Exchange, Inc. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research*. Mai 2021. URL: <https://insights.stackoverflow.com/survey/> (Zitiert auf den Seiten 8, 9).

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig angefertigt und mich fremder Hilfe nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichtem oder nicht veröffentlichtem Schrifttum entnommen sind, habe ich als solche kenntlich gemacht.

Wernigerode, den 16.11.2020

Alexander Johr