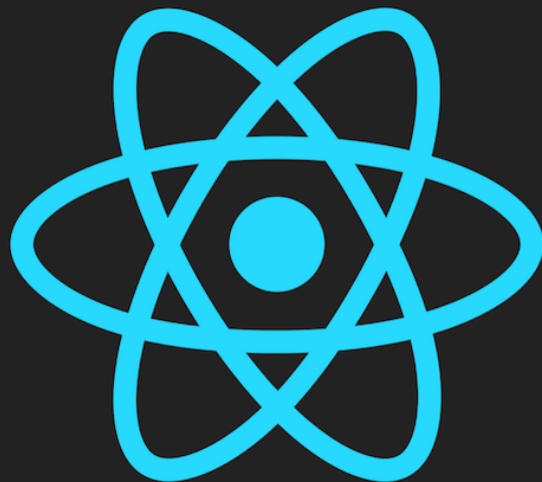


2020

Felchen – Statistik Applikation



React Native

Patrick Tomasi

Modul: 335 -Mobile Applikation realisieren

1 Inhaltsverzeichnis

2	Einleitung.....	3
3	Angaben zum Dokument.....	3
4	Projektauftrag.....	4
4.1	Detail zum Projektauftrag.....	5
4.1.1	Anforderungen und Planung.....	5
4.1.2	Lösungskonzept erarbeiten.....	5
4.1.3	Mobile App planen.....	5
4.1.4	Mobile App programmieren.....	6
4.1.5	Mobile App publizieren.....	6
4.1.6	Mobile App gemäss Testplan überprüfen.....	6
5	Informieren.....	6
5.1	Idee zur App.....	7
6	Planung.....	8
6.1	Use Case Diagramm.....	9
6.2	Zustands Diagramm.....	10
6.3	Mockup.....	11
6.3.1	Home Screen.....	11
6.3.2	Detail Screen.....	12
6.3.3	Fang hinzufügen Screen.....	13
6.3.4	Wetter Screen.....	14
6.3.5	Einstellungen Screen.....	15
7	Entscheiden.....	16
7.1	Ist die Entscheidung Sinnvoll?.....	16
8	Realisieren.....	17
8.1	Vorgehensmodell.....	17
8.2	Navigation.....	18
8.3	Tab Einstellungen.....	19
8.4	Wetter Screen.....	20
8.5	Home Screen.....	21
8.6	Detail Screen.....	22
8.7	Neuer Eintrag.....	23
9	App veröffentlichen.....	24
9.1	Expo publish.....	24
9.2	Building Standalone Apps.....	24
9.3	Auf Google Play eine App hochladen.....	26

9.3.1	APK.....	26
9.3.2	Store-Eintrag.....	27
9.4	Eigene App mit Google Play Console veröffentlichen	29
9.5	Eigene App im Google Play Store	30
10	Tests Cases.....	31
10.1	Test Case Home Screen	31
10.2	Test Case Wetter Screen.....	32
10.3	Test Case Einstellungen Screen.....	32
11	Reflexion.....	33
12	Abbildverzeichnis	34
13	Tabellenverzeichnis.....	34

2 Einleitung

Dieses Dokument gilt der Dokumentation der Felchen – Statistik -App welche als Projekt für das Modul 335: Mobile Applikationen realisieren entwickelt wurde.

Die App wurde mit React Native entwickelt. Es wurde Expo verwendet und in Visual Code programmiert.

3 Angaben zum Dokument

Name des Dokuments: Dokumentation Felchen-Statistik-App

Art des Dokumentes: Dokumentation

Aktuelle Version: 0.8

Kontaktperson: Tomasi Patrick
Postgasse 10a
5079 Zeihen
p.tomasi@hotmail.ch

Auftraggeber: Bénédict Schule AG
Militärstrasse 106
8004 Zürich

4 Projektauftrag

Die nachfolgende Abbildung zeigt den originalen erhaltenen Projektauftrag. Darin ist die Beschreibung des Auftrages.

Informatik
Modul 335 – Kompetenznachweis

Kompetenznachweis 335: App mit 2 Elementen planen, entwickeln und publizieren



Ausgangslage:

Sie erstellen eine Mobile-App mit den erarbeiteten Kompetenzen aus dem Modul, alle Kompetenzen müssen hier abgelegt werden (siehe Kompetenzraster).

Zeitbedarf:	16 Lektionen
Hilfsmittel:	npm tools/ Frameworks
Methode/Sozialform:	EA/PA
Handlungsziele:	✓ Siehe Kompetenzraster

Aufgabe 1: Mobile App – Anforderungen und Planung

- a) Erstellen Sie die Planung einer mobile App nach einer eigenen Idee, dazu erstellen sie geeignete Screen Skizzen ihrer Idee auf Papier in einer Art Storyboard in welchem Sie den Ablauf der Screen-Folgen darstellen. Sie listen alle Funktionalitäten auf!
- b) Die Mobile –App muss min. 2 Elemente wie 2 Aktoren/Sensoren, 1 Aktor/Sensor und 1 persistente Storage (z.B. firebase – firestore), 1 Aktor/Sensor und Authentifizierung (z.B. firebase – Authentifizierung) oder 1 Aktor/Sensor und On – Offline Modus unterstützen!

Aufgabe 2: Mobile App – Lösungskonzept erarbeiten

Beschreiben Sie kurz wie sie die Mobile App entwickeln werden, mit welchem Framework und was für ein Typ der Mobilen App es sein wird (native App, Hybrid, WebApp).

Aufgabe 3: Mobile App – Mobile App planen

Planen Sie die App, entwickeln Sie die Screens welche Sie brauchen und die entsprechenden Anwendungsfälle, die Anwendungsfallbeschreibungen mit einem Aktivitätsdiagramm, diese Aufgabe wird für die Bewertung der Kompetenzen im Modul 326 verwendet!

Aufgabe 4: Mobile App – Mobile App programmieren

Erstellen Sie die App mit einem geeigneten Framework und Entwicklungsoberfläche!

Aufgabe 5: Mobile App – Mobile App publizieren

Führen Sie hier die nötigen Schritte durch um die App zu publizieren!
Das Produkt muss eine fertig paketierte Datei sein welche zum publizieren auf dem App-Store oder Google Play Store bereit ist! Beschreiben Sie die Schritte in der Dokumentation.

Aufgabe 6: Mobile App gemäss Testplan überprüfen

Führen Sie hier die nötigen Tests gemäss Testplan aus und halten Sie die Ergebnisse fest bezüglich funktionalen und nicht funktionalen Tests!

4.1 Detail zum Projektauftrag

Nachfolgend wird genauer auf die einzelnen Aufgaben des Projektes eingegangen.

4.1.1 Anforderungen und Planung

- Erstellen Sie die Planung einer mobilen App nach einer eigenen Idee, dazu erstellen sie geeignete Screen Skizzen ihrer Idee auf Papier in einer Art Storyboard in welchem Sie den Ablauf der Screen-Folgen darstellen. Sie listen alle Funktionalitäten auf!
- Die Mobile –App muss min. 2 Elemente wie 2 Aktoren/Sensoren, 1 Aktor/Sensor und 1 persistente Storage (z.B. firebase – firestore), 1 Aktor/Sensor und Authentifizierung (z.B. firebase – Authentifizierung) oder 1 Aktor/Sensor und On – Offline Modus unterstützen!
- Erstellen Sie hier einen Testplan in dem ersichtlich ist wie Sie die App am Ende testen werden!

4.1.2 Lösungskonzept erarbeiten

- Beschreiben Sie kurz wie sie die Mobile App entwickeln werden, mit welchem Framework und was für ein Typ der Mobilen App es sein wird (native App, Hybrid, WebApp).
- Beschreiben Sie kurz welche Schritte nötig sind, um diese App später zu publizieren!
- Beschreiben Sie welche Elemente und wie Sie die Aktoren/Sensoren, die persistente Speicherung, die Authentifizierung oder die offline Funktionalität benutzen werden!

4.1.3 Mobile App planen

- Planen Sie die App, entwickeln Sie die Screens, welche Sie brauchen und die entsprechenden Anwendungsfälle, die Anwendungsfallbeschreibungen mit einem Aktivitätsdiagramm, diese Aufgabe wird für die Bewertung der Kompetenzen im Modul 326 verwendet!

4.1.4 Mobile App programmieren

- Erstellen Sie die App mit einem geeigneten Framework und Entwicklungsoberfläche!

4.1.5 Mobile App publizieren

- Führen Sie hier die nötigen Schritte durch, um die App zu publizieren! Das Produkt muss eine fertig paketierte Datei sein, welche zum Publizieren auf dem App-Store oder Google Play Store bereit ist! Beschreiben Sie die Schritte in der Dokumentation.

4.1.6 Mobile App gemäss Testplan überprüfen

- Führen Sie hier die nötigen Tests gemäss Testplan aus und halten Sie die Ergebnisse fest bezüglich funktionalen und nicht funktionalen Tests!

5 Informieren

Viele Informationen zum Thema Mobile App entwickeln mit React Native bekamen wir in der Schule direkt im Unterricht.

Wir haben viele kleine Aufgaben selbst versucht und ausgetestet, wie z.B. eine Navigation einbauen, Listen wie FlatList oder SectionList zu verwenden.

Wir haben auch eine API angesteuert und die Testdaten direkt verarbeitet, und in einer App angezeigt.

Ich habe mich aber selbst noch sehr viel Informiert zum ganzen Thema mobile App entwickeln. Ich habe einige Dokumentation angeschaut wie auch unzählige Videos zum Thema angeschaut damit ich mich selbst noch mehr Informieren konnte wie ich eine App entwickeln kann.

Zudem habe ich auch zum Thema wie man eine App veröffentlicht einige Beiträge durchgelesen, vom Erstellen einer APK Datei bis hin zur Veröffentlichung einer App auf Google Play Store.

5.1 Idee zur App

Da ich ein leidenschaftlicher Angler bin wusste ich schon früh das mein Thema irgendwas mit Angeln zu tun haben wird.

Ich habe mich dann entschieden eine App zu entwickeln die man beim Hegenenfischen auf Felchen einsetzen kann. Die Fischerei auf Felchen ist eine sehr grosse Wissenschaft unter den Anglern. Die Fischerei wird mit einer Hegene ausgeübt.



Abbild einer Hegene mit verschiedenen Nymphen.

Wie man auf dem Abbild erkennen kann gibt es verschiedene Farben, die man verwenden kann und es sind dem Hersteller keine grenzen der Fantasie gesetzt welche Farben er verwendet. Wie er die Köpfchen gestaltet welche Hakengrösse usw. verwendet wird.

Zudem spielt auch die Tiefe des Standortes, die Temperatur und der Luftdruck eine Rolle bei der Fischerei auf die Felchen.

Da ich weiss, dass einige Fischer sich über diese Informationen Gedanken machen und sich diese auch notieren wie auch ich selbst, habe ich mich entschieden eine App zu entwickeln die diesen Vorgang vereinfachen und festhalten soll.

6 Planung

Für die Planung der App habe ich einen Fischerkollegen über meine Idee informiert. Da er selbständig Nymphen und Hegenen zum Vertrieb herstellt und ein Experte im Bereich Felchen Fischerei ist habe ich mich mit Ihm über meine Idee unterhalten und Ihn gebeten sich ein paar Gedanken zu machen welche Faktoren für eine Statistik Sinnvoll sind.

Ich habe von Ihm dann auch rasch ein gutes Feedback bekommen um mein Projekt zu Planen.

E-Mail von Andy, Geschäftsführer von hegene.ch

Hoi, also hab noch etwas «gehirnt»

Lenkbare Faktoren – must have

- Hakengrösse
- Nymphenfarbe
- Art der Nymphe (einfarbig, mehrfarbig, gerippt, Bodyglass)
- Köpfchen der Nymphe: (mit Perle, gebundener Kopf)

Lenkbare Faktoren (zusätzlich wählbar)

- Name einer Nymphe (falls von Hersteller)
- Art des Hakens (Felchenhaken, langer Haken)
- Hegenenlänge / Abstand
- Mit Nachläufer / ohne

Äussere Faktoren (Must)

- Gewässer
- Tiefe des Standortes
- Tiefe des Fischfanges
- Wassertemp
- Wetter bei Fang (Sonnig, leicht bewölkt, bewölkt, regen, Schneefall) → optional ev. Livedaten
- Lufttemperatur → optional ev. Livedaten

Äussere Faktoren (Zusätzlich wählbar)

- Wind → optional ev. Livedaten
- Luftdruck
- Höhe ü. M.
- Wellengang

Gruss Andy

6.1 Use Case Diagramm

Um die Ansprüche an die App aus Sicht des Benutzers darzustellen, wurde ein Use Case Diagramm erstellt, welches die Anforderung an die App beschreibt.

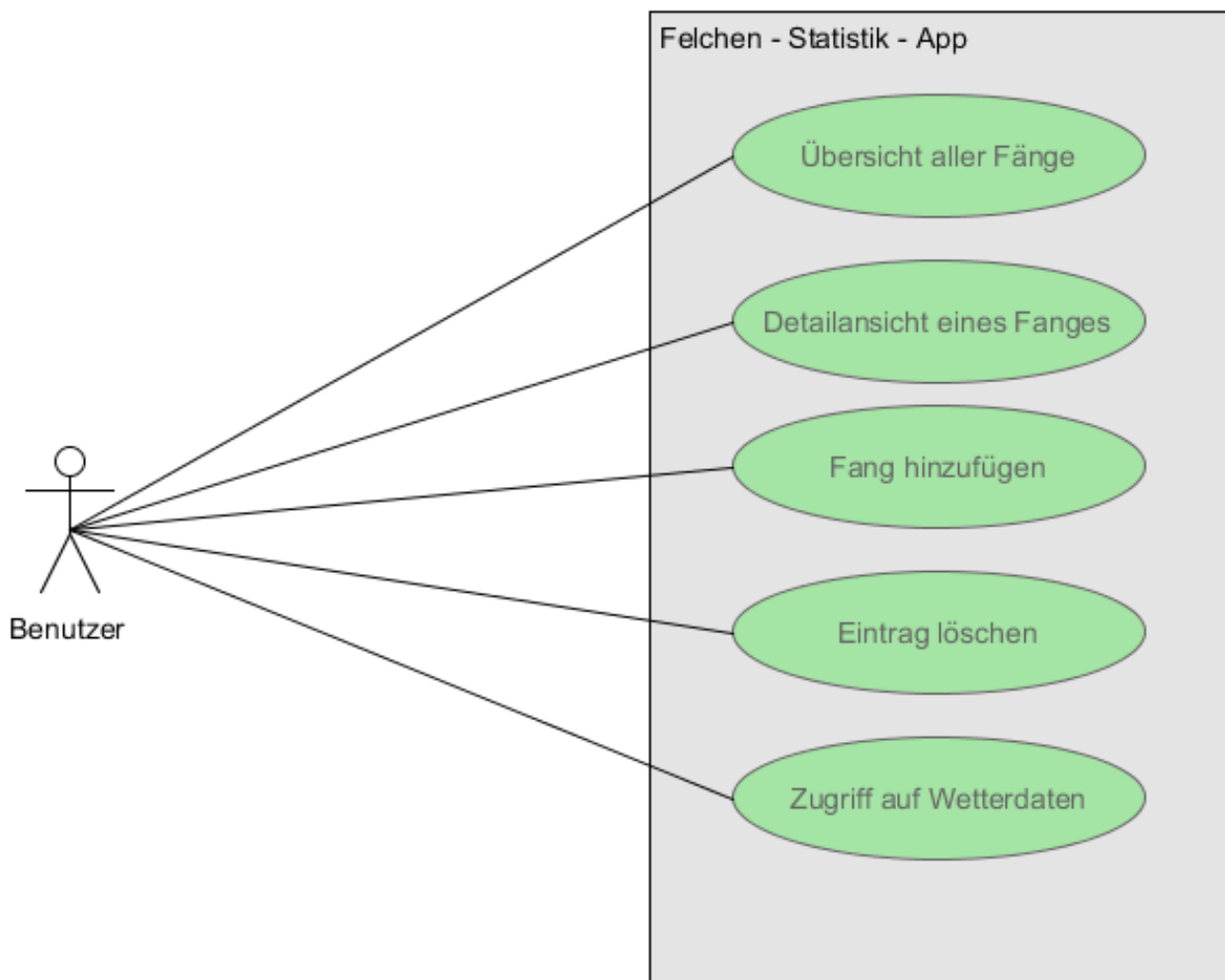


Abbildung 1: Use-Case Diagramm

6.2 Zustands Diagramm

Home Tab:

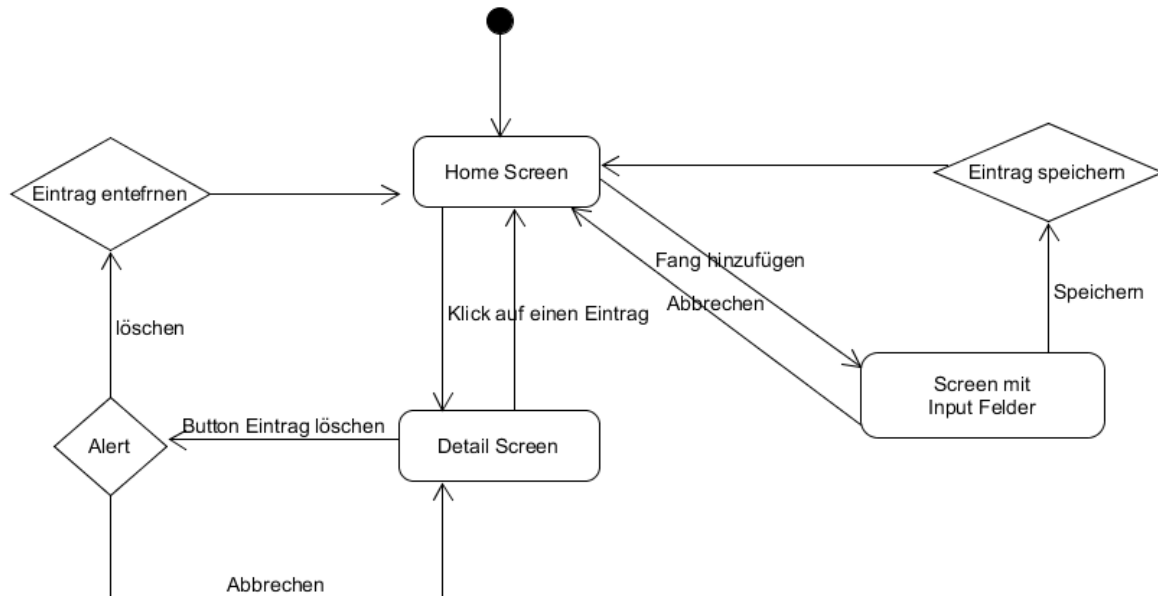


Abbildung 2: Zustandsdiagramm Home Tab

Wetter- und Einstellungen Tab:

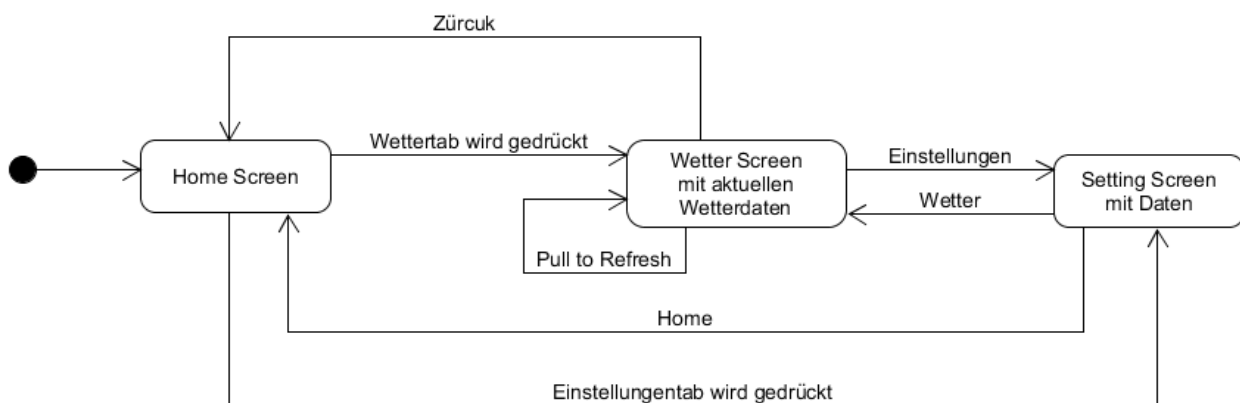


Abbildung 3: Zustandsdiagramm Wetter- und Einstellung Tab

6.3 Mockup

6.3.1 Home Screen

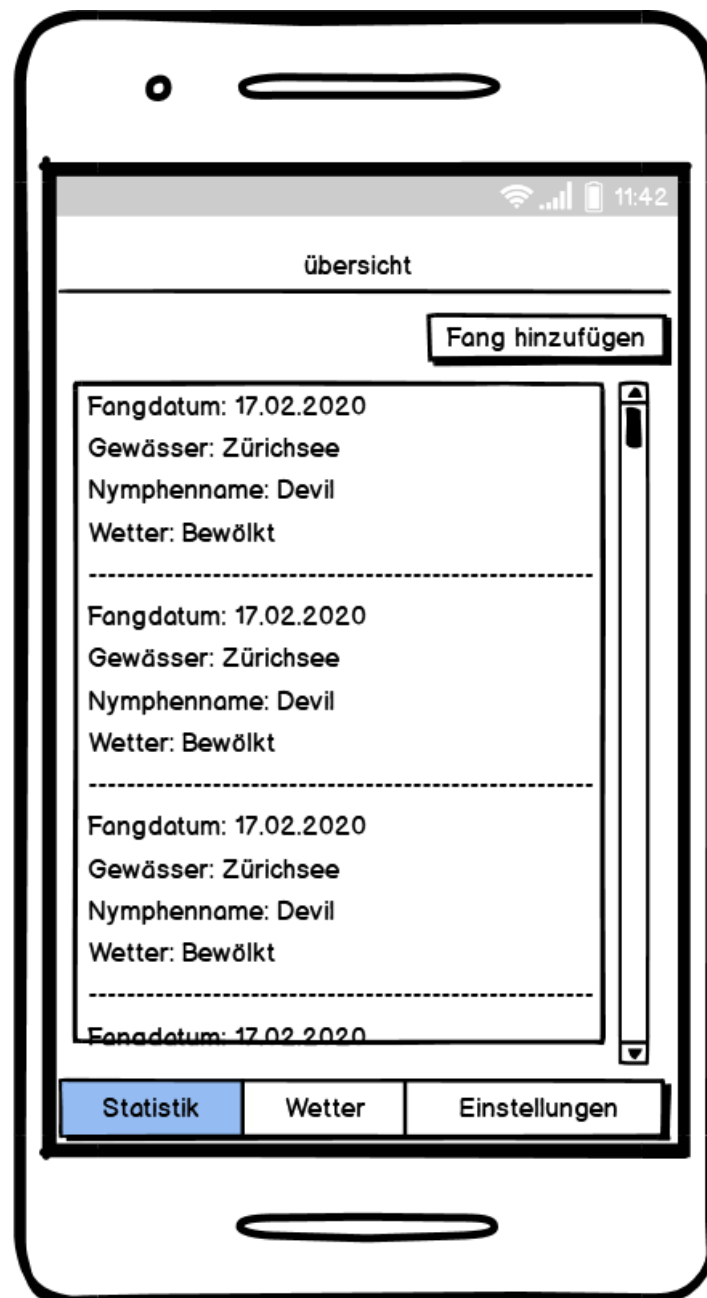


Abbildung 4: Mockup Home Screen

6.3.2 Detail Screen

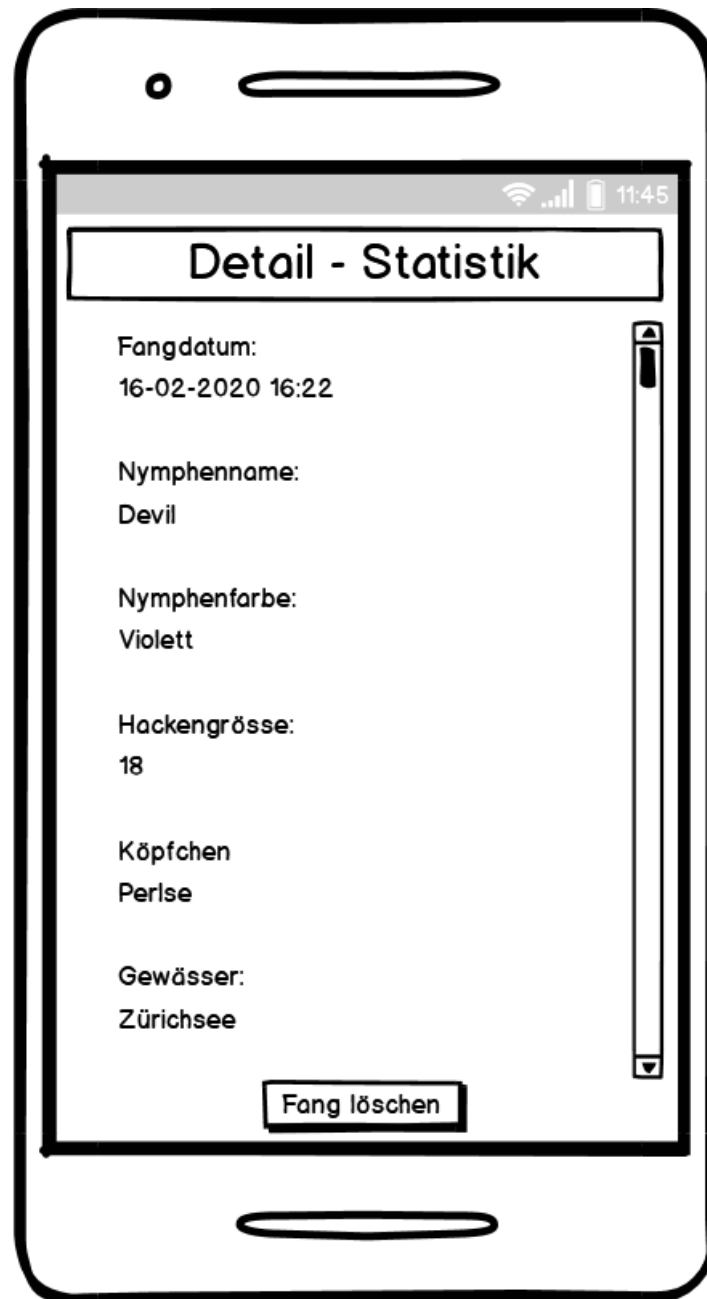


Abbildung 5: Mockup Detail Screen

6.3.3 Fang hinzufügen Screen

Mockup of the 'Fang hinzufügen' (Add Catch) screen on a mobile phone. The screen displays a form with input fields for the following fields:

- Name der Nympe:
- Nymphenfarbe
- Hakengrösse
- Köpfchen der Nympe
- Gewässername
- Tiefe des Standorts
- Tiefe des Fischfangs

Below the input fields are two buttons:

- Speichern
- Live-Daten laden

Abbildung 6: Mockup Fang hinzufügen Screen

6.3.4 Wetter Screen



Abbildung 7: Mockup Wetter Screen

6.3.5 Einstellungen Screen

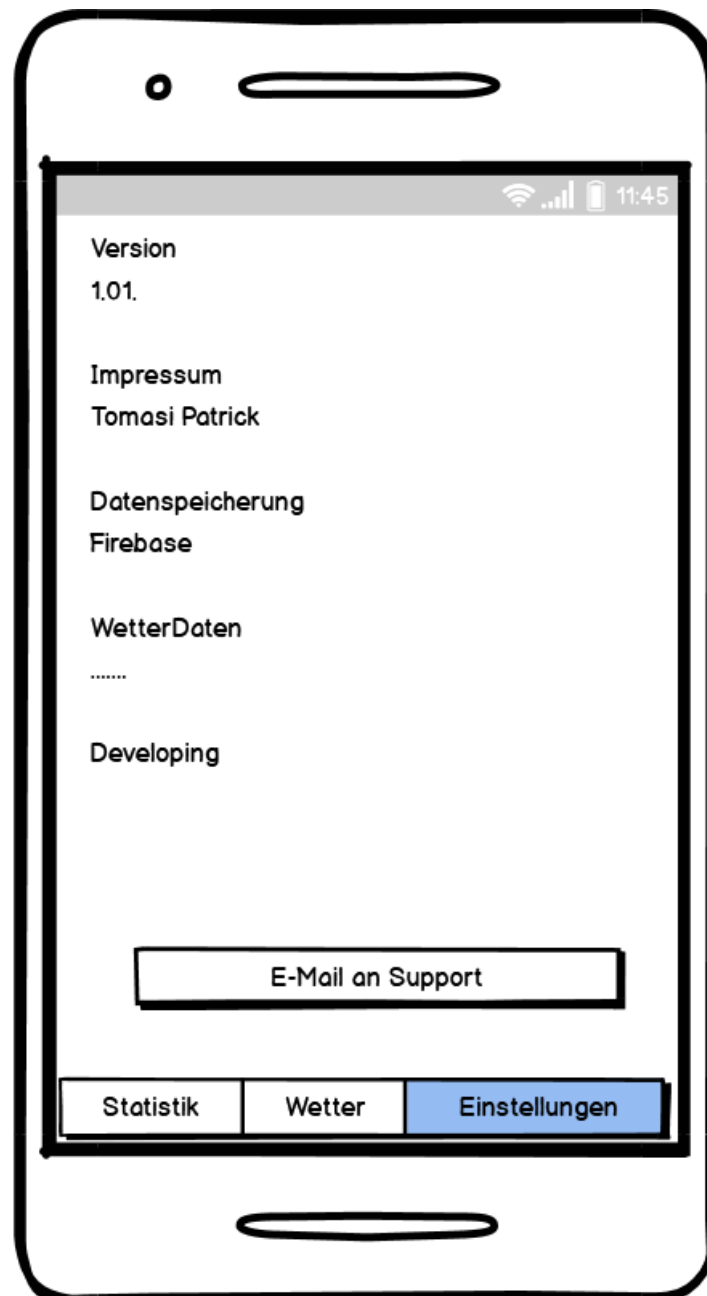


Abbildung 8: Mockup Einstellungen Screen

7 Entscheiden

Nach dem ich die Mockups erstellt habe und das Use Case Diagramm möchte ich die App genau so entwickeln wie in der Planung vorgesehen ist.

Ich finde den Ansatz gut den ich gewählt habe.

Die App soll eine Tab Navigation erhalten, um zwischen 3 Screens zu wechseln.

Der Home Screen soll alle bisher eingetragenen Fänge kurz mit den nötigsten Angaben anzeigen. Mit einem Klick auf einen Fang soll zu einem Detail Screen gewechselt werden, welcher eine Detaillierte Ansicht des Fanges ausgibt.

Über das Wetter Tab soll dem Benutzer das aktuelle Wetter seines Standorts mit verschieden Daten angezeigt werden.

Das Einstellungen Tab soll nur eine kleine Übersicht über die Version und den Entwickler anzeigen. Der Gedanke dahinter ist das dieser Screen eventuell in einem späteren Zeitpunkt erweitert werden kann mit z.B. Userdaten oder einem Login.

7.1 Ist die Entscheidung Sinnvoll?

Die Entscheidung ist sinnvoll, weil ich so schon geübte Aufgaben aus dem Unterricht wie eine FlatList anzeigen, eine Navigation einbinden und eine API abfragen in das Projekt einbinden kann.

Ich werde für die Abfragen der Wetterdaten die GPS-Position des Smartphones nützen und die Daten in der Firebase Cloud speichern.

Mit diesem Entscheid kann ich die gelernten Aufgaben des Unterrichts vertiefen, neue Elemente wie die Speicherung bei Firebase lernen und erfülle so die Anforderungen an die Applikation das diese aus mindestens einem Sensor und einer Datenspeicherung bestehen muss.

8 Realisieren

8.1 Vorgehensmodell

Die Realisierung dieses Projekt wird mit React Native durchgeführt. Es wird Expo eingesetzt und als Entwicklungsumgebung Visual Code.

Das ganze Projekt wird in einem GitHub Repository gespeichert, um jeder Zeit eine aktuelle Version des Projektes zu haben.

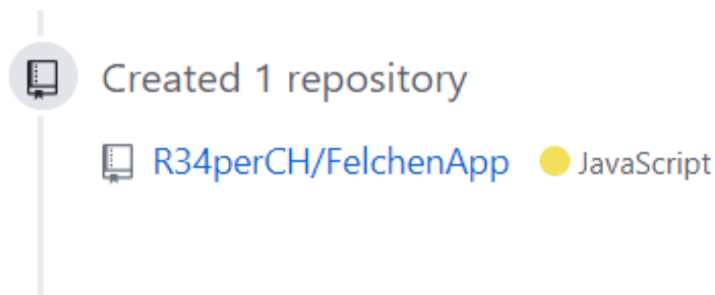


Abbildung 9: GitHub Repository

npm Version: 13.4
Expo Version: 3.11.9
NodeJS Version: 12.4.0
Visual Studio Code: 1.42.1

8.2 Navigation

Als Hauptnavigator wird ein Bottom Tab Navigator eingesetzt welcher die 3 Screens: Home, Wetter und Einstellungen verwaltet.

Da über den Home Screen zusätzlich der Detail Screen aufgerufen werden kann wird der Home Screen in ein Stack Navigator gepackt.

Der Bottom Tab Navigator ruft somit beim Home den Home Stack Navigator auf.

```
<Stack.Navigator>
  <Stack.Screen name="Home"
    component={HomeScreen}
    options={{
      headerTitle: 'Übersicht',
      headerTitleAlign: 'center',
      headerTintColor: 'darkorange',
      headerStyle: { backgroundColor: 'aliceblue' },
    }} />
  <Stack.Screen name="Detail"
    component={DetailScreen}
    options={{
      headerTitle: 'Detail-Statistik',
      headerTitleAlign: 'center',
      headerTintColor: 'darkorange',
      headerStyle: { backgroundColor: 'aliceblue' },
    }} />
</Stack.Navigator>
```

```
<Tab.Screen
  name="Statistik"
  component={HomeStack}
  options={{
    tabBarIcon: () => (
      <MaterialCommunityIcons name="home" color={'darkorange'} size={30} />
    ),
  }}
/>
```

8.3 Tab Einstellungen

Der Einstellung Screen wurde eher für die Zukunft entwickelt um in einem späteren Zeitpunkt z.B. ein User Login zu erstellen oder auch das der User gewisse Einstellungen in der App ändern kann.

Momentan gibt der Screen eine Section List aus mit verschiedenen Informationen wie die aktuelle Version der App, den Entwickler oder welche Wetter API im Einsatz ist.

Im unteren Teil wurde ein Button implementiert über den eine E-Mail direkt an den Entwickler gesendet werden kann.

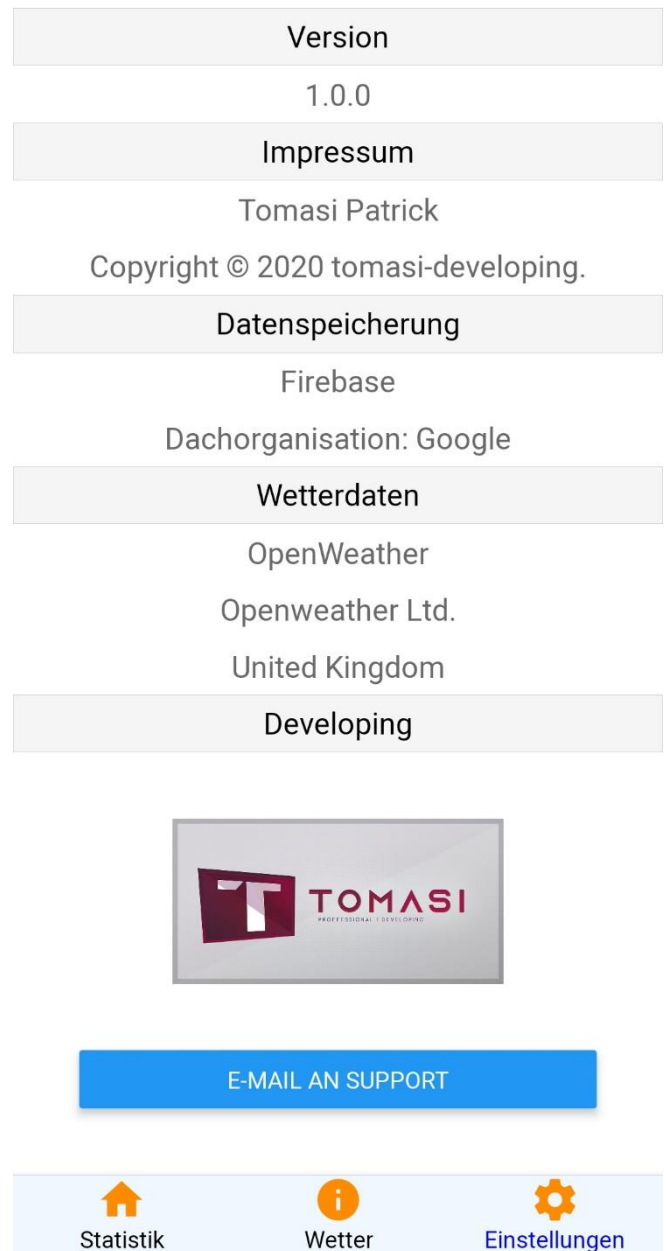


Abbildung 10: Einstellungen Screen

8.4 Wetter Screen

Wird der Wetter Screen aufgerufen wird im Hintergrund die GPS-Position des Benutzers ermittelt und Anhand der Position wird dann die Wetter API abgefragt.

Die Wetter API gibt ein JSON Objekt zurück, mit verschiedenen Daten die direkt verarbeitet werden können.

Da die Wetter Beschreibung auf English übermittelt werden, wurde eine eigene Funktion implementiert welche anhand des Wetter Codes eine bestimmte Beschriftung zurück gibt.

In diesem Fall wird der Code 803 von der Wetter API übergeben.

Für die Windrichtung wurde auch eine eigene Funktion implementiert die die Windrichtung zurück gibt, da die Windrichtung der Wetter API als Grad Zahl übergeben wird.

Die Wetter API überliefert auch ein Code um über einen Link das passende Icon abzurufen.



Abbildung 11: Wetter Screen

```
export const weatherConditions = {
  200: {
    title: 'Gewitter',
    subtitle: 'Gewitter mit leichtem Regen'
  },

```

```
const iconUrl = 'http://openweathermap.org/img/wn/' + icon + '@2x.png';
```

```
if (wind == 315) {
  return 'Nord-West'
}
if (wind > 0 && wind < 45) {
  return 'Nord-Nord-Ost'
}
```

8.5 Home Screen

Die App startet im Home Screen und die eingetragenen Fänge werden aus der Firebase Cloud geladen.

Die Daten aus der Cloud werden in einem Array abgespeichert und einer FlatList zur Darstellung übergeben. Sind keine Daten in der Cloud vorhanden, wird ein Text Tag angezeigt mit der Information dass keine Daten vorhanden sind. Ist die Internet Verbindung nicht vorhanden wird dies in einem Alert angezeigt. Die FlatList lässt sich über Pull to Refresh aktualisieren.

Die ganzen Listen Einträge werden mit einem TouchableOpacity umgeben damit die Liste angeklickt werden kann und in den Detail Screen Wechselt für eine Detaillierte Ansicht des Fanges.



Abbildung 12: Home Screen

```
catches.length <= 0 ? (
  <Text style={style.noData}>Keine Daten gespeichert</Text>
) : (
  <FlatList
    data={catches}
    keyExtractor={item => item.id}
    renderItem={({ item }) => (
      <HomeListItem fang={item} onPress={() =>
        this.props.navigation.navigate('Detail', { fang: item })}>
      </HomeListItem>
    )}
    onRefresh={() => this._refresh()}
    refreshing={this.state.isLoading}
    ItemSeparatorComponent={() => <View style={style.listSeparator} />}
  </FlatList>
```

8.6 Detail Screen

Wird im Home Screen ein bestimmter Fang angeklickt wird der Detail Screen aufgerufen und die Daten an den Screen übergeben. Im Detail Screen werden alle Daten aus der Cloud angezeigt die zu diesem Fang gespeichert wurden.

Damit alle Daten angesehen werden können wurde eine ScrollView verwendet.

Am Ende der Anzeige befindet sich noch ein Löschbutton mit dem der ausgewählte Fang aus der Datenbank entfernt werden kann. Wurde der Button betätigt wird ein Alert ausgegeben und nochmals nachgefragt ob der Eintrag gelöscht werden möchte.

Wird dieser mit löschen bestätigt wird der Eintrag aus dem aktuellen Array und der Datenbank entfernt und der Benutzer wird auf den Home Screen geleitet.



Abbildung 13: Detail Screen

```
<ScrollView
  style={style.ScrollView}
  contentContainerStyle={style.container}>
  <Text style={style.label}>Fangdatum:</Text>
  <Text style={style.daten}>{fang.fangDatum}</Text>

  _deleteButton(fang) {
    Alert.alert('Fang löschen ?', 'Der Fang wird aus der Datenbank entfernt !', [
      { text: 'Abbrechen', style: 'cancel' },
      { text: 'Löschen', style: 'destructive', onPress: () => this.props.navigation.n
avigate('Home', { toDelete: fang.id }) }
    ]);
  }
}
```

8.7 Neuer Eintrag

Der Screen NewCatch wurde als Modal implementiert und wird angezeigt sobald der Benutzer im Home Screen den Fang hinzufügen Button klickt. Wird das Modal aufgerufen werden die Wetterdaten erneut abgefragt und in die entsprechenden Inputfelder eingetragen. Auch für diesen Screen wurde eine ScrollView verwendet um alle Inputfelder anzeigen zu können. Am ende befinden sich noch 2 Buttons. Einen um die Live Daten neu zu laden und ein zweiter um den Fang in die Datenbank zu speichern.

Die Inputfelder mit den Live Daten lassen sich auch jederzeit vom Benutzer überschreiben wen er mit dem Eintrag nicht zufrieden ist.

Abbildung 14: Fang hinzufügen Screen

```
<Text style={styles.label}>Name der Nympe:</Text>
  <TextInput
    style={styles.input}
    placeholder='Name der Nympe'
    underlineColorAndroid='transparent'
    multiline={true}
    onChangeText={text => this.setState({ nymphenName: text })} />
```


9 App veröffentlichen

Da ich selbst ein Android Smartphone besitze werde ich mich bei der Veröffentlichung der App an die Android Version halten.

9.1 Expo publish

Mit dem Befehl `expo publish` im Terminal wird bei Expo eine Version hochgeladen, die mit der Expo App ausgeführt werden kann. Nach Erfolg erhält man ein Link, mit dem die App getestet werden kann. Der Link kann auch an andere weitergegeben werden, um die App zu testen. Mann braucht nur die Expo App aus dem Store zu installieren und den QR Code über den Link zu scannen. Dies funktioniert aber nur mit einem Android Smartphone.

Die Felchen App ist unter <https://exp.host/@tomasideveloping/FelchenStatistik> erreichbar.

9.2 Building Standalone Apps

Um eine App auf dem Smartphone zu installieren oder in den Google Play Store zu laden muss aus der App als erstes eine APK Datei erzeugt werden.

Expo nimmt uns diese Arbeit ab und erstellt eine APK Datei.

Als erstes muss die `App.js` angepasst werden:

```
"android": {
  "package": "ch.tomasi_developing.felchenapp",
  "versionCode": 1
},
"ios": {
  "supportsTablet": true,
  "bundleIdentifier": "com.yourcompany.yourappname",
  "buildNumber": "1.0.0"
}
```

Bei package kann die eigene Domain angegeben werden muss aber nicht sein.

Wurde die `App.js` angepasst kann im Terminal die APK erzeugt werden mit folgendem Befehl: `expo build:android -t apk`

Expo führt nun den Befehl aus und überprüft einige Einstellungen:

```
Checking if there is a build in progress...
Publishing to channel 'default'...
Building iOS bundle
Building Android bundle
Analyzing assets
Uploading assets
No assets changed, skipped.
Processing asset bundle patterns:
- D:\Schule\Modul_335_Mobile-Applikation realisieren\ProjektFelchenApp\FelchenStatistik\**\*
Uploading JavaScript bundles
```

Ist alles ok kommt man in eine Warteschleife dies kann gut bis zu 30 Minuten dauern.

In meinem Fall sogar 1 Stunde.

Den aktuellen Stand kann man in seinem Expo Account verfolgen.

Ist die APK erstellt bekommt man im Terminal eine Meldung und im Expo Account kann die APK heruntergeladen werden

PROJECT NAME	STATUS
@tomasideveloping/FelchenStatistik	● Finished
RELEASE CHANNEL	PLATFORM
default	Android
START TIME	DURATION
Tuesday, Feb 18 2020, 10:57 AM	an hour
BUILD ID	
5cae859b-2ff7-487e-875b-ffb45a2e5f4d	

[Download](#)[Delete](#)

Abbildung 15: Expo Build APK Success

9.3 Auf Google Play eine App hochladen

Besuchen Sie den Webauftritt der Google Play Developer Console und registrieren sich dort: Eine Anmeldung auf dieser Plattform ist für das Einreichen von Apps im Play Store notwendig. **Die Mitgliedschaft als Google Play Developer kostet Sie eine einmalige Gebühr von 25 US-Dollar und muss per Kreditkarte bezahlt werden.** Nachdem Sie den Bezahlvorgang abgeschlossen haben, müssen Sie anschließend nur noch die geforderten Informationen angeben und schon ist Ihre persönliche Developer Console einsatzbereit. Der „Entwicklername“ wird übrigens später im Google Play Store als Urheber der App angegeben – allerdings können Sie ihn noch ändern.

Der Upload Ihrer App erfolgt über die Google Play Developer Console. Nach dem Log-in auf der Plattform klicken Sie im Menü unter „Alle Apps“ den Button „Neue App hinzufügen“. Dort geben Sie dann die Sprache der Anwendung an und tragen den Namen der App ein (wobei Sie beide Angaben zu einem späteren Zeitpunkt auch wieder ändern können). Durch einen Klick auf „APK hochladen“ gelangen Sie in ein neu erstelltes Menü, das den Namen Ihrer App trägt. Dort sehen Sie auf der linken Seite verschiedene Menüpunkte („APK“, „Store-Eintrag“, „Einstufung des Inhalts“ etc.), denen Sie sich nun nach und nach widmen.

9.3.1 APK

Dies Menü dient dem Hochladen der signierten APK-Datei Ihrer App. Es stehen dort mehrere Auswahlmöglichkeiten zur Verfügung:

- Falls sich Ihre Applikation noch im Teststadium befindet, können Sie einen **Alpha- oder Betatest** für die App einrichten. Hierbei haben Sie die Wahl zwischen einem geschlossenen oder offenen Test. Bei einem geschlossenen Test bestimmen Sie selbst, welche Nutzer Ihre Anwendung testen – Sie laden diese per E-Mail zu dem Test ein (die Testpersonen benötigen hierfür entweder einen Google-Account oder ein Google-Apps-Konto). Demgegenüber richtet sich ein offener Test meist an eine größere Zahl an Nutzern, da hierbei Ihre Applikation von jedem Play-Store-Nutzer getestet und mittels eines privaten Feedbacks an Sie bewertet werden kann (öffentliche Bewertungen Ihrer App gibt es bei beiden Testformen nicht).
- Wenn Ihre App hingegen bereits getestet wurde und Sie schnellstmöglich Ihre Android-App veröffentlichen möchten, nutzen Sie die Option **„Erste APK-Datei in der Produktionsphase hochladen“**.

Unabhängig davon, ob Sie eine Testversion oder die finale Anwendung Ihrer Android-App einreichen möchten: In beiden Fällen laden Sie Ihre App in Form der signierten APK hoch, die Sie zuvor mit dem Android Studio erstellt haben. Damit befindet sich die App zwar in Ihrer Developer Console – allerdings ist sie bisher weder im Google Play Store verfügbar noch für die Veröffentlichung im App-Store angemeldet. Hierfür sind noch einige weitere Angaben in den anderen Menüpunkten vonnöten.

9.3.2 Store-Eintrag

Als Nächstes wenden Sie sich dem Store-Eintrag Ihrer App zu – in diesem geben Sie jene Informationen an, die Nutzer später auf der App-Seite im Google Play Store finden sollen.

- In der Sektion „Produktdetails“ können Sie die **Sprache und den Titel der App** ändern. Außerdem müssen Sie auch eine **kurze Beschreibung, die die App vorstellt, und eine zusätzliche, ausführlichere Beschreibung der Applikation eintragen** – beide Texte erscheinen nach der Veröffentlichung auf der Detailseite der App im Play Store.
- Des Weiteren benötigt jeder Eintrag im Play Store eine gewisse Anzahl an „Grafikinhalten“. So müssen Sie **mindestens zwei Screenshots** hochladen, die der Veranschaulichung Ihrer App im Play Store dienen. Die Bilder benötigen ein bestimmtes Format: Achten Sie dabei auf die Mindestgröße (320 Pixel) sowie die Maximalgröße (3840 Pixel). Außerdem dürfen Grafiken nur als JPEG- oder 24-Bit-PNG-Datei hochgeladen werden. Zudem darf das Seitenverhältnis der Screenshots nicht größer als 2:1 sein.
Neben den Screenshots sind auch ein **hochauflösendes Symbol** (als 32-Bit-PNG mit den Maßen 512 x 512 Pixel) sowie eine **Vorstellungsgrafik** (1024 Pixel breit und 500 Pixel hoch – als 24-Bit-PNG) obligatorisch. Ob man zusätzlich noch eine Werbegrafik, ein Banner für Android-TV-Geräte oder ein Werbevideo hinzufügt, steht einem aber selbst frei zu entscheiden.
- Nach dem Upload der Bilddateien wird nun unter „Kategorisierung“ der Inhalt der App näher eingegrenzt. In dem Feld **App-Typ** legen Sie fest, ob es sich bei der Android-App um ein Spiel oder um eine anderweitige Applikation handelt. In der Sektion **Kategorie** grenzen Sie anschließend das Themenfeld der Anwendung näher ein.
Als Nächstes müssen Sie unter Einstufung des Inhalts eine Einschätzung dazu abgeben, ob Ihre App einer Altersbeschränkung unterliegen sollte. Sie haben die Wahl zwischen vier verschiedenen Niveaus („alle“, „niedrig“, „mittel“, „hoch“). Mit der Einstufung „alle“ kennzeichnen Sie die App-Inhalte als unbedenklich für sämtliche Nutzer. Wenn in Ihrer Anwendung hingegen Dinge thematisiert werden, die für jüngere Nutzer eher ungeeignet sind, sollten Sie dies auch dementsprechend angeben. Mehr Informationen zu der grundlegenden Einstufung der App-Inhalte finden Sie auf der Hilfeseite der Google Play Developer Console.
In der Vergangenheit gab es lediglich diese vierstufige Bewertung der App-Inhalte. Mittlerweile hat Google die Beurteilung um den Punkt **neue Einstufung des**

Inhalts ergänzt. Dort sind die geforderten Einstellungen deutlich ausführlicher und die Beurteilung der App erfolgt durch die Beantwortung eines Fragebogens. Der Einstufung der App-Inhalte durch diesen neuen Fragenkatalog wird inzwischen eine hohe Relevanz beigemessen und dementsprechend findet man links in der Übersicht der Developer Console dazu einen eigenen Menüeintrag.

Mit dem Fragebogen können Sie sich jedoch auch später noch genauer auseinandersetzen – vorerst ist in diesem Schritt nur wichtig, dass Sie über das Drop-down-Menü die Inhalte Ihrer Applikation einer der vier Stufen zuordnen.

- Im Folgenden ist unter „Kontaktinformationen“ die **Nennung einer E-Mail-Adresse notwendig** – die Angaben zu einer Website und Telefonnummer sind hingegen optional.
- Die Sektion „Datenschutzerklärung“ markiert das letzte Feld im Store-Eintrag. Hinweise zum Datenschutz sind vor allem dann erforderlich, wenn Ihre App Zugriff auf vertrauliche Nutzer- bzw. Gerätedaten der User verlangt. Generell sollte aber jede App Informationen über die Speicherung und Verwertung von persönlichen Daten beinhalten – egal, wie viele Daten erhoben werden.

Die Datenschutzerklärung der App hinterlegen Sie auf einer Webseite und tragen die URL der Seite in dem dafür vorgesehen Feld ein. Falls Sie über keine eigene Website verfügen, auf der Sie die erforderliche Seite für die Datenschutzerklärung einrichten können, gibt es dafür viele alternative Optionen. Beispielsweise lässt sich über kostenlose Dienste wie Google Docs oder Google Sites eine solche Webseite erstellen.

Der Hinweistext einer Applikation ähnelt zu großen Teilen der [Datenschutzerklärung einer Website](#) – jedoch sind einige **app-spezifische Anpassungen** vonnöten (z. B. müssen Sie über die Zugriffsrechte der App und die Speicherdauer der erhobenen Daten informieren). Einen guten Überblick über die App-Datenschutzerklärung finden Sie auf datenschutzbeauftragter.info.de.

Nun haben Sie alle notwendigen Einstellungen im Store-Eintrag vorgenommen und speichern den Entwurf über den Button ganz oben auf der Seite. Als Nächstes widmen Sie sich der Bewertung der Inhalte Ihrer App.



Mit dem Upload der Applikation ist diese nicht direkt im Google Play Store verfügbar, denn **jede eingereichte App wird vor der Publikation von Google geprüft**. Wie viel Zeit dies in Anspruch nimmt, variiert von Fall zu Fall. Meist ist die eingereichte App jedoch nach wenigen Stunden im App-Store verfügbar.

Quelle: <https://www.ionos.de/digitalguide/websites/web-entwicklung/die-eigene-app-entwickeln-android-app-veroeffentlichen/>

9.4 Eigene App mit Google Play Console veröffentlichen

Der Eintrag wurde nach der obigen Anleitung durchgeführt, um die eigene App im Google Play Store zu veröffentlichen.


Noch ein Nachtrag zu der Anleitung: Die Veröffentlichung einer App dauert mehr als nur ein paar Stunden. Dies wird auch von Google so mitgeteilt.

▲ App-Name	Aktive Installationen ?	Bewertung auf Google Play ?	Zuletzt aktualisiert	Status
 Felchen-Statistik ch.tomasi_developing.felchenapp	—	★ —	19.02.2020	Veröffentlichung ausstehend 

Seite 1 von 1

Abbildung 16: Eigene App in der Warteschlange

Wenn alles ausgeführt wurde wird die App hochgeladen und kommt in die Warteschlange wo die App von Google geprüft wird.

▲ App-Name	Aktive Installationen ?	Bewertung auf Google Play ?	Zuletzt aktualisiert	Status
 Felchen-Statistik ch.tomasi_developing.felchenapp	—	★ —	22.02.2020	Veröffentlicht

Seite 1 von 1


Abbildung 17: Eigene App veröffentlicht

Ist die App von Google freigegeben bekommt man eine Nachricht und die App ist ab dann sofort im Play Store zum Download verfügbar.

Ein Update der App kann ab dann auch einfach mit der Google Play Console hochgeladen und verwaltet werden. Wichtig dabei ist das der Versionscode in der App.js höher ist als die veröffentlichte App.

Das Update wird dann auch viel schneller von Google bearbeitet als ein neues Realease.

Filtern ▼

▲ App-Name	Aktive Installationen ?	Bewertung auf Google Play ?	Zuletzt aktualisiert
 Felchen-Statistik ch.tomasi_developing.felchenapp	—	★ —	22.02.2020

Benachrichtigungen


 **Dein Update ist online** ✕
Dein Update für Felchen-Statistik, das am 22.02.2020, 21:43 erstellt wurde, ist jetzt im Store online.
Heute

Abbildung 18: Eigene App Update

9.5 Eigene App im Google Play Store

Wurde die App von Google veröffentlicht ist sie dann direkt über den Google Play Store zum Download bereit.

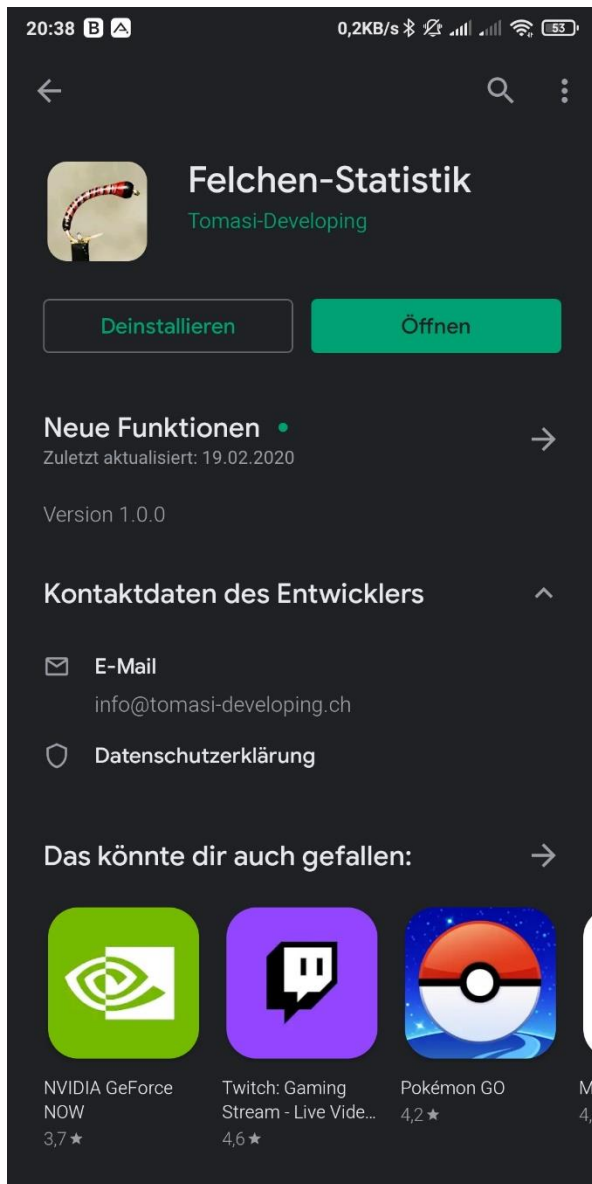


Abbildung 19: Eigene App im Store

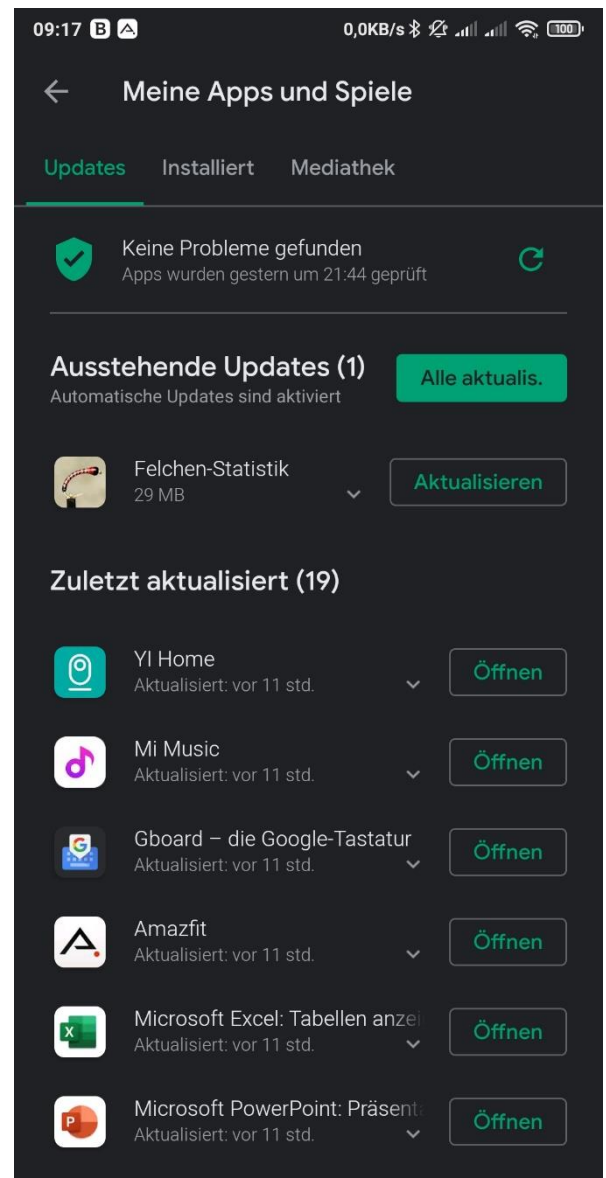


Abbildung 20: Eigene App bekommt Update

10 Tests Cases

Da die Applikation als APK Datei vorliegt wurde diese auf meinem Smartphone installiert und die Tests gleich dort ausgeführt.

Tester: Patrick Tomasi
Smartphone: Smartphone Xiaomi Mi Mix 2s
Android Version: 10QKQ1.190828.002
Android Sicherheitspatch-Level: 2020-01-01

10.1 Test Case Home Screen

Schritt	Aktivität	Erwartetes Resultat	Erfüllt
1	App wird gestartet	Anzeige keine Daten	✓
2	Fang hinzufügen ohne Daten	Der Fang ohne Daten wird nicht in der Cloud gespeichert	✓
3	Fang hinzufügen mit Daten	Der Fang wird in der Cloud gespeichert und wird im Home Screen angezeigt	✓
4	Klick auf den hinzugefügten Fang	Der Detail Screen öffnet sich mit den Detaillierten Daten des Fanges	✓
5	Button löschen klick im Detail Screen	Ein Alert öffnet sich mit der Frage ob der Eintrag gelöscht werden möchte	✓
6	Abbrechen auf dem Löschbutton	Der Alert wird geschlossen und der Eintrag nicht gelöscht	✓
7	Löschen auf dem Löschbutton	Der Alert wird geschlossen, Fang aus der Datenbank gelöscht und auf Home Screen weitergeleitet	✓
8	App neu laden ohne Internetverbindung	Alert mit Hinweis das keine Internetverbindung vorhanden ist	✓
9	App neu laden mit Internetverbindung	Die zuvor gespeicherten Daten werden aus der Cloud geladen und angezeigt	✓
10	Pull to Refresh	Daten werden neu geladen aus der Cloud, Actvitildentictor wird beim Laden angezeigt	✓

Tabelle 1: Test Case Home Screen

10.2 Test Case Wetter Screen

Schritt	Aktivität	Erwartetes Resultat	Erfüllt
1	Wetter Tab wird gedrückt	Wetterdaten werden geladen mit aktuellem Ort Zeihen	✓
2	Wetterdaten in Frick laden	Wetterdaten werden geladen mit aktuellem Ort Frick	✓
3	Wetterdaten ohne Internetverbindung laden	Alert mit Hinweis keine Internetverbindung wird ausgegeben, Anzeige Wetterdaten konnten nicht geladen werden inkl. Button um Wetterdaten neu zuladen	✓
4	Wetterdaten erneut laden mit Internetverbindung	Wetterdaten werden erneut geladen und angezeigt.	✓

Tabelle 2: Test Case Wetter Screen

10.3 Test Case Einstellungen Screen

Schritt	Aktivität	Erwartetes Resultat	Erfüllt
1	Einstellungen Tab wird gedrückt	Das Einstellungen Tab wird geladen und die Daten angezeigt	✓
2	Button-E-Mail an Support	Das Mail Programm des Smartphones wird geöffnet mit korrekter E-Mail-Empfänger Adresse und Betreff: Felchen APP	✓

Tabelle 3: Test Case Einstellungen Screen

11 Reflexion

Das Modul hat mir sehr viel Spass bereitet und war sehr interessant. Ich wollte schon länger mal eine App entwickeln und habe auch schon einen kleinen ersten Versuch mit Angular und Ionic gemacht.

Das Entwickeln der Felchen Statistik App hat mich aber sehr viel mehr Zeit gekostet als ich erwartet hatte. Ich musste vieles nachlesen und nach Lösungen suchen für meine Probleme.

Dadurch konnte ich aber sehr viel lernen und auch in die App einbringen. Ich konnte so auch viele Komponenten von React einbinden.

Am Ende bin ich sehr stolz das auf dem eigenen Smartphone die eigene App läuft. Sie wird auch beim nächsten Angeln auf Felchen zum Einsatz kommen.

Die App hat noch einige Punkte die Verbessert werden können. Ein nächstes Ziel ist bestimmt das es ein Benutzerlogin für die App geben wird damit jeder Benutzer sein eigenes Konto in der App hat und bei Firebase hat.

Die App wird auch an Andy und ein paar weitere Angler zum Testen weitergegeben und wen diese auch gefallen an der App haben werde ich mich nach meiner Lehre nochmals an die App setzen und sie erweitern mit einem eignen Backend damit die Daten nicht mehr in Firebase gespeichert werden sondern in einer eigenen Datenbank was dann auch für die Angler Interessanter wird wen auf die Daten eine Auswertung gemacht werden kann und so eventuell bestimmte Muster für Jares Zeiten, Gewässer und Nymphen Arten ausgewertet werden können.

12 Abbildungsverzeichnis

Abbildung 1:Use-Case Diagramm	9
Abbildung 2:Zustandsdiagramm Home Tab	10
Abbildung 3: Zustandsdiagramm Wetter- und Einstellung Tab	10
Abbildung 4: Mockup Home Screen	11
Abbildung 5: Mockup Detail Screen	12
Abbildung 6: Mockup Fang hinzufügen Screen	13
Abbildung 7: Mockup Wetter Screen.....	14
Abbildung 8: Mockup Einstellungen Screen.....	15
Abbildung 9: GitHub Repository	17
Abbildung 10: Einstellungen Screen	19
Abbildung 11: Wetter Screen	20
Abbildung 12: Home Screen	21
Abbildung 13: Detail Screen	22
Abbildung 14: Fang hinzufügen Screen.....	23
Abbildung 15: Expo Build APK Success.....	25
Abbildung 16: Eigene App in der Warteschlange	29
Abbildung 17: Eigene App veröffentlicht	29
Abbildung 18: Eigene App Update.....	29
Abbildung 19: Eigene App im Store	30
Abbildung 20: Eigene App bekommt Update.....	30

13 Tabellenverzeichnis

Tabelle 1: Test Case Home Screen	31
Tabelle 2: Test Case Wetter Screen	32
Tabelle 3: Test Case Einstellungen Screen	32