base.camp Project Proposal: EHIC2IoT

Jan Harder (7027986) Johnvir Khattar (7029342)

11.06.2020

Abstract

When doctors and laboratories are communicating about probes, e.g. those taken from corpses, the documentation of test results is currently handled by hand-written notes and pages which have to be physically transmitted to each party or have to be typed in a software manually. EHIC2IoT wants to help doctors, laboratories and medical staff to deal with probes more efficiently so that measurement can be applied faster.

INTRODUCTION

The basic idea for this project one may find here: https://devpost.com/software/ehic2qr. During the EUvsVirus Hackathon in 2020 students of the university of Hamburg came together to build an app for the laboratories especially in Hamburg who are dealing with the probes taken from suspected Coronavirus infected people.
The idea is to extend and adjust the EHIC2QR App so that it has not only one use case of helping the medical procedure when dealing with a Coronavirus case, but rather helping the general procedure of medical staff communicating with laboratories. To get the time and effort in this communication task reduced is a very crucial challenge as it is the key factor of how fast infectious diseases can spread. Just imagine, today an infected person possibly could be told one week after testing if he/she was tested positive or negative. This is unacceptable and that is why Dr. Armin Hoffmann from the UKE, Hamburg and J. Harder and J. Khattar want to build this app.

METHOD AND IMPLEMENTATION

So the first step which we already finished is to brainstorm with A.Hoffmann who is a doctor at the UKE how we can extend and change the app EHIC2QR so that we have a broader use case and benefit.
Therefore, we found that following functionalities have to be added in the product backlog to get the EHIC2IoT run:

- **Communication** of the app with printers in hospitals
- **Usage of GPS** so that we can verify and locate every probe taken via Google Maps: It is known that GPS probably is not the most accurate technology to get the address but for our use case it is already an improvement if 70-80% of the address data is filled automatically, so that the final address can be simply adjusted by the doctor who types in the data. UX for this feature: By clicking a button “Adresse eintragen” the current address of the device will be received via GPS and the address fields in the app will be automatically filled. Now the user

has to check if there is everything ok with the address. If not, the user can manually change that address data. This is an improvement as the doctor/user will get 70-80% of the data automatically filled, so that this task will take significantly less time.

- **Adjusted UX** so that this app can be used for different laboratories and for multiple diseases.
    - Skeleton-App so that different laboratories can use this app. Every laboratory has a different standard of communication. This means that the rules applied to generate the QR-Code will be different from laboratory to laboratory. That is why there has to be a screen/space where the doctor/laboratory can decide via which standard they want to generate the labels.
    - So one main aspect of the UX has to be changed as there has to be a functionality that the laboratories can name their standard of encoding and then can apply this to their labels of taken probes. As one doctor could have multiple standards depending on the laboratory he/she is currently working with, there must be a space to decide which standard will be used. In this specification of the app there will be no Corona-specific data collected as the disease is not necessary for the sample/probe itself.

The app will be built in react-native and will be deployed as a serverless app. As we are doing this as a scientific project, we would like to get 3CPs for the creation of EHIC2IoT

DATA PRIVACY

EHIC2IoT does not need any internet connection to work. As we are dealing with very personal and sensitive data the app user is the only one who will see and edit the data when going through the app process. While working with the app the user will type in all the information needed. After typing in all the information the user gets a label which he/she can print out directly from the app. The printed sticker can be applied to any document so that when someone else wants to read the information he or she has to scan the barcode/QR Code and then get all the information needed. The information is not stored at any storage, or similar, but rather just encoded within the QR-Code/Barcode according to a specific standard. This kind of technique of communication is already used in some laboratories across Germany. If the phone gets lost, or stolen any user who can access the phone could also possibly access the app. To deal with this kind of misuse it is necessary to have some kind of authentication running before the app can be used. This could also be done with Google Firebase Auth Tools. This feature is out of scope for the initial MVP which we are building here. The process of the app is that only one probe is dealt with at a time. After the whole process of one probe is finished the data of this probe will be printed offline and the task of any next probe will be started. In case of any virus attack the worst case is that data from one single patient could get stolen. This certainly has to be discussed for future releases building on top of this MVP.

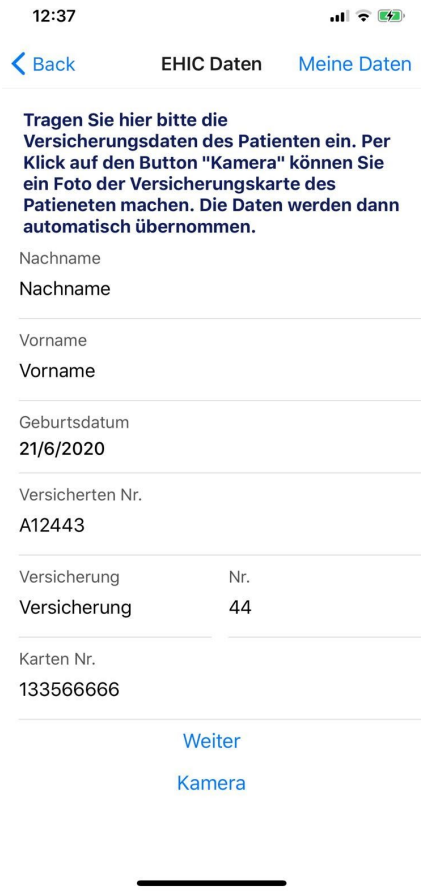
RELATED WORK

Please also see here for description and video of previous project:

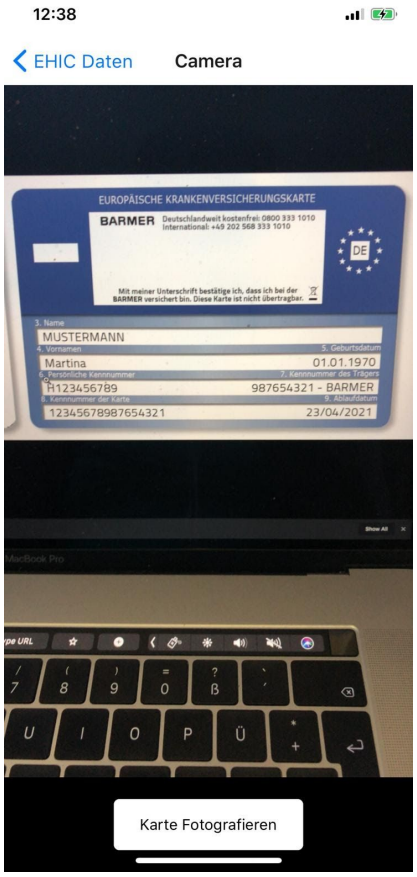
HTTPS://DEVPOST.COM/SOFTWARE/EHIC2QR

base.camp Project Results: EHIC2IoT

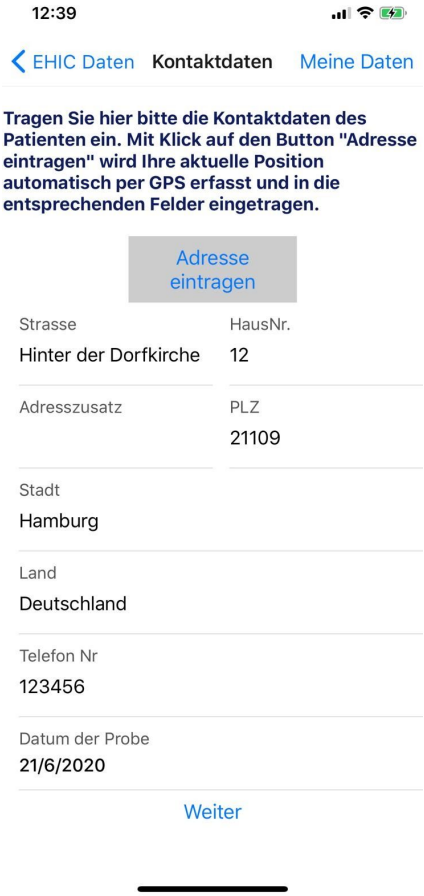
Landing Page der Anwendung: In Form eines Hackathons wurde eine Lösung für Ärzte und Labore gebaut, bei welcher die Kommunikation zwischen diesen beiden Instanzen effizienter abläuft. Ärzte haben jetzt die Möglichkeit nicht nur Corona-Test-Proben, sondern auch jegliche Art von Probe und die zugehörige Kommunikation mit den Laboren, mit Hilfe der Anwendung abwickeln kann. Ziel dieses Hackathons war es, die bestehende Anwendung so umzugestalten, dass diese nicht nur für einen Corona-Usecase verwendet werden kann und zusätzlich mit IoT-Geräten per Bluetooth (aktuell spezialisiert auf Drucker) und GPS kommunizieren kann.



EHIC Datenerfassung: Dieses Projekt baut auf eine vorherige Anwendung, bei der es möglich gemacht wurde, mithilfe der Smartphone-Kamera die Krankenkassenkarte im EHIC-Standard zu fotografieren, wodurch automatisch die relevanten Daten in die entsprechenden Datenfelder eingetragen werden.



EHIC-Datenextraktion aus Kamera: Hier sieht man, wie ein User eine Krankenkassenkarte fotografieren kann. Anschließend würden die Daten von der Karte in die Datenfelder aus dem vorherigen Screenshot eingetragen.

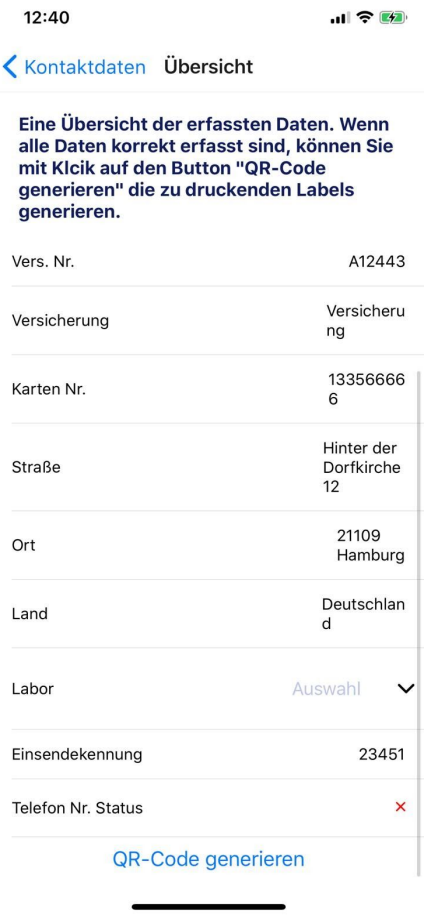


Kontaktdaten: Im Zuge dieses Hackathons wurde eine GPS-Funktionalität in der Anwendung implementiert. Durch Klick auf den Button “Adresse eintragen”, wird per GPS eine Ortung des Smartphones durchgeführt. Dadurch werden die Kontaktdaten des Patienten, oder der Ort der Probenentnahme größtenteils automatisch erfasst. Der Arzt, der die Probe entnimmt, spart sich somit Zeit, die Adresse ausfindig zu machen. Insgesamt wird der Prozess der Probenentnahme und der -auswertung transparenter. Die Genauigkeit der Adresse bei der Nutzung von GPS hat bei unseren Tests ca. in 90% der Fälle direkt die richtige Adresse erreicht.

Die UX nochmal erläutert:

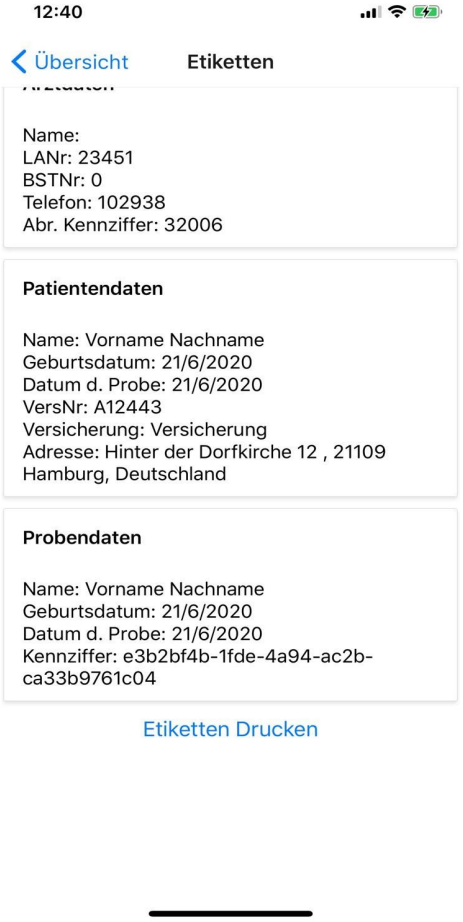
Bei Klick auf den Button “Adresse eintragen” wird die Adresse automatisch gefunden und eingetragen. Der Benutzer muss nun aber nochmal die Felder kontrollieren, bevor auf Weiter geklickt wird. Dies ist insofern hilfreich, da im Normalfall 80-90% der Daten schon korrekt sind und der Arzt nur noch zusätzliche Infos/ kleine Korrekturen vornehmen muss.

Globaler “Meine Daten”-Bereich: Hier kann der Nutzer globale Stammdaten anlegen. Welche Labore angebunden sind und welche Daten der Anwender hat. Diese Angaben erfolgen hier und werden bei Erstellung der Labels berücksichtigt.

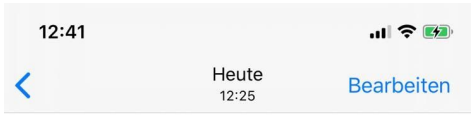


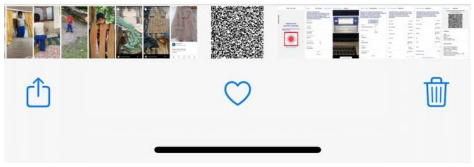
Zusammenfassung: Eine Übersicht der eingegeben Daten. Hier sieht der Arzt, welche Daten eingegeben wurden. Der Arzt hat an dieser Stelle die Möglichkeit mit dem Klick auf das Feld “Labor” ein Labor auszuwählen. Dies ist insofern relevant, als dass die Auswahl des Labors darüber entscheidet, nach welchem Standard die erfassten Daten codiert werden. Ist ein Labor ausgewählt, kann der Arzt auf den Button “QR-Code generieren” klicken und gelangt dann zu den zu druckenden Dateien.

QR-Code und Labels zum Ausdrucken: Der Arzt hat hier eine Übersicht der druckbaren Label und Codes. In diesem QR Code sind jetzt alle erfassten Daten codiert, so dass ein entsprechender Scan alle relevanten Daten aufzeigen würde. Zusätzlich sieht man die Labels, welche gedruckt werden können. Diese dienen dem Arzt dazu, auf entsprechenden Dokumenten Label-Sticker anzubringen und so eine Transparenz und Konsistenz in der Dokumentation zu erreichen.



Druck der Etiketten: Durch Klick auf den Button “Etiketten Drucken” wird der Druck der QR Codes und der Etiketten ausgelöst. Prototypisch wurde zunächst nur der Druck des QR Codes getriggert, da dieser am relevantesten ist. Der Arzt kann jedoch auswählen, welches Element er drucken möchte.





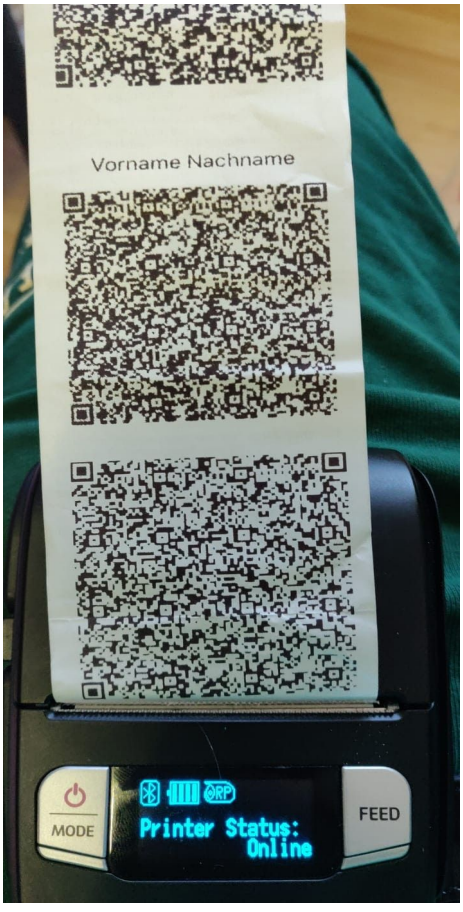


Bild wurde gedruckt: Der Arzt wird die gedruckte bzw. druckbare Datei in seiner Fotobibliothek auf dem Smartphone finden. Sobald der Arzt nach seinem Arbeitstag inkl. der Probenentnahmen bei allen Patienten vor Ort durch ist, kann er in seinem Büro dann alle Dateien drucken und dokumentieren. Der Grund für die Gestaltung des Prozesses, so wie hier beschrieben, ist der, dass die Nutzer der Anwendung somit nicht an ein spezifisches Druck-Gerät gebunden sind.

Datenschutz:
Die Patientendaten werden nur für die Erstellung des Labels in der App gespeichert. Spätestens bei der Eingabe von neuen Patientendaten werden die vorherigen Daten überschrieben. So werden nie mehr als die Daten von einer Person zur Zeit auf dem Gerät des Arztes gespeichert um einen höchstmöglichen Datenschutz zu gewährleisten. Das erstellte Label kann von Arzt in die Galerie des Gerätes exportiert werden. In diesem Fall wären die Daten dauerhaft auf dem Gerät gespeichert. Die Ärzte sollen jedoch angehalten werden, die Labels nach dem Drucken vom Gerät zu löschen. Der bewusste Umgang der Ärzte mit den Patientendaten scheint uns realistisch, da sie tagtäglich mit dem Datenschutz konfrontiert sind.

Druck der Anwendung: In diesem Bild sieht man, wie der lesbare QR Code (aktuellster Druck, unten im Bild) gedruckt wird. Dieser QR Code hat alle Informationen, die der Arzt benötigt gespeichert. Die Verbindung zum Drucker wurde per Bluetooth hergestellt. Bei dem Drucker handelt es sich um einen mobilen Belegdrucker einer führenden Marke (**Star SM-L200 series).**

EHIC2IoT - Implementation & Data Privacy

Frameworks

Frontend & State-Management:
react-native + redux

Dependencies:

```
"@react-native-community/cameraroll": "^1.5.2",
    "@react-native-community/cli": "^4.8.0",
    "@react-native-community/datetimepicker": "^2.3.2",
    "@react-native-community/geolocation": "^2.0.2",
    "@react-native-community/masked-view": "^0.1.10",
    "@react-native-firebase/app": "^6.4.0",
    "@react-native-firebase/ml-vision": "^6.7.1",
    "@react-navigation/bottom-tabs": "^5.2.7",
    "@react-navigation/native": "^5.1.6",
    "@react-navigation/stack": "^5.2.13",
    "moment": "^2.24.0",
    "native-base": "^2.13.12",
    "pod": "^0.9.0",
    "react": "16.11.0",
    "react-native": "0.62.2",
    "react-native-camera": "^3.23.1",
    "react-native-geocoding": "^0.4.0",
    "react-native-geolocation-service": "^5.0.0",
    "react-native-gesture-handler": "^1.6.1",
    "react-native-image-to-pdf": "^1.2.0",
    "react-native-pdf-lib": "^1.0.0",
    "react-native-qrcode-generator": "^1.2.2",
    "react-native-reanimated": "^1.8.0",
    "react-native-safe-area-context": "^0.7.3",
    "react-native-screens": "^2.7.0",
    "react-native-view-shot": "^3.1.2",
    "react-native-webview": "^9.3.0",

    "react-navigation": "^4.3.8",
```

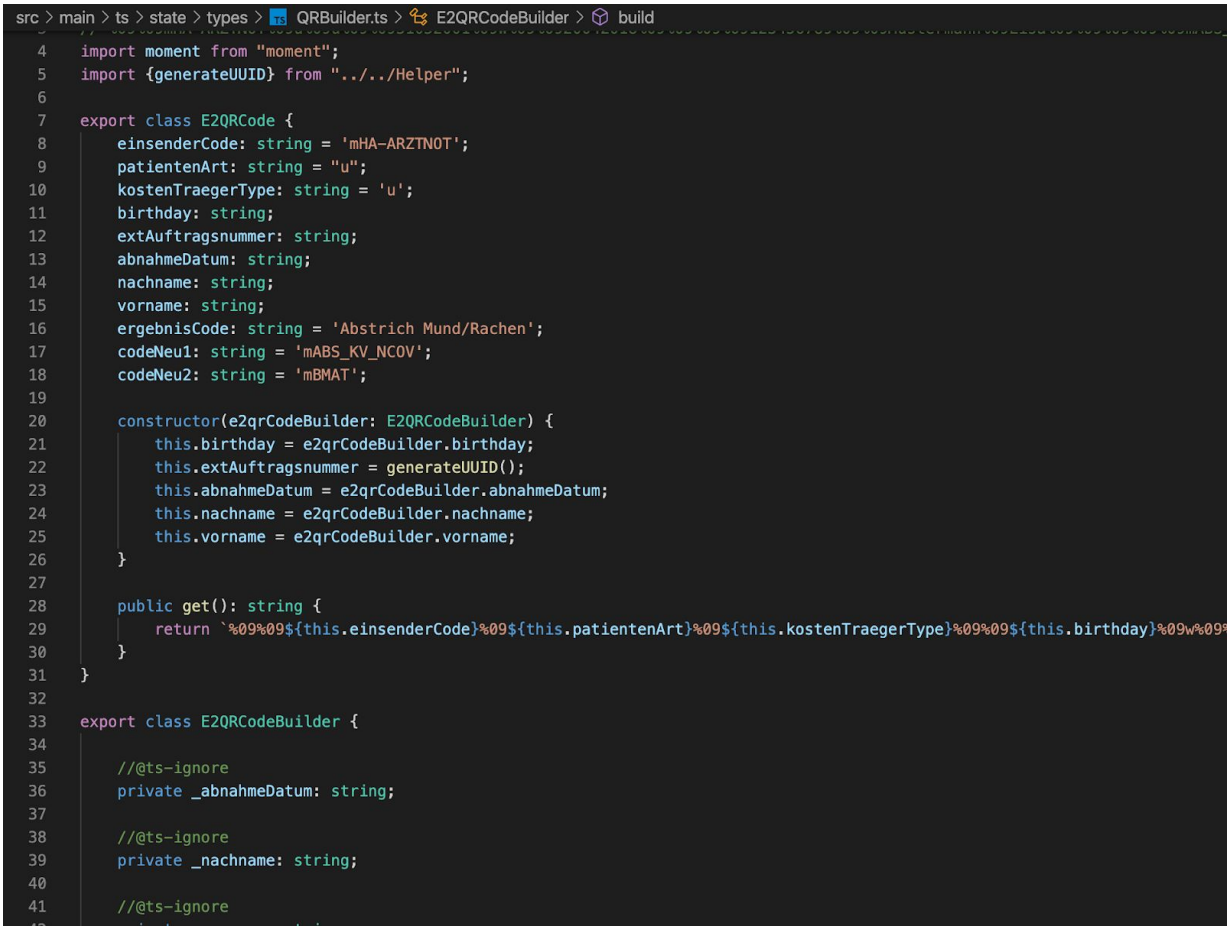
GPS (Geolocation)

```
52      Geolocation.getCurrentPosition(
53        (position) => {
54          console.log(position);
55          console.log(position.coords.latitude);
56          console.log(position.coords.longitude);
57          Geocoder.from({
58            lat: position.coords.latitude,
59            lng: position.coords.longitude,
60          }).then((json) => {
61            var addressComponent = json.results[0].address_components;
62
63            var geoHNr = addressComponent[0].long_name;
64            var geoStreet = addressComponent[1].long_name;
65            var location = addressComponent[2].long_name;
66            var geoCity = addressComponent[3].long_name;
67            var geoCountry = addressComponent[5].long_name;
68            if (geoCountry == "Germany"){
69                  geoCountry = "Deutschland";}
70
71            var geoPlz = addressComponent[6].long_name;
```

Die Position des Arztes wird per GPS-Ortung mittels seines Mobiltelefons ermittelt. Um die GPS-Funktion des Gerätes nutzen zu können, haben wir das "react-native-geolocation-service"-Modul installiert und die Abfragen für die erforderlichen Berechtigungen definiert. Da wir die GPS-Ortung nur während der App-Nutzung benötigen, waren die Berechtigungen "`NSLocationWhenInUsage`" (iOS) und "`ACCESS_COARSE_LOCATION & ACCESS_FINE_LOCATION`" (Android) ausreichend. Mit Hilfe des Aufrufs *Geolocation.getCurrentPosition()* haben wir den Längen- und Breitengrad der Position des Arztes ermittelt und zwischengespeichert. Die Koordinaten müssen nun in die Adresse umgewandelt werden, an der sich der Arzt befindet (Geocoding). Dazu nutzen wir das Modul "react-native-geocoding". Mit Hilfe dieses Moduls können wir auf die Google-Maps-Geocoding-Api zugreifen. Zunächst müssen wir die Geocoding-Implementation jedoch durch Eingabe eines Google-Maps keys konfigurieren, da die Google-Maps-Api bei hoher Anzahl von Abfragen kostenpflichtig wird. Die Anfrage mittels Geocoder.from() liefert ein JSON-Objekt in welchem unter anderem die Straße, Hausnummer und Stadt der aktuellen Position gespeichert sind. Wir nutzen nur die "long_names" (die ausgeschriebnen und keine Abkürzungen) der Straßen und Städte, um die Daten so genau wie möglich zu halten. Da die Daten in Englisch zurückgegeben werden, übersetzen wir "Germany" per simpler if-Abfrage ins Deutsche. Die Straßennamen haben keine Englische Übersetzung

und können daher so genutzt werden. Auch Städte mit abweichendem Namen (z.B “Munich”) werden von der API in Deutsch zurückgegeben und benötigen daher keine Übersetzung. Anschließend werden die Daten im State gespeichert und der Arzt kann noch Änderungen vornehmen.

QR-Code generation



Der QR-CODE wird anhand einer QR-Code-Builder Funktion erstellt. Dieser Klasse/Funktion werden die zu codierenden Daten übergeben und anschließend übernimmt diese Funktion die codierung. Aus der Datensicherheits und Privatsphäre-Perspektive lässt die Anwendung es zu, dass für

die Erfassung der Daten keine Kommunikation mit dem Internet stattfinden muss. Der Arzt muss den generierten QR-Code lediglich drucken und anschließend für Dokumentationszwecke aufbewahren. Die gedruckte Datei kann sicher im Krankenhaus-/ Labornetzwerk abgelegt werden, oder auch gelöscht werden. Diese Verantwortung obliegt dann dem Arzt.

Label Printing

```
async function print() {
  if (Platform.OS === 'android') {
    await PermissionsAndroid.request(
      PermissionsAndroid.PERMISSIONS.WRITE_EXTERNAL_STORAGE,
    );
  }

  console.log('PRESS PRINT');
  //handler to take screnshot
  viewShotRef0.current.capture().then(
    //callback function to get the result URL of the screnshot
    (uri) => CameraRoll.save(uri), //
    (error) => console.error('Oops, Something Went Wrong', error),
  );
  Alert.alert('Datei wird gedruckt!');
}
```

Als große Challenge stellte sich heraus, dass die Zugriffe auf Dateisysteme auf Android und IOS unterschiedlich erfolgen. Während man bei Android Abfragen im Code implementieren muss, werden bei IOS über Key-String Parameter in XCode (IDE für IOS-Entwicklung) entsprechende Abfragen automatisch getriggert, sobald die Anwendung eine bestimmte native Funktionalität verwenden möchte, wie das Speichern von Dateien, oder das Tätigen von Bildschirmaufnahmen. Während im ersten Hackathon primär auf IOS optimiert wurde, wurde im Zuge dieses Hackathons Tests sowohl auf Android, als auch IOS durchgeführt und sichergestellt, dass keine ungewollte Datenkommunikation außerhalb der Anwendung geschieht. Dies ist für solch sensible Daten, wie sie hier in der Anwendung erfasst werden essentiell und damit sehr wichtig für die Ärzte, welche diese Anwendung nutzen.

Dadurch, dass die Labels lokal auf dem Smartphone gespeichert werden und keine Kommunikation/Schnittstelle direkt aus der Anwendung heraus mit dem Drucker implementiert wurde, ist es für die Abwicklung des Prozesses nicht relevant, welche Art von Drucker im Labor, oder beim Arzt vorliegt. Der

Testdrucker, der in der Entwicklung verwendet wurde, ist ein relativ beliebter mobiler Drucker der Firma Star Micronics. Das Modell des Druckers lautet: SM-L200. Dieser Drucker wurde für das Testen ausgesucht, da er ein reiner Bluetooth und USB-Drucker ist und somit ebenfalls keine Internetkommunikation benötigt, welche ein potentielles Risiko für die Datensicherheit und Privatsphäre darstellen könnte. Für diesen Drucker gibt es hier auch eine Dokumentation, wie eine entsprechende Schnittstelle direkt in der react-native Anwendung E2IoT implementiert werden könnte: https://github.com/infoxicator/react-native-star-prnt. Wir haben das Modul zunächst in unserer Anwendung installiert um die Kommunikation mit dem Drucker direkt aus der App möglich zu machen. Der Drucker war in der App mit seinen Eigenschaften (mac-Adresse, Port-Nummer,…) sichtbar, jedoch war das Drucken aufgrund der nicht gepflegten Api nicht möglich. Daher haben wir uns dazu entschieden, die QR-Codes lokal auf dem Gerät zu speichern. Dies hat den Vorteil, das die QR-Codes mit Hilfe der Hersteller-Apps des Druckers einfach gedruckt werden können und die App nicht von einem Hersteller abhängig ist. Da keinen Zugriff auf ein Gerät von Hersteller, welchen das UKE verwendet, hatten, haben wir es bei dieser universellen Lösung belassen.