

Dokumentation Viva Corona - Team Foxtrot



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Christian Kuessner, Timo Kullmann, Benjamin Laumann, Patrick Siebke, Kolja Straub, Patrick Vimr
22. September 2020

Inhaltsverzeichnis

1 Motivation	2
2 Technische Dokumentation	3
2.1 Übersicht	3
2.2 Backend	3
2.2.1 Datenbank	3
2.2.2 Deployment	3
2.3 Frontend	4
2.3.1 Interaktive Elemente	4
2.3.2 Hintergrunddienste	5
3 Benutzerhandbuch	6
3.1 User	6
3.1.1 Login und Registrierung	6
3.1.2 Standortverlauf	6
3.1.3 Infektionsstatus	7
3.1.4 Handel - Angebote	9
3.1.5 Handel - Angebotssuche	9
3.1.6 Handel - Warenge such	10
3.1.7 Handel - Supermärkte	11
3.1.8 Achievements	12
3.1.9 Check Status	13
3.1.10 Quiz	14
3.2 Admin	15
3.2.1 Admin Promotion	15
3.2.2 Spreadmap	15
3.3 Administrative Funktionen	16
3.3.1 Admin-JWT	16
3.3.2 Kategorien hinzufügen	17
3.3.3 Admin Promotion & Widerrufung	17
3.3.4 QR-Code Generierung	17
4 Features	19
5 Zusammenfassung	20

1 Motivation

Das Jahr 2020 ist geprägt vom Coronavirus (SARS-CoV-2). Das Virus betrifft nicht nur die öffentliche Sicherheit und die allgemeine Gesundheit jedes einzelnen, sondern hat auch einen sehr großen Einfluss auf das soziale Leben. Um dieses trotzdem erträglich und vor allem sicher zu gestalten, haben wir uns die Aufgabe gegeben, eine App zu entwickeln, die jeden in diesen schwierigen Zeiten unterstützt.

Aufgrund von stetig schwankenden Infektionszahlen und dem Risiko einer massenhaften Ausbreitung des Virus, ist es wichtig eine Möglichkeit der Nachverfolgung von Infektionen zu haben. Kontakte mit infizierten Personen sollten an die Betroffenen einfach weitergeleitet werden können, damit diese sich auf das Virus testen lassen. Durch eine schnelle Warnung an Kontakt Personen von Infizierten kann das Virus besser eingedämmt werden, indem die Betroffenen die erforderlichen Schritte unternehmen.

Zur heutigen Zeit ist das Thema Privatsphäre für die deutsche Bevölkerung von großer Bedeutung. Deshalb ist es im Kontext einer solchen Tracking-App wichtig, dass diese so anonym wie möglich gestaltet wird, damit niemand durch andere Benutzer zurückverfolgt werden kann. Dadurch wird das Sicherheitsgefühl durch frühzeitige Benachrichtigung über problematische Kontakte erhöht und gleichzeitig die Gewährleistung der eigenen Privatsphäre sichergestellt.

In Krisenzeiten, wie es im Frühjahr 2020 der Fall war, ist aber nicht nur die Gesundheit der Bevölkerung zu sichern, sondern auch die Versorgung mit lebensnotwendigen Gütern, wie bspw. Lebensmitteln und Hygieneartikeln. Dafür ist es ebenso wichtig, eine Plattform zum Handel von Produkten und zum Informieren von Verfügbarkeiten gewisser Produkte bereitzustellen.

Das Coronavirus beeinflusst aber nicht nur die physische Verfassung der betroffenen Menschen. Auch die Psyche der Bevölkerung leidet mutmaßlich unter den Maßnahmen, die zur Eindämmung des Coronavirus ergriffen wurden und für jeden Einzelnen verpflichtend sind. So leiden seit Beginn der Coronapandemie zum Beispiel zunehmend mehr Menschen an Depressionen¹. Um diesen Folgen etwas entgegenzuwirken und um für einen Lichtblick am Ende des Tunnels zu sorgen, soll auch der Spaß bei der Nutzung der App nicht fehlen. Nur wenn die Nutzer die App gerne verwenden, kann sie einen Beitrag leisten, um diese Zeit schnell zu überstehen, indem die Bevölkerung zur Mitarbeit und Hilfe motiviert wird.

Um einen Beitrag zur Verbesserung der Gesamtsituation zu leisten, entwickeln wir daher die App 'Viva Corona'. Sie vereint die obigen drei Bereiche und stellt damit eine Grundlage bereit, um das Coronavirus und seine Folgen einzudämmen.

¹<https://idw-online.de/de/attachmentdata80057.pdf>

2 Technische Dokumentation

2.1 Übersicht

Viva Corona funktioniert nur unter ein paar Annahmen korrekt, die in Rücksprache mit dem Auftraggeber getroffen wurden. Diese sind wie folgt:

- Personen sind solange infiziert, bis sie einen entsprechenden Test machen, der die Genesung bestätigt (s. Frontend)
- Genesene Personen können sich nicht erneut infizieren

2.2 Backend

Das Backend besteht aus einer REST-Schnittstelle und einem Websocket-Service, der sich um das Versenden von Benachrichtigungen an Nutzer kümmert. Um eine gute Verwendbarkeit durch Programmierer zu gewährleisten ist die REST-Schnittstelle mit OpenAPI spezifiziert.

Diese API lässt sich so über das auch im Produktivmodus erreichbare Swagger-UI studieren und benutzen, was nicht nur für Testzwecke, sondern auch für Administratoren der App sehr praktisch ist.

Die gewählte Programmiersprache ist TypeScript mit der Laufzeitumgebung NodeJS. TypeScript wurde gewählt, da bei der Entwicklung durch die statische Typisierung einige Fehler verhindert werden, die in einer rein dynamisch typisierten Sprache, wie JavaScript, verbreitet sind. Außerdem bietet TypeScript eine bessere Dokumentation der internen Schnittstellen, was die Entwicklung erleichtert.

Um sauberen Code sicherzustellen und einige Fehler zu finden haben wir eine Kombination aus automatischer Formatierung über die verwendete IDE Visual Studio Code, dem Linterframework eslint und einer strikten Konfiguration des TypeScript Compilers verwendet.

Für Tests nutzen wir das Mocha Framework und verwenden hierfür eine volatile in-memory Instanz von MongoDB. Diese Tests werden neben dem Linting auch durch ein npm-Skript ausgeführt, das durchlaufen muss, bevor ein Merge-Request von dem Peer-Reviewer akzeptiert wird.

Die Anwendung baut auf dem Webframework Express auf und gliedert sich in Router, Controller, Datenbankschemata und Datenbankzugriffskapselung. Dies ermöglicht eine saubere Trennung der einzelnen Schichten. Auch die einzelnen Anwendungsbereiche des Backends (Trading, Quiz, etc.) sind durch eine Aufgliederung in unterschiedliche Ordner und durch die Verwendung von Modulen getrennt.

Die Authentifizierung der Nutzer und Admins wird jeweils über sogenannte JSON Web Token abgewickelt. Das sorgt dafür, dass der Server sich nicht merken muss, welche Clients angemeldet sind, ohne dass der Nutzer sein Passwort bei jeder Kommunikation mit dem Server eingeben muss.

2.2.1 Datenbank

Als Datenbank verwenden wir die dokumentenbasierte MongoDB. Diese ermöglicht die Formulierung schneller ortsbasierte Datenbankabfragen, weshalb sie sich besonders gut für die Kontaktverfolgung, sowie andere ortsbezogene Funktionalitäten der App eignet. Außerdem sprach für MongoDB, dass nur ein kleiner Anteil der Datenbankanfragen relationaler Natur ist und das Dokumentenformat, sowie das Ökosystem sehr gut zu NodeJS passen.

Zur Validierung von Nutzereingaben gegen ein Schemata und dem Abbilden von Dokumenten auf Objekte haben wir das Mongoose ODM verwendet.

2.2.2 Deployment

Damit der Server einfach und reproduzierbar bei allen Entwicklern, sowie auf dem Produktivserver gestartet werden kann, benutzen wir Docker Container um unsere Anwendung in einer reproduzierbaren Umgebung auszuführen, sowie Docker Compose um die benötigten Services zu bündeln. Die Konfiguration für die Entwicklung enthält zusätzlich einen Editor, mit dem sich die OpenAPI Spezifikation editieren und validieren lässt, sowie eine MongoDB Instanz, mit der das Backend vollständig lokal ausgeführt werden kann. Diese Datenbank kann auch über das mongo-express Webinterface administriert werden, wenn während der Entwicklung schnelle Datenbankänderungen vorgenommen werden müssen.

Der Produktivserver nutzt hingegen einen MongoDB Atlas Cluster und läuft als Container bei dem PaaS-Dienst Heroku hinter einem Reverse Proxy.

2.3 Frontend

Das Frontend von Viva Corona ist in Kotlin geschrieben. Die Entscheidung für Kotlin begründet sich hauptsächlich in der Modernität und Ausdrucksstärke Koltins gegenüber Java. Dennoch ist Kotlin kompatibel zu Java Abhängigkeiten, der Java Standardbibliothek, sowie den Android APIs. Obwohl die Entwickler unserer Gruppe anfangs keine relevanten Kotlin-Kenntnisse besaßen, hat sich diese Entscheidung als nützlich erwiesen. Kotlin bietet einige Sprachfeatures und Syntactic Sugar, die bei der Entwicklung häufig genutzt wurden und sehr zur Geltung kamen, was für eine bessere Lesbarkeit und damit auch einer höheren Produktivität geführt hat.

Das Frontend lässt sich grob in zwei Teile aufteilen. Erstens die Definition der Interaktionsmöglichkeiten mit der App in den Activities und Fragments und deren ausgelöste Aktionen, und zweitens die passiven Aktionen, wie das empfangen von Benachrichtigungen oder Aufzeichnen der Standortdaten.

2.3.1 Interaktive Elemente

Der visuelle Teil der App gliedert sich in Activities und Fragments. Für in sich geschlossene, einzigartige Ereignisse gibt es jeweils eine eigene Activity. Diese können sowohl nur an einer Stelle auftreten, wie bspw. die `RegisterActivity`, oder auch an mehreren, wie die `LocationPickerActivity`, die einen Standort auswählt und als Ergebnis zurückgibt.

Die wichtigsten Ereignisse und Interaktionsmöglichkeiten liegen jedoch alle in der `MainActivity`. Diese ist über einen NavigationDrawer in die unterschiedlichen Themenbereiche (Handel, Infektionsstatus, Kontaktverfolgung, ...) unterteilt. Um ein einfaches und effizientes Wechseln zwischen den Bereichen zu erreichen, besteht jeder Bereich zunächst aus einem Hauptfragment. Dieses kann noch weitere, geschachtelte Fragments beinhalten, die ebenfalls ausgetauscht werden können. Durch die Modulare Architektur ist es bei der Entwicklung einfach möglich, die bestehenden Elemente zu erweitern, anzupassen oder neue Bereiche hinzuzufügen.

Die Kommunikation mit dem Backend erfolgt über die Clients. Für jeden Themenbereich, der sich auch als Teilstiel in der URL widerspiegelt, gibt es einen eigenen Client. Dieser implementiert die notwendigen REST-Funktionen, wie zum Beispiel der `InfectionApiClient` mit den Methoden `getInfectionStatus` und `postInfectionStatus`. Die Clients orientieren sich somit sehr stark am Backend. Die Ergebnisse, die sie zurückliefern, entsprechen direkt den Antworten des Servers, welche in eigene Frontend-Modellklassen übersetzt wurden.

Auf Basis dieser Modelle gibt es ViewModels für die meisten visuellen Komponenten. Diese dienen zur Hin- und Rückübersetzung der vom Server erhaltenen Modelldaten in visuell einfach darstellbare Daten. Zum Teil enthalten sie auch Informationen über den aktuellen Darstellungsstatus. Das `SearchOffersViewModel` enthält so beispielsweise alle Daten, die zu einer Angebotssuche gehören. Es beinhaltet sowohl die Details zur Abfrage (Query), wie auch die zugehörigen Ergebnisse. Außerdem ist hierüber die Kommunikation möglich, ob die Anzeige eines Angebot (und wenn ja, welches) zur Kartendarstellung oder zur Listendarstellung wechseln soll.

Die ViewModels werden letztendlich durch die Views dargestellt. Diese sind hauptsächlich Fragments und Activities, vereinzelt auch separat implementierte UI-Komponenten. Wie oben bereits erwähnt können Views an nur einer Stelle auftreten, oder auch mehrfach verwendet werden. So wird zum Beispiel das `InfectionStatusBaseFragment`, das einen Infektionsstatus visualisiert, mehrfach verwendet. Zum einen zeigt es die Daten des gescannten QR-Codes mit einem neuen Status an, der dann hochgeladen werden kann. Zum anderen wird es verwendet, um den aktuellen Status darzustellen, der auf dem Server gespeichert ist. Das Fragment besteht wiederum selbst aus kleineren Einheiten, den `InfectionStatusDataItems`, die für einzelne Attribute (Status, Zeitpunkt, etc.) zuständig sind. Ein wichtiges Werkzeug für diese Wiederverwendbarkeit ist das oft eingesetzte Prinzip des Data Binding. Dabei lassen sich die durch die View dargestellten Daten durch ViewModels einfach und übersichtlich anpassen, denn sobald sich das ViewModel ändert, ändert sich direkt auch die Darstellung selbst.

Die Änderung der ViewModels wird durch die Views selbst initiiert. Durch bestimmte Aktionen, wie das Drücken eines Knopfes, ruft die View die zugehörige Methode des entsprechenden Clients auf. Dieser liefert asynchron das Ergebnis, das die View dann an das ViewModel weitergibt, wodurch die bereits erwähnten Aktualisierungen entstehen.

Für komplexe Datentypen in den ViewModels werden häufig `LiveData`-Objekte verwendet, auf die dann das Observer-Pattern angewendet wird. Der Grund hierfür ist die notwendige Verarbeitung vor der Verwendung. Als Beispiel ist die Anzeige der verfügbaren Waren zu nennen. Es gibt hier eine Liste, die bereits die entsprechenden ViewModels für jedes einzelne Angebot enthält. Da die Darstellung aber mittels eines `RecyclerViews` mit entsprechendem `ListAdapter` geschieht, muss die aktualisierte Liste bei einer Änderung an den Adapter weitergereicht werden. Außerdem wurde Live Data auch verwendet, um eine Kommunikation zwischen einzelnen Views zu ermöglichen, wie bereits bei dem Wechsel zwischen Listen- und Kartenansicht der Suchergebnisse erwähnt. Die Kombination der Verwendung von Data Binding und Live Data vereinfacht die Aktualisierung der Benutzeroberfläche somit stark.

Um die App weiterhin simpel zu halten und die Logik komplett im Backend zu halten, aktualisieren wir die Daten nie nur auf Seiten des Nutzers (Clients). Dadurch erreichen wir eine konsistente Darstellung der wirklich gespeicherten Daten, wenngleich die Kommunikation mit dem Server häufiger stattfindet. Da die App insgesamt einen eher geringen Internetdatenbedarf hat, überwiegen die Vorteile der Konsistenz, und damit auch der User Experience. So wird beispielsweise bei der Erstellung eines neuen Angebots nicht das neu erstellte Angebot direkt auf Clientseite hinzugefügt. Nach der POST-Anfrage wird dann vielmehr eine erneute GET-Anfrage für alle Angebote gestellt.

2.3.2 Hintergrunddienste

Das Aufzeichnen der Standortdaten erfolgt über den `LocationTrackingService`. Dieser läuft als sogenannter *Foregroundservice*, um das Aufzeichnen des Standorts auch nach Schließen der App zu gewährleisten. Dies wird dem Nutzer permanent über eine Benachrichtigung in der Statusleiste angezeigt. Dadurch ist man sich als Benutzer jederzeit bewusst, dass der eigene Standort aufgezeichnet wird.

Um die Standorte zu speichern bevor sie an den Server hochgeladen werden, wird eine Room Database verwendet. Dies ist eine SQLite Datenbank. Da nicht davon ausgegangen werden kann, dass Standortdaten direkt nach dem Aufzeichnen hochgeladen werden können, ist eine Datenbank hier besser als die Standorte in den Shared Preferences zwischenzuspeichern. Da die Room Datenbank kein rohes SQL verlangt, können Datenbankstruktur und Abfragen direkt in Kotlin definiert werden. Um Standorte zu speichern, wird die Entität `DBLocation` verwendet. Diese speichert Zeit, Längen- und Breitengrad. Um auf die Daten wieder zugreifen zu können, gibt es sogenannte DAOs oder Data Access Objects. Hier sind die Abfragen für die Datenbank definiert. Für die Locations sind dies `getLocations`, um alle Standorte abzurufen, `addLocation`, um einen neuen Standort hinzuzufügen, und `deleteLocations`, die übergebenen Einträge zu löschen.

Das Hochladen der Daten wird über einen periodischen erstellten Service durchgeführt, den `UploadService`. Dieser wird über einen periodischen Intent ca. alle 10 Sekunden gestartet. Dies wird vom Android internen AlarmManager verwaltet. Werden Standortdaten auf den Server hochgeladen, so werden diese bei einem erfolgreichen Upload aus der lokalen Datenbank gelöscht, um keine inkonsistenten Zustände zu haben.

Ein weiterer Service, der im Hintergrund arbeitet, ist der `WebSocketService`. Dieser hält eine Verbindung zum Server aufrecht, um es dem Server zu ermöglichen Push-Benachrichtigungen an den Client zu schicken. Zu den Benachrichtigungen gehören zum Beispiel Warnung vor Kontakt mit infizierten Personen, oder passende Produktangebote für die eigenen Produktbedürfnisse. Der `WebSocketService` enthält dabei einen `WebSocketListener`, um auf Benachrichtigungen des Servers zu reagieren. Hierfür wurde die `okHttp3` Library verwendet. Falls die Verbindung des WebSockets unterbrochen wird, gibt es zusätzlich den `WebSocketReconnectService`. Dies ist wieder ein über den `AlarmManager` gesteuerter, periodischer Service, der alle 10 Sekunden versucht erneut eine Verbindung aufzubauen.

Damit Benachrichtigungen des Servers nicht nur beim `WebSocketService` ankommen, sondern auch als Android Benachrichtigungen dem Benutzer angezeigt werden, müssen die Server Benachrichtigungen zu Android Benachrichtigungen umgewandelt werden. Diese können mit dem `NotificationHelper` erzeugt werden. Ab Anroid SDK 26 gibt es zusätzlich die Möglichkeit Benachrichtigungen auf Channels auzuteilen, um dem Benutzer mehr Flexibilität bei den Einstellungen der Benachrichtigung zu bieten. Daher nutzen wir die vier Channels Location Tracking, Infected Notification, Product Notification und Quiz Notification. Der Location Tracking Channel gibt die Benachrichtigung für den `LocationTrackingService` aus. Der Infected Notification Channel erstellt Benachrichtigungen, um den Nutzer auf Kontakte mit Infizierten hinzuweisen. Der Product Notification Channel benachrichtigt den User über passende Produktangebote. Der Quiz Notification Channel sendet Benachrichtigungen, wenn sich der Zustand eines Quizes ändert.

Um die oben beschriebenen Services nach Hochfahren des Gerätes zu aktivieren (sofern die notwendigen Berechtigungen vorhanden sind), gibt es den `AlarmBroadcastReceiver`. Registriert dieser ein Boot Ereignis, so werden der periodische Standortupload, der `WebSocketService`, sowie der `LocationTrackingService` gestartet.

3 Benutzerhandbuch

Die Funktionen der App unterteilen sich in Features für normale Nutzer (User, s. 3.1) und privilegierte Nutzer (Admin, s. 3.2). Zudem gibt es administrative Aufgaben (s. 3.3), die planmäßig sehr selten durchgeführt werden und daher lediglich über alternative Methoden, wie bspw. HTTP-Requests oder swagger-ui erreichbar sind. In einer Produktivinstallation können für die administrativen Aufgaben noch zusätzliche Zugriffsbeschränkungen hinzugefügt werden, wie beispielsweise ausschließlicher Zugriff über das lokale Netzwerk (localhost).

3.1 User

3.1.1 Login und Registrierung

Beim ersten Öffnen der App müssen die Nutzer sich entweder registrieren durch die Angabe eines Passwortes oder mit einem bestehenden Account einloggen, dafür wird die sogenannte *User ID* benötigt. Die User ID kann bei der Registrierung nicht frei gewählt werden, sondern wird automatisch generiert und zugewiesen. Nach der Registrierung besteht die Möglichkeit, sich die zugewiesene ID per E-Mail zuzusenden. Die Nutzer-ID kann später über den Menüpunkt *Check Status* (s. 3.1.9) abgefragt werden. Nach dem erfolgreichen Login sind nur User-Funktionalitäten verfügbar. Um Adminrechte zu erhalten und die zusätzlichen Funktionen nutzen zu können, müssen die in Kapitel 3.2.1 entsprechenden Schritte durchgeführt werden.

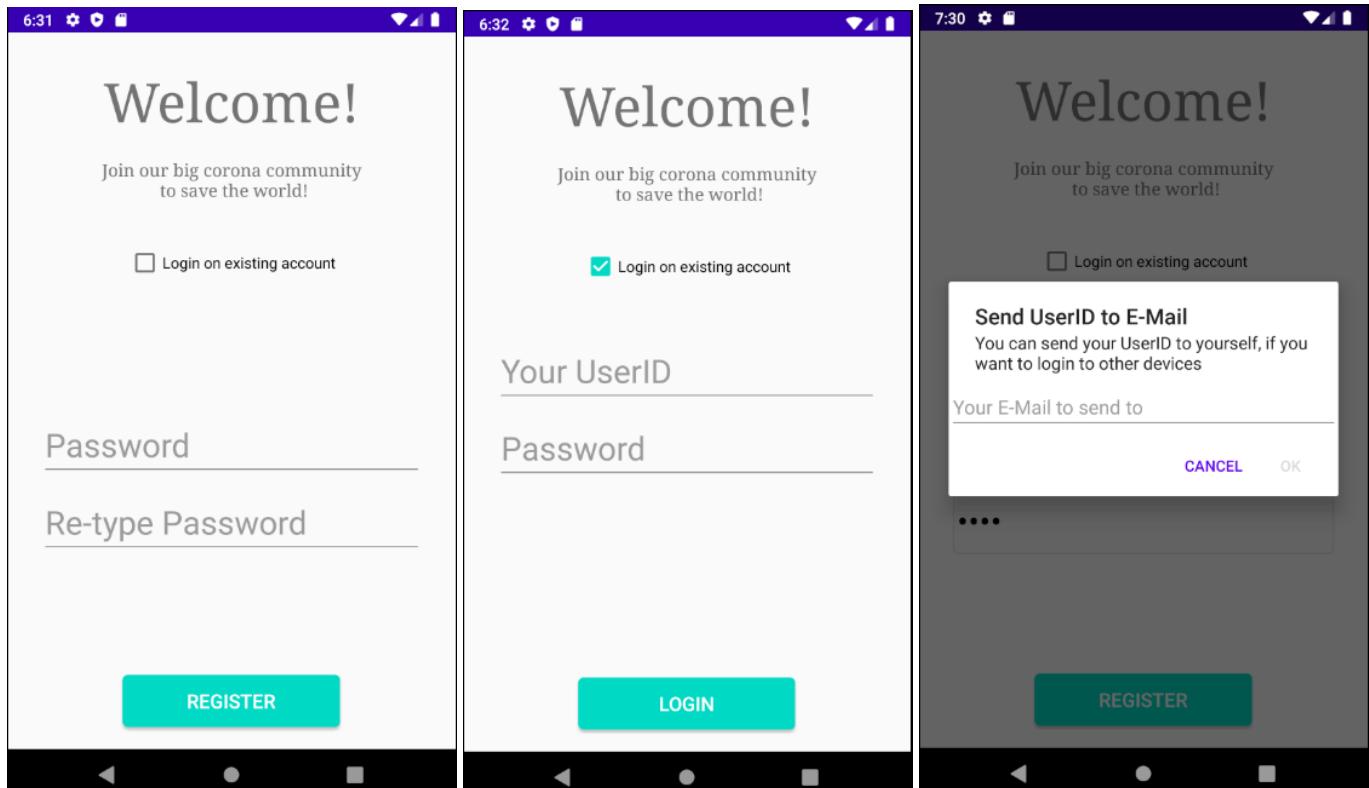


Abbildung 1: Von links nach Rechts: Registrierungsansicht, Anmeldeansicht, Senden der UserID per Email

3.1.2 Standortverlauf

Nach dem Login wird zunächst die Berechtigung eingefordert, auf den Standort zuzugreifen. Diese Berechtigung ist essenziell für die Kontaktverfolgung und ist damit Grundvoraussetzung für den Hauptnutzen von Viva Corona. Ab Erhalt der Berechtigung wird der Standort bei einer Standortveränderung um 15 Meter oder nach 30 Sekunden aufgezeichnet. Zum Aufzeichnen des Standorts muss die App nicht geöffnet sein. Die Standortaufzeichnung wird nur durch Beenden der App (über die Android Einstellungen) angehalten und fortgeführt, sobald sie wieder gestartet wurde. Das Aufzeichnen des Standorts wird durch eine dauerhafte Benachrichtigung signalisiert.

Der aufgezeichnete Standort wird in 10 Sekunden Abschnitten hochgeladen. Auch dies geschieht automatisch im Hintergrund, sofern eine Internetverbindung besteht. Ist dies nicht der Fall, so werden die Standorte erst mit der nächsten Welle hochgeladen.

Den aufgezeichneten Standort kann man sich dann unter dem Menüpunkt *History* anschauen. Der farbliche Verlauf der Linien impliziert den zeitlichen Verlauf, wobei der Anfangs- und Endpunkt jeweils markiert ist. Je bläulicher eine Wegstrecke ist, desto früher war sie und je rötlicher, desto später.

Der angezeigte Verlauf lässt sich über den linken Knopf zeitlich einschränken, sodass nur der Standortverlauf eines Zeitraums angezeigt wird. Dabei sind die ausgewählten Tage jeweils inklusive. Über den rechten Knopf lässt sich der gewählte Zeitraum auf den Default-Wert (nur heute) zurücksetzen.

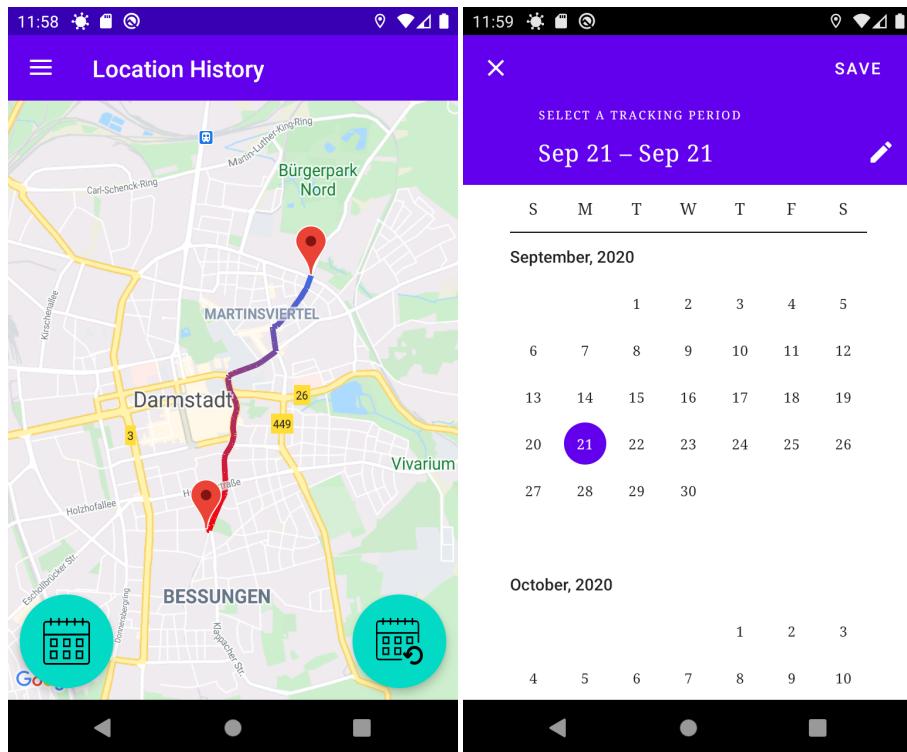


Abbildung 2: Links ist der Standortverlauf zu sehen und rechts die Auswahl des Zeitraums

3.1.3 Infektionsstatus

Über den Menüpunkt *Infection Status* lässt sich der aktuelle Infektionsstatus einsehen. Hat man noch keinen Test absolviert oder das Ergebnis noch nicht eingescannt, so ist der Status unbekannt (*Unknown*). *Approved at* beinhaltet den Zeitpunkt des bestätigenden Tests, *Approximately at* bezieht sich auf Ihre Schätzung der Ansteckung, falls Sie eine Vermutung haben.

Durch Betätigen des Knopfes am unteren Bildschirmrand lässt sich ein QR-Code einscannen, der zuvor von einer offiziellen Behörde ausgestellt worden sein muss. Der QR-Code muss in das angezeigte Fenster passen, damit er eingelesen werden kann. Anschließend werden die Inhalte des QR-Codes angezeigt. Wenn diese richtig sind, können die Daten über den entsprechenden Knopf hochgeladen werden. Sollte der QR-Code falsche Daten enthalten, wenden Sie sich bitte an den Aussteller.²

Nach dem erfolgreichen Hochladen des neuen Status wird dieser in der Übersicht angezeigt. Falls nun der Status *Infected* vorliegt, ermittelt der Server automatisch alle Personen, mit denen seit der Infektion Kontakt bestand. Als Referenz wird die Schätzung für den Infizierungszeitpunkt verwendet, falls dieser angegeben wurde. Andernfalls wird der Zeitraum ab zwei Wochen vor der bestätigten Infektion berücksichtigt. Die ermittelten Personen werden durch eine Push-Benachrichtigung gewarnt. Dadurch kann die Verbreitung weiter eingedämmt werden!

²In unserem Szenario generieren wir die Testergebnisse natürlich selbst über den QR-Code Generator (s. 3.3.4)

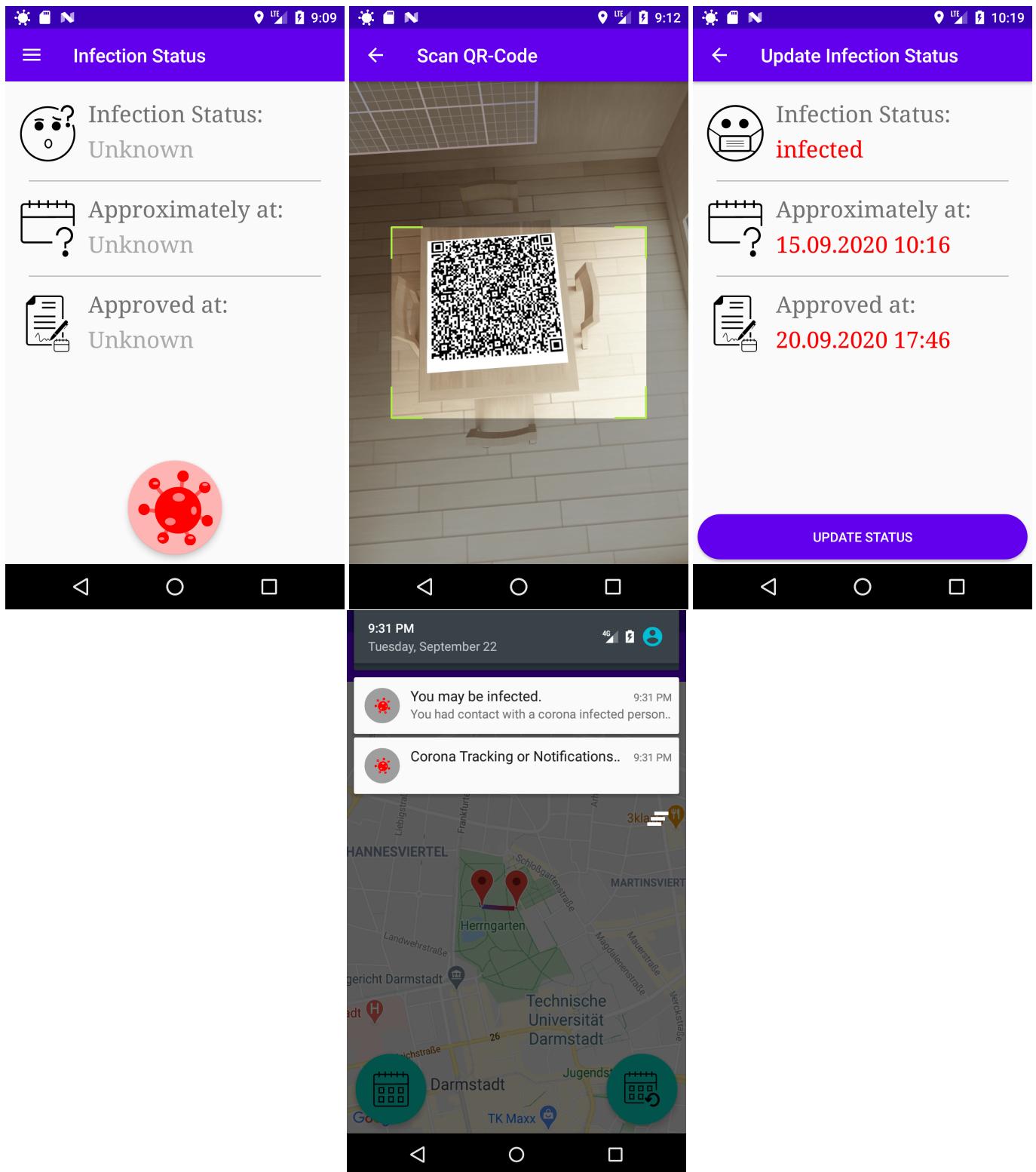


Abbildung 3: Von Links nach Rechts sind die Schritte zu sehen, wie ein Nutzer seinen Infektionsstatus verändert

3.1.4 Handel - Angebote

Um die Versorgung von raren Gütern zu gewährleisten, gibt es eine Handelsplattform unter dem Menüpunkt *Trading*. Diese ist unterteilt in die eigenen Angebote von Waren, der Suche nach Angeboten und Waren, der Anmeldung eines Gesuchs und einer Übersicht für Supermarktangebote. Die eigenen Angebote können unter dem Menüpunkt *My Offers* am linken, unteren Bildschirmrand eingesehen werden.

Hier besteht durch Betätigen des "+"-Knopf zunächst die Möglichkeit, eigene Waren anzubieten. Es müssen hier mindestens der Produktnname *Product* und *Category* angegeben werden, alles andere ist optional. Es empfiehlt sich jedoch, so viel wie möglich auszufüllen, um unnötige Nachfragen zu vermeiden. Der eigene Standort, wo die Ware abgeholt werden kann, kann über die Standortnadel angegeben werden. Hier genügt ein einfaches Drücken auf die Karte, um einen neuen Standort auszuwählen. Falls kein Preis eingetragen wird, wird das Angebot mit 0€ erstellt. Über die Menge kann spezifiziert werden, wie viel von dem Produkt verfügbar ist. Dies bezieht sich auf die Stückzahl, also bspw. 5 für 5 Packungen Mehl von 1kg. Die Menge wird standardmäßig mit 1 angenommen. Die Telefonnummer dient zur weiteren Kommunikation mit potenziellen Käufern und ist für jeden einsehbar. Über das Freitextfeld *Details* können zusätzliche Informationen angegeben werden.

Änderungen eines Angebots können vollzogen werden, in dem das entsprechende Angebot in der Übersicht zunächst ausgeklappt (angetippt) wird. Über den Stift gelangt man dann zum Bildschirm für die Bearbeitung. Nach abgeschlossener Bearbeitung wird dann die Angebotsliste automatisch aktualisiert.

Angebote können in der Übersicht über das rote Kreuz gelöscht werden. Wenn es mithilfe der App verkauft wurde, kann der Verkäufer angeben, dass das Produkt verkauft wurde, um Achievements zu erhalten.

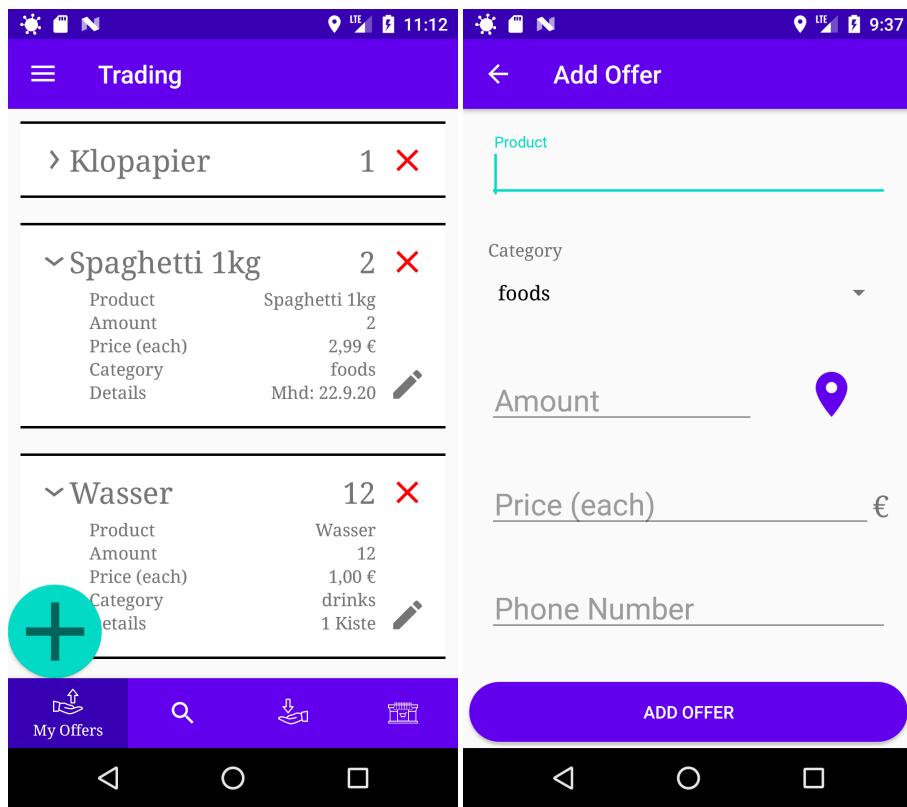


Abbildung 4: Von Links nach Rechts: Listenübersicht der eigenen Angebote, Detailansicht der Angebot Erstellung

3.1.5 Handel - Angebotssuche

Über *Search*, den 2. Menüpunkt in der unteren Menüleiste des *Trading*, können Nutzer zur Angebotssuche gelangen. Über die Suchleiste kann ein Produktnname angegeben werden. Durch Betätigen des Suchicons auf der Tastatur unten rechts wird die Suche ausgeführt. Die Suchanfrage kann weiterhin über das Filtersymbol oben rechts durch zusätzliche Filter verfeinert werden. Die Ergebnisse müssen dann allen Filtern gerecht werden. Besonders interessant ist hier der Standortfilter. Durch Tippen auf die Standortnadel bei den Filteroptionen kann ein Standort auf der Karte mit einfachem Tippen ausgewählt werden. Zusätzlich kann ein Radius angegeben werden. Ohne Wahl des Radius (0km) werden lediglich die Entferungen des Angebots zum gewählten Standort in der Ergebnisliste angezeigt.

Die Ergebnisse werden in einer Liste angezeigt. Um ein Angebot auf der Karte einzusehen, kann mithilfe der jeweiligen Standortnadel zur Kartenansicht gewechselt werden. Das interessante Angebot wird in diesem Fall blau markiert. Um zurück zur Listenansicht zu gelangen, kann erneut auf das Angebot gedrückt werden. Sollten sich mehrere Angebote nahe beieinander befinden, werden diese zusammengefasst. Diese können durch heranzoomen aufgelöst werden. Alternativ kann auch auf den Cluster geklickt werden, wodurch die gruppierten Angebote angezeigt werden. Nach Auswahl eines Angebots gelangt man zur Listenansicht.

Falls Interesse an einem Angebot besteht, kann der Verkäufer über den "Call"-Button angerufen werden. Anrufe laufen über die normale Telefonapp, es fallen die üblichen Gebühren des Mobilfunkanbieters an.

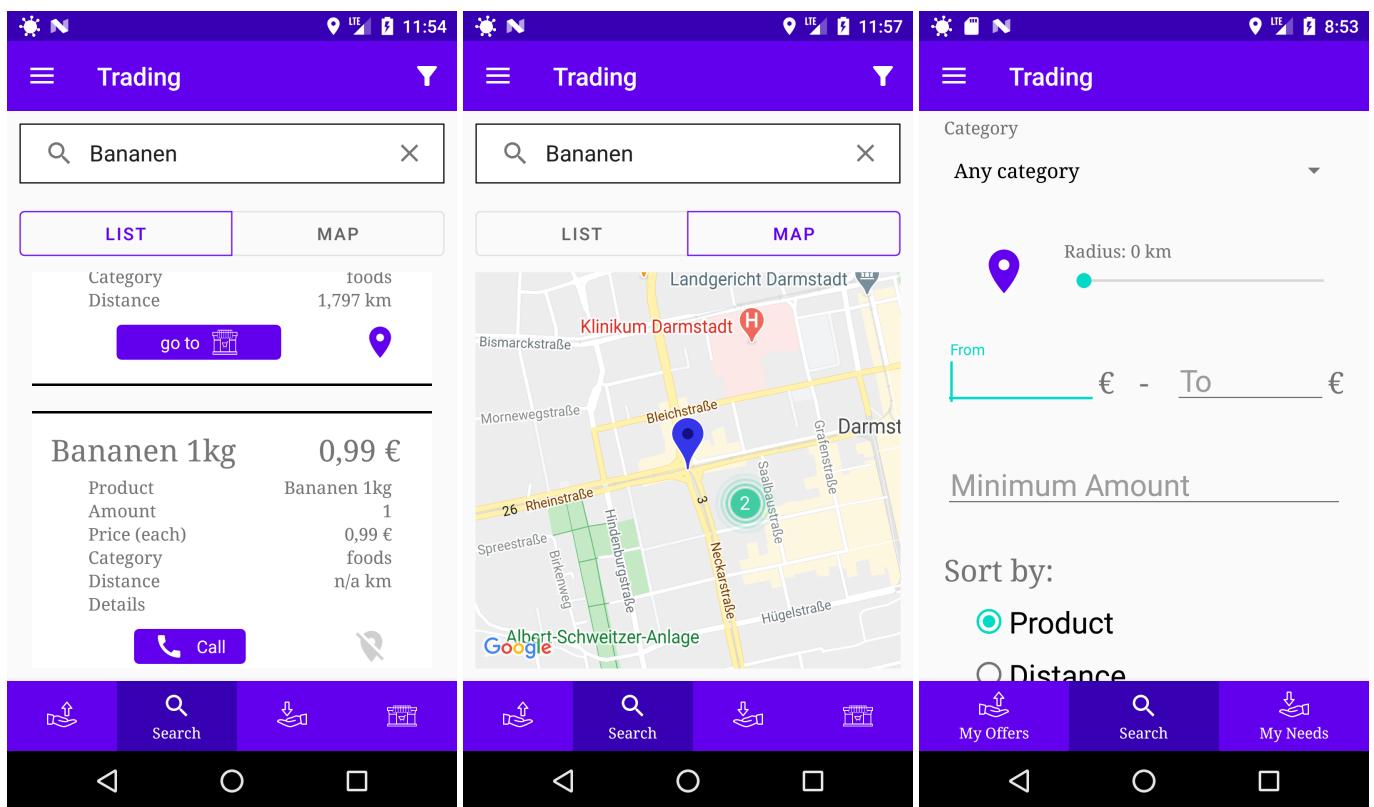


Abbildung 5: Von links nach rechts: Produktsuche mit Listenansicht, Kartenansicht für gefundene Produkte, Filterung für die Suche

3.1.6 Handel - Warengesuch

Falls eine Ware über die Suchfunktion nicht gefunden wurde und dennoch dringend benötigt wird, gibt es die Möglichkeit, unter dem 3. Menüpunkt der unteren Leiste, *My Needs*, ein Warengesuch anzumelden. Nach Anmelden eines Warengesuchs erhält man eine Benachrichtigung, sobald es Angebote gibt, zu denen das Gesuch passt. Durch Antippen der entsprechenden Benachrichtigung gelangt man zur Suche, die zu dem Gesuch passt und unter der man die passenden Angebote findet. Dort können dann weitere Schritte unternommen werden, um die entsprechende Ware zu erhalten (z.B. Ansehen der passenden Angebote, Kontaktieren des Verkäufers, etc.).

Wenn ein Gesuch erfüllt wurde oder der Bedarf nicht mehr besteht, kann es über das rote Kreuz in der Bedarfsliste entfernt werden. Ab sofort werden keine Benachrichtigungen für dieses Gesuch versendet. Um entsprechende Achievements zu erhalten, sollte über das Popup angegeben werden, ob das Gesuch erfüllt wurde oder nicht.

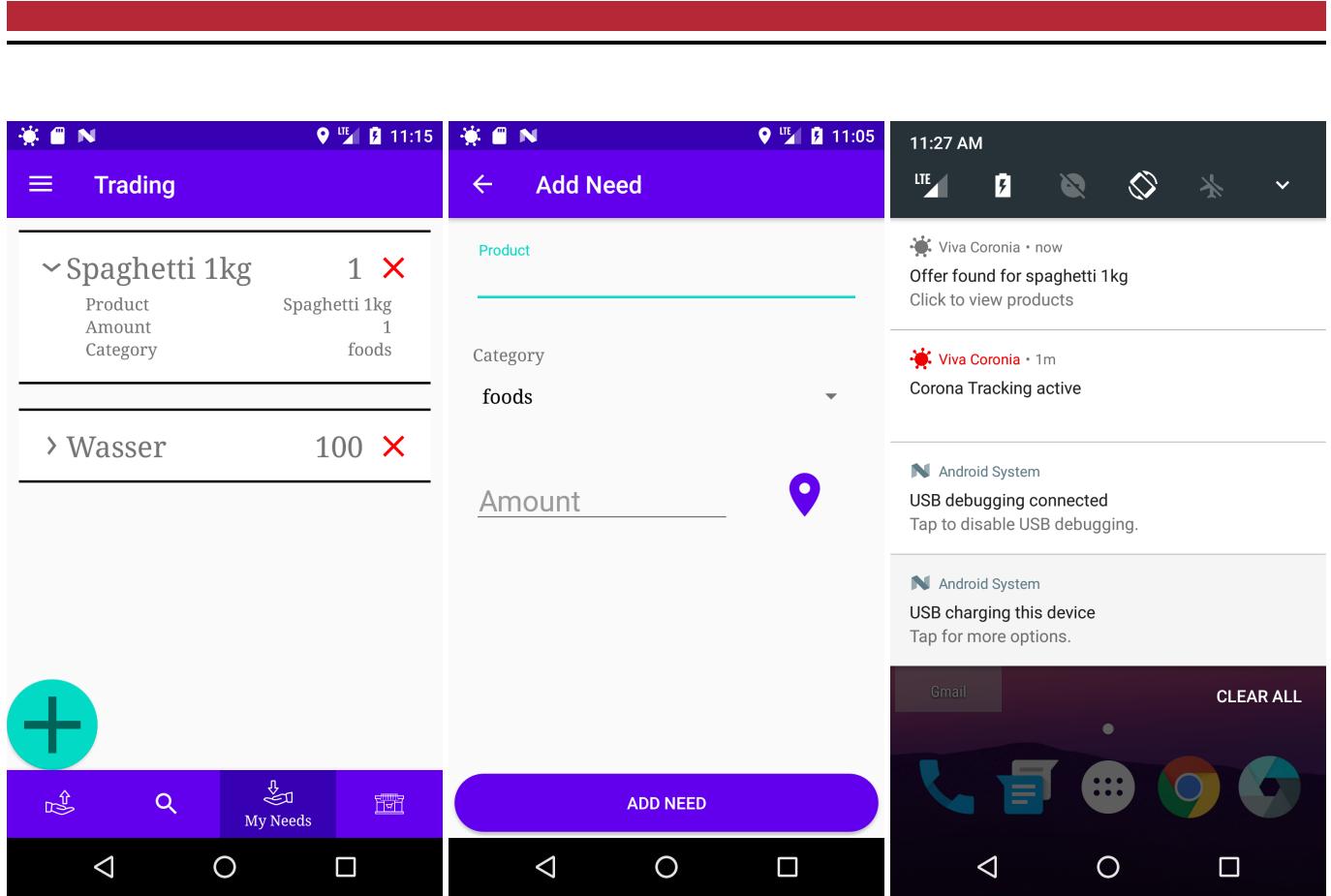


Abbildung 6: Von Links nach Rechts: Listenansicht der eigenen Gesuche, Erstellen eines Gesuchs, erhalten einer Benachrichtigung für ein passendes Produkt

3.1.7 Handel - Supermärkte

Um nicht nur private, sondern auch kommerzielle Angebote zu unterstützen, gibt es zudem einen 4. Menüpunkt namens *Supermarkets*. Hier ist eine Karte zu sehen, auf der nach einem langen Click verfügbare Supermärkte angezeigt werden. Sind mehrere Supermärkte eng beieinander, so werden diese wie bei der Angebotssuche zusammengefasst. Durch heranzoomen werden diese aufgelöst, und die Märkte sind einzeln erkennbar.

Durch Klick auf den Cluster werden die verfügbaren Supermärkte in diesem Bereich in einer Liste angezeigt und sind einzeln auswählbar. Ist der Supermarkt schon einzeln durch einen roten Marker vorhanden, wird durch Klick auf diesen ein Pop-up über diesem geöffnet, mit dem zu der Supermarktubersicht navigiert werden kann.

Wird ein Supermarkt ausgewählt, der noch keine Produkte enthält, so öffnet sich ein Popup-Menü, welches fragt, ob eine neuen Produkt angelegt werden soll. Sind schon Produkte vorhanden, wird direkt zur Supermarktubersicht gewechselt. Angelegte Produkte sind in einer Liste auffindbar. Für jedes Produkt kann die Verfügbarkeit direkt geändert werden. Optionen sind: Nicht vorhanden, wenig vorhanden, genug vorhanden und viel vorhanden.

Durch den Knopf unten links können zudem neue Produkte für diesen Supermarkt hinzugefügt werden. Hierfür gibt man einen Produktnamen, die Kategorie und eine Verfügbarkeit an. Die Verfügbarkeit kann nach eigenem Ermessen eingetragen.

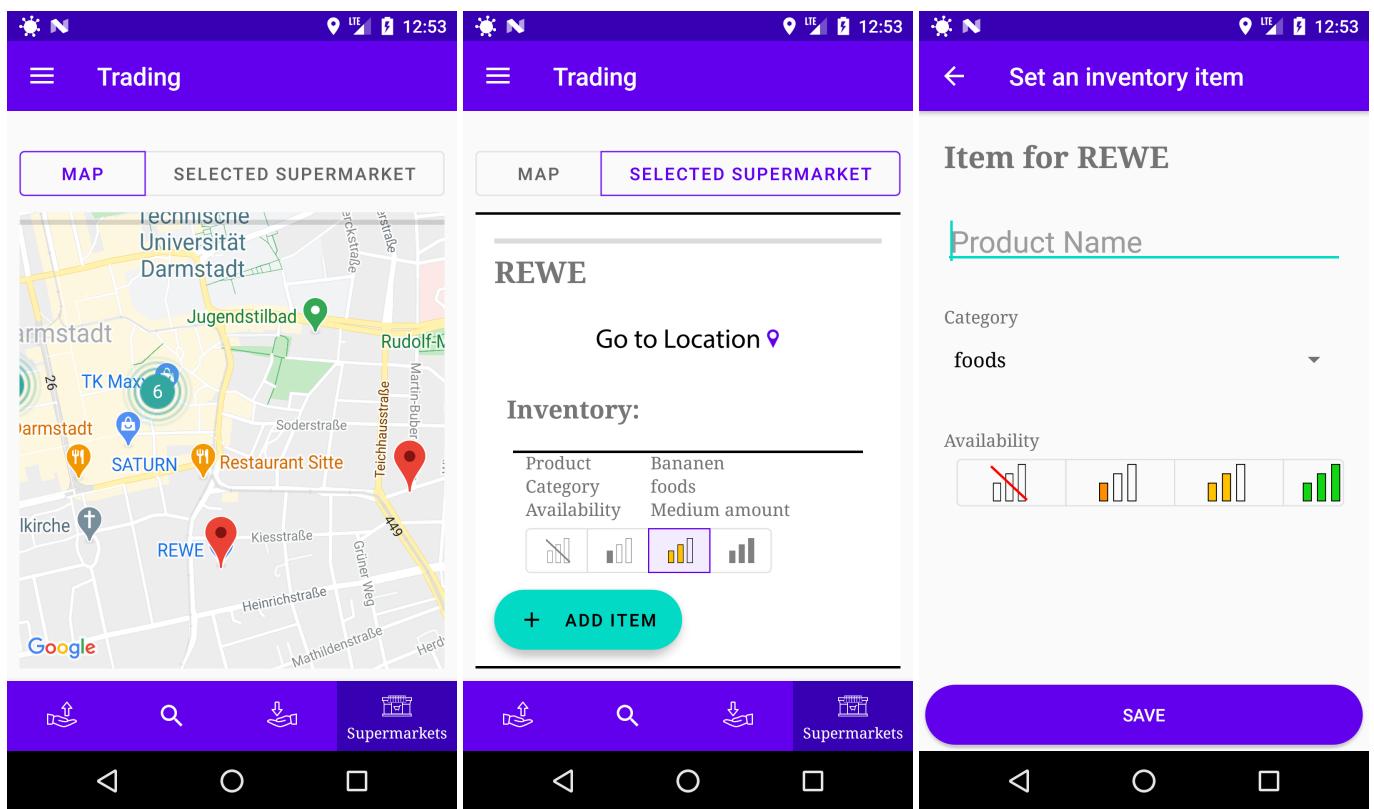


Abbildung 7: Von links nach rechts: Kartenübersicht der Supermärkte, Inventarliste eines Supermarkts, Erstellen eines neuen Produkts

3.1.8 Achievements

Um besonderes Verhalten eines Nutzers festzuhalten, kann er im Laufe der Benutzung der App sogenannte *Achievements* sammeln. Diese lassen sich unter dem gleichnamigen Menüpunkt anzeigen. Dabei gibt es insgesamt sechs verschiedene zu erreichende Abzeichen, die sich in drei verschiedene Abstufungen unterteilen lassen. Für ein Abzeichen ist es möglich, dieses in Bronze, Silber oder Gold zu erreichen. Sobald eine neue Stufe erreicht wurde, wird der Nutzer über eine Benachrichtigung darüber informiert und durch einfachen Klick direkt auf die Seite der Abzeichen weitergeleitet. Damit ein Abzeichen aktualisiert werden kann, muss eine Verbindung zum Server bestehen.

Bereits erreichte Abzeichen werden dabei in voller Farbe angezeigt, wohingegen noch nicht erreichte Abzeichen ausgegraut dargestellt werden. Neben jeder Kategorie wird durch Klick auf das Fragezeichen-Symbol ein kleines Dialogfenster geöffnet, in welchem eine genauere Erklärung des Abzeichens angezeigt wird. Außerdem wird dann auch angezeigt, was für die nächste Stufe des jeweiligen Abzeichens noch erfüllt werden muss und wie hoch der prozentuale Anteil der User ist, die das Abzeichen ebenfalls erreicht haben.

Neben diesen Abzeichen wird in der Sektion Achievements dem Nutzer ein *Infection Score* angezeigt, welche dem Nutzer suggeriert, wie wahrscheinlich eine Ansteckung ist. Dabei liegt der Wert zwischen 0 und 100, wobei ein hoher Wert eine hohe Infektionswahrscheinlichkeit und ein niedriger Wert eine niedrige Infektionswahrscheinlichkeit ausdrückt. Die Berechnung dieses Wertes ist abhängig von der Häufigkeit der Kontakte mit infizierten Menschen. Falls der Nutzer in der App angegeben hat, dass er infiziert ist, liegt dieser Score bei 100. Hier kann ebenfalls durch Klick auf das Fragezeichensymbol eine genauere Erklärung zum Infection Score abgefragt werden.

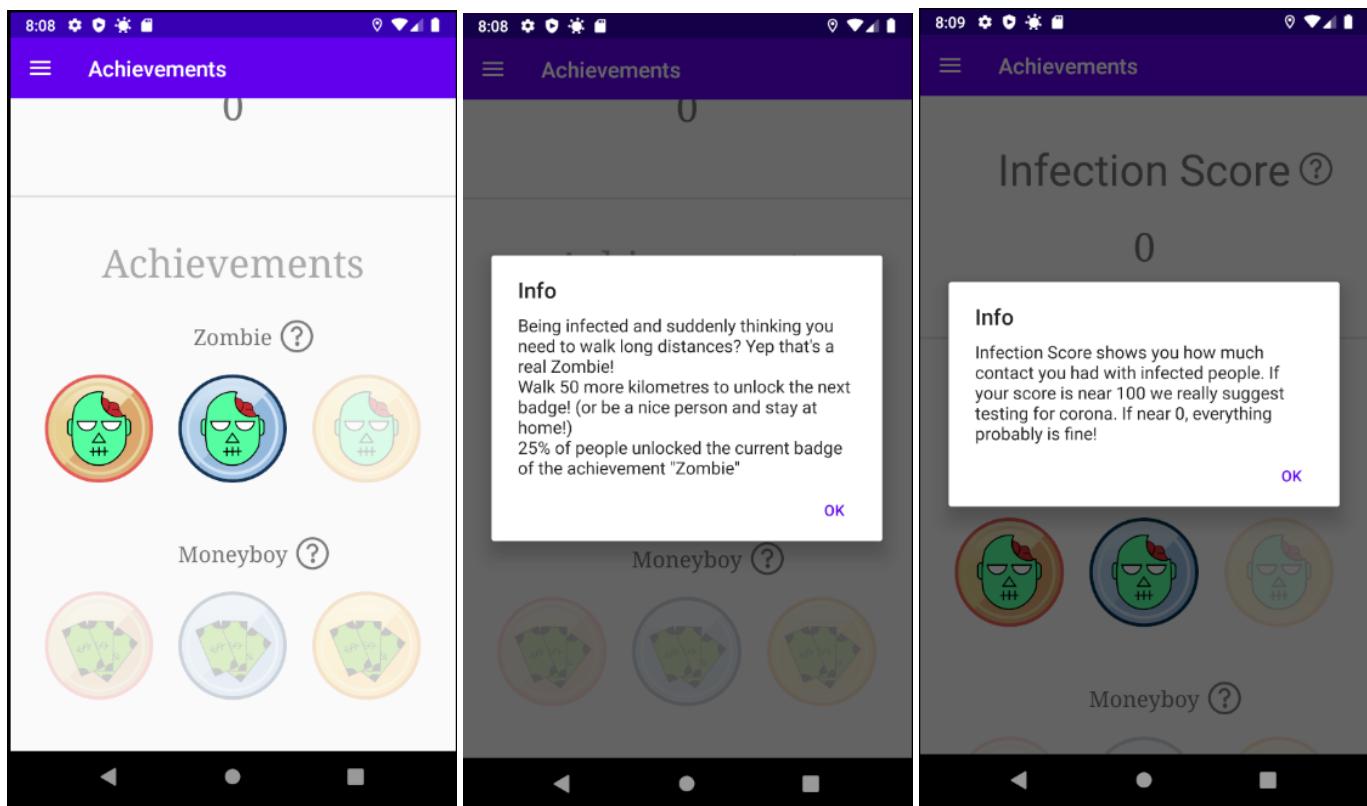


Abbildung 8: Von links nach rechts: Beispielhafte Ansicht von freigeschaltenen Abzeichen, Info zu dem Zombie Achievement, Infection Score und Info dazu

3.1.9 Check Status

Unter dem Menüpunkt Check Status ist der aktuelle Status einsehbar.

Dieser kann entweder *User* oder *Admin* sein und über den Knopf *Reload Status* aktualisiert werden, wofür jedoch eine Internetverbindung benötigt wird. Bei einer Anhebung des Status als Admin erscheint ein Dialog, mit dem die Admin-Funktionen aktiviert werden können durch erneutes Einloggen des Users(s. 3.2.1).

Bei einer Widerrufung der Adminrechte des Nutzers wird der User beim Nutzen von Admin-Funktionen auf den Check Status Menüpunkt weitergeleitet. Zudem kann hier die Nutzer-ID über *Show UserID* angezeigt werden. Sie wird zusätzlich in die Zwischenablage kopiert, sodass sie an anderer Stelle, bspw. in einem Passwortmanager, eingefügt oder eingetragen werden kann.

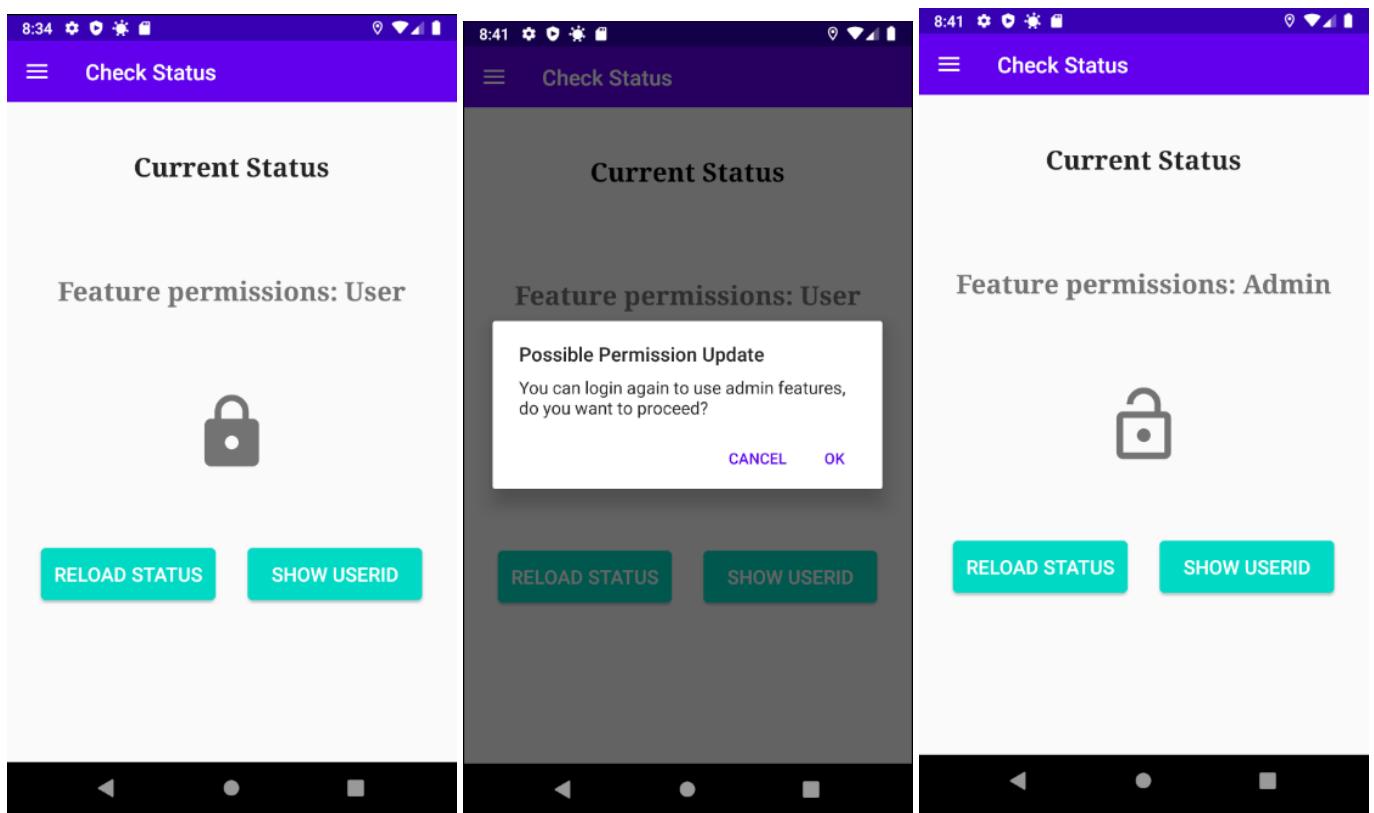


Abbildung 9: Von links nach rechts: Der Status eines normalen Nutzers, der Dialog zur Möglichkeit die Rechte zu Adminrechten zu aktualisieren, aufgewerteter Status zu einem Admin

3.1.10 Quiz

Unter dem Menüpunkt Quiz kann gegen anderen Nutzer ein Quiz gespielt werden, das Fragen in Bezug auf Corona enthält. In der Übersicht sind die aktuellen Spiele, so wie die letzten 5 beendeten zu sehen. Die unteren 4 Balken einer Übersicht für ein Quiz zeigen das Verhältnis pro Frage an. Ist der Balken orange, haben beide Spieler die Frage richtig oder beide falsch beantwortet.

Sie erhalten eine Benachrichtigung, wenn:

- ein Spieler Sie zu einem Quiz herausfordert
- es ihr Zug in einem Spiel ist
- das Spiel vorbei ist.

Um auch beim Quiz anonym zu bleiben, werden zufällig Namen für jeden Spieler generiert. Dieser ändert sich aber nicht, sodass man bei gleichem Namen auch potenziell gegen den gleichen Spieler spielt. Es kann nur gegen einen zufälligen Spieler gespielt werden, der gerade online ist. Falls gerade niemand online ist, kann kein Spiel gestartet werden.

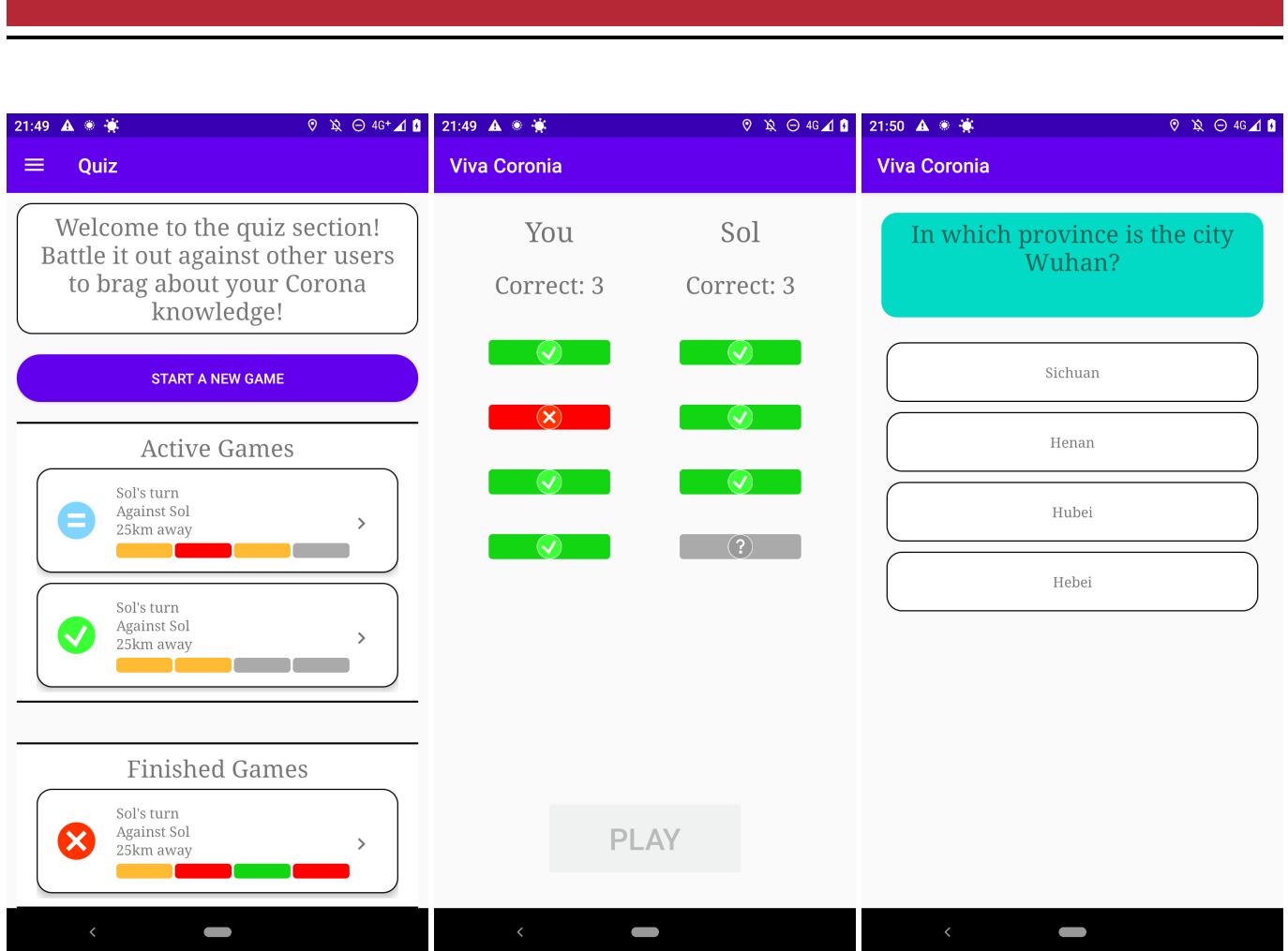


Abbildung 10: Von Links nach Rechts: Überblick über die Quizspiele, Details zu einem bestimmten Quiz, Spielen einer Quizfrage

3.2 Admin

Einige Funktionen der App sind nur für Nutzer mit erweiterten Rechten (Admins) freigeschaltet. Diese sollten nur benannt werden, wenn sie vertrauensvoll und diskret sind, da Admins Zugriff auf sensible Daten erhalten. Auch wenn die Daten nicht personalisiert sind, so geben sie doch Einsicht über Bewegungsprofile und potenziell Aufenthaltsorte von App-Nutzern!

3.2.1 Admin Promotion

Um Admin zu werden, wird zunächst die Nutzer-ID benötigt (s. 3.1.9). Diese muss an den Betreiber der App auf alternativem Wege übermittelt werden (z.B. per E-Mail). Nachdem dieser die neuen Rechte erteilt hat (s. 3.3.3), kann über den Menüpunkt *Check Status* der neue Status abgefragt werden. Wenn die Rechte erteilt wurden, erscheint nun ein Dialog, über den erneut das Passwort eingegeben werden muss. Nur nach erfolgreichem Login mittels des Passworts ändert sich der Status des Nutzers auf Admin. Ab sofort ist der Zugriff auf die erweiterten Funktionen möglich. Sollten die Rechte widerrufen worden sein, ist der Zugriff auf die Funktionen nicht mehr möglich. Eine Aktualisierung der Rechte führt ebenfalls zu einer Aktualisierung des angezeigten Menüs für den Nutzer, abhängig von seinen derzeitigen Rechten. Falls schon vor dem ersten Login auf einem Gerät Adminrechte erteilt wurden oder sich der Nutzer nach einem Logout mit dem bestehenden Account erneut einloggt, muss für Nutzung von Admin-Aktivitäten der Status aktualisiert werden. (s.o.).

3.2.2 Spreadmap

Nach Erhalt der Adminrechte und Aktualisieren des Status wird im linken Menü der zusätzliche Menüpunkt *Check Status* angezeigt. Über diesen kann Einblick in die Verbreitung des Virus gelangen. Dazu muss ein Bereich auf der Karte ausgewählt und ein Radius durch den Slider am unteren Bildschirmrand festgelegt werden. Zudem sollte ein Zeitfenster über die Datumsauswahl oben

links bestimmt werden. Bei zu großem Zeitfenster oder zu großem Radius kann es passieren, dass zu viele Daten übermittelt werden. Beide Faktoren sollten daher zu Beginn möglichst klein sein, und dann schrittweise erhöht oder verschoben werden!

Auf der Karte erscheinen nun die Bewegungen aller infizierten Personen als rote Linien. Personen, die Kontakt mit diesen hatten, werden zunächst mit bunten Linien markiert, die dann ab dem Kontaktzeitpunkt in rote Linien übergehen. Wenn auf der Karte keine Linien erscheinen, so hat sich dort im ausgewählten Zeitraum niemand mit einer Infektion aufgehalten.

Es werden prinzipiell nur direkte Kontakte ersten Grades angezeigt. Kontakte von nur möglicherweise Infizierten werden nicht berücksichtigt, wobei möglicherweise infizierte Personen solche sind, die Kontakt zu Infizierten hatten.

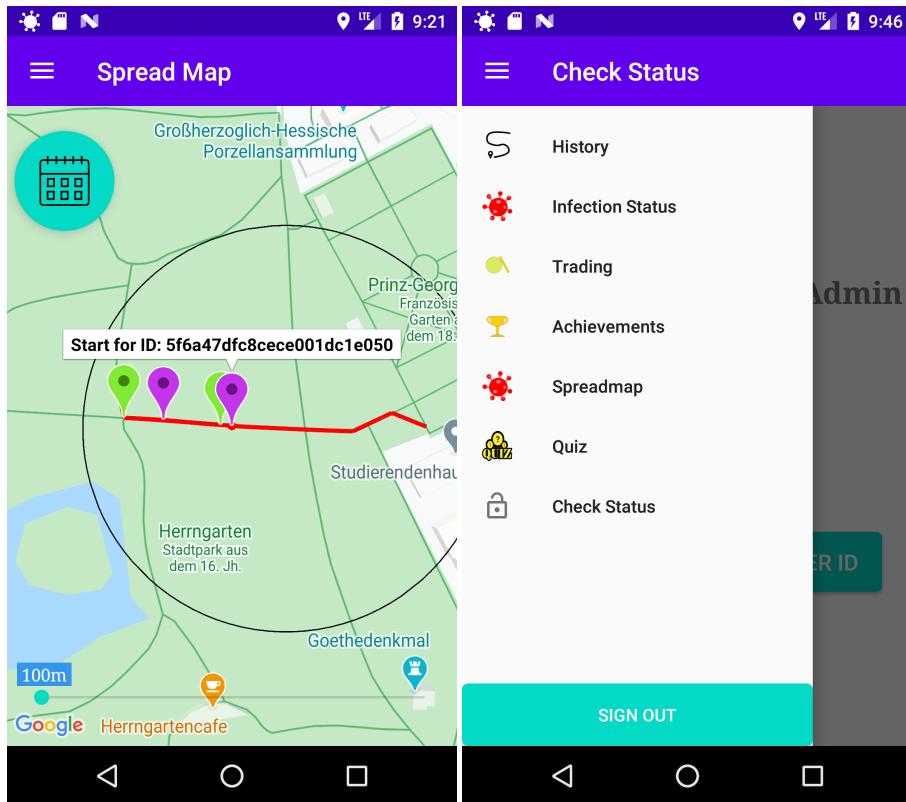


Abbildung 11: Von links nach rechts: Spreadmap mit einem Kontakt, Menü für den Admin

3.3 Administrative Funktionen

Administrative Funktionen können von Admins ausgeführt werden, sind allerdings nicht über die App erreichbar, da sie nur in seltenen Fällen benötigt werden. Sie können entweder per HTTP-Requests oder die Swagger-UI aufgerufen werden. Sie sollten weiterhin besonders vor externen Zugriffen geschützt werden, bspw. könnten sie nur über den localhost verfügbar gemacht werden.

Für die Funktionen wird außerdem ein Admin-JSON-Web-Token (Admin-JWT) benötigt, s. 3.3.1.

3.3.1 Admin-JWT

Für administrative Funktionen wird immer ein Admin-JSON-Web-Token (Admin-JWT) benötigt. Dieses kann durch Eingabe von Nutzer-ID und Passwort eines Admins unter https://vivacoronia.herokuapp.com/swagger/#/Admin%20authentication/post_admin__userId__login_ (Try it out) erhalten werden. Die in der Antwort unter `jwt` stehende Zeichenfolge muss bei den Requests als `adminjwt` angegeben werden. Für den Root-Admin ist die Nutzer-ID `root` das Passwort `GeilnessfaktorForLife!!!`

POST /admin/{userId}/login/

Returns a new JWT for admin functionality

Parameters

Name	Description
userId * required string (path)	The id of the user root

Request body

application/json

Examples: [Modified value]

```
{
  "password": "GeilnessfaktorForLife!!!"
}
```

Execute Clear

3.3.2 Kategorien hinzufügen

Unter https://vivacoronia.herokuapp.com/swagger/#/Trading/post_trading_categories_ können neue Kategorien für den Handel hinzugefügt werden. Dazu muss das Feld *name* mit der entsprechenden Kategorie ausgefüllt werden. Groß-/Kleinschreibung wird berücksichtigt!

Für das Entfernen von Kategorien ist nur der Betreiber verantwortlich, da dies zusätzliche Konsequenzen bedeutet, wie das Verändern von Angeboten oder Warengesuchen.

3.3.3 Admin Promotion & Widerrufung

Unter https://vivacoronia.herokuapp.com/swagger/#/Admin%20feature%20granting/patch_user__userId_ können Nutzern Adminrechte erteilt oder widerrufen werden. Neben dem üblichen Admin-JWT muss die Nutzer-ID unter *userId* angegeben werden. Zudem kann im Request-Body der Parameter *isAdmin* auf *true*, bzw. *false* gesetzt werden, um die Rechte zu erteilen, bzw. zu widerrufen.

3.3.4 QR-Code Generierung

Mit dem QR-Code Generator können QR-Codes nach abgeschlossenen Tests ausgestellt werden. Unter *Date and time of infection or recovery* kann eine Schätzung angegeben werden, seit wann der Patient bereits erkrankt oder gesundet ist. Dies hat einen Einfluss auf die Berechnungen der Kontakte, die wegen einer Infektion gewarnt werden oder wegen einer Gesundung nicht mehr gewarnt werden müssen. Mithilfe von *Additional information* können zusätzliche Informationen angegeben werden, die aber später nicht mehr relevant sind. Beispielsweise könnten bei Gruppen von getesteten Personen, die dieselbe Kontaktmöglichkeit angeben, der jeweilige Name der Person angegeben werden, für den das Ergebnis gültig ist. Die Daten müssen dann im JSON-Object-Format³ angegeben werden. Es sollten nicht zu viele Zusatzinformationen eingegeben werden, damit der QR-Code nicht zu groß wird und einscannbar bleibt.

Es gibt zur Zeit nur ein Private-Public-Key Paar. Dieses kann bei Bedarf ausgetauscht werden, wobei dann alle Aussteller den neuen Private-Key verwenden müssen. Der Public-Key muss dann im Backend unter *res/qrcode_keys/public_key* aktualisiert werden. Der verwendete Signierungsalgorithmus ist Sha256.

³s. https://de.wikipedia.org/wiki/JavaScript_Object_Notation

Der Betreiber der App ist für die Verbreitung des QR-Code Generators selbst verantwortlich. Aktuell ist er durch Herunterladen des Verzeichnisses unter <https://git.rwth-aachen.de/iptk/ss20/team-foxtrot/frontend/-/tree/master/QrCodeGenerator> und Öffnen der index.html erreichbar.

Status * Infected

Date and time of infection or recovery 13.09.2020 14:30

Date and time of approving test* 18.09.2020 17:06

Additional information [{"Issuer": "Dr. Timo Nolle"}]

RSA Private Key
----BEGIN RSA PRIVATE KEY----
MIICWwlBAAKBgHhknWF2wunjg
KCEIH5D3WzCbzZQiHyLwz9azip
7EDCMEACH...

Go

* Required



Abbildung 12: Erstellung eines QR-Codes

4 Features

1. Nutzer müssen vor der Nutzung der App ein Passwort angeben, um sich zu registrieren
2. Nutzer können sich nach der Registrierung ihre Nutzer-ID per E-Mail zuschicken
3. Nutzer können sich vor der Nutzung der App mit einer bereits existierenden Nutzer-ID und Passwort einloggen
4. Nutzer können sich ausloggen
5. QR Codes für den Infektionsstatus können über eine externe HTML-Seite erstellt werden
6. Der Infektionsstatus kann über einen unfälschlichen QR-Code eingescannt werden
7. Der Infektionsstatus kann an den Server gesendet und von ihm abgerufen werden
8. Standortdaten werden regelmäßig aufgezeichnet
9. Standortdaten werden regelmäßig an den Server gesendet
10. Nutzer können sich ihren aufgezeichneten Standortverlauf auf einer Karte anzeigen lassen
11. Nutzer können ihren angezeigten Standortverlauf nach Datum filtern
12. Kontakte mit infizierten Personen werden ermittelt
13. Ermittelte Kontakte werden an die betroffenen Nutzer gesendet
14. Nutzer können in der App mittels eines Drawer-Menüs navigieren
15. Es können Produktkategorien hinzugefügt werden (Swagger)
16. Nutzer können Produkte zum Verkauf anbieten
17. Nutzer können ihre bestehenden Angebote verändern
18. Nutzer können ihre Angebote löschen und dabei angeben, ob diese verkauft wurden oder nicht
19. Nutzer können sich ihre Angebote in einer Liste ansehen
20. Nutzer können sich die Angebote anderer Nutzer in einer Liste anzeigen lassen
21. Nutzer können sich die Angebote anderer Nutzer auf einer Karte anzeigen lassen
22. Nutzer können nach Angeboten suchen und dabei Filtereinstellungen vornehmen
23. Nutzer können Bedarf an einem Produkt anmelden
24. Nutzer können Bedarf an einem Produkt löschen und dabei angeben ob dieser Bedarf erfüllt wurde
25. Nutzer werden benachrichtigt, falls bestehende oder neue Angebote zu ihren Bedürfnissen passen
26. Nutzer können Supermärkte über eine Karte auswählen
27. Nutzer können das Inventar eines ausgewählten Supermarktes ansehen und verändern
28. Nutzer können sich ihren Nutzerstatus (Admin/User) und Nutzer-ID anzeigen lassen
29. Nutzer können sich, falls sie über die nötigen Rechte verfügen, als Administrator anmelden
30. Ein Nutzer mit Administratorrechten sieht zusätzlich den Menüpunkt 'Spreadmap'
31. Es gibt eine Spreadmap für Administratoren, auf welcher der Standortverlauf der Nutzer angesehen werden kann
32. Die Anzeige der Spreadmap kann nach Nutzer und Zeit gefiltert werden
33. Achievements für Trading und Kontakte
34. Nutzer können Quizes gegeneinander spielen
35. Es können Quizfragen hinzugefügt werden (Swagger)

5 Zusammenfassung

Wie jede andere App, die Lösungen anbietet, hat auch Viva Corona Problem. Zum einen ist die Standortaufzeichnung über GPS etwas ungenau, wenn man von einem Infektionsradius von 1,5m-2m spricht, da die Genauigkeit von GPS (mit ca. 5m) nicht ausreichend gegeben ist. Auch der angezeigte Standortverlauf enthält Zick-Zack-Muster, da die Aufzeichnung zu ungenau ist. Eine weitere Schwierigkeit entsteht bei der Generierung des QR-Codes. Dieser benötigt einen digitalen Signatur, damit die Infiziertenmeldung nicht gefälscht werden kann. Hierbei steigt jedoch die Größe des QR-Codes an. Außerdem sind die Nutzer-IDs automatisch generiert und somit nicht leicht zu merken. Dies beeinflusst die Nutzerfreundlichkeit negativ. Die Änderung ist möglich, aber nur über dritte Wege (Backend). Da die Nutzer-ID in den meisten Fällen aber überhaupt nicht eingegeben werden muss, ist dies ein relativ unwichtiges Problem.

Des weiteren existieren noch Stellen, die in der Zukunft ausbaufähig sind. So kann man bei der QR-Code Generierung aktuell nur einen einzigen Private-Key verwenden. Hierfür könnte man noch eine Schnittstelle bieten, damit das Ausstellen der QR-Codes besser kontrolliert werden kann. Weiterhin könnte man die Produktgesuche durch verschiedene Funktionen erweitern. Möglich wäre ein zusätzlicher Knopf in der Suchansicht, um diese Suche zu abonnieren und so automatisch ein Gesuch zu erstellen. Außerdem könnte man das Benachrichtigen von passenden Produkten auf die Supermarktangebote erweitern, sodass auch bei Aktualisierung der Supermarktinventare die Nutzer benachrichtigt werden. Zudem könnte man die Nutzer-ID bedienungsfreundlicher gestalten, damit Nutzer sich diese einfacher merken können.

Insgesamt bekämpft Viva Corona aber einige wichtige Probleme, die mit dem Aufkommen der Corona-Pandemie entstanden sind. Auch wenn manche Funktionen der App ausgenutzt werden können, bzw. bei bösartiger Benutzung zum Absturz führen, so überwiegen doch die nützlichen Funktionen. Der Handel mit Produkten, um die Warenknappheit zu bekämpfen, hätte im Frühjahr 2020 sicherlich einen breiten Anklang gefunden und einigen Menschen die Lebenssituation erleichtert. Insbesondere die etwas übervorsichtige Kontaktverfolgung kann mit einer breiten Nutzung sicherlich einen erheblichen Beitrag leisten, um die aktuelle Situation zu verbessern.