num compass

Erläuterung der technischen Entwicklungs-varianten für Pandemie-Apps

COMPASS Arbeitspaket 3_Regulatorische Anforderungen

Das vorliegende Dokument dient der detaillierten Beschreibung und Betrachtung von Vor- und Nachteilen technischer Entwicklungs-varianten und -alternativen, welche für die Gestaltung von Pande-mie-Apps potentiell denkbar sind.

Version 1.0 _ 30.04.2021

netzwerk universitäts medizin

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Inhaltsverzeichnis

Ziele des Dokumentes

Das vorliegende Dokument dient, entsprechend der Definition im Projektantrag, der detaillierten Beschreibung und Betrachtung von Vor- und Nachteilen technischer Entwicklungsvarianten und -alternativen, welche für die Gestaltung von Pandemie-Apps potentiell denkbar sind.

Das Hauptaugenmerk wurde hier auf die native und die webbasierte App-Entwicklung gelegt, da diese, Stand heute, die am weitesten verbreiteten Entwicklungsmethoden der angebotenen Apps darstellen. Nichtsdestotrotz wird versucht auch weitere Varianten, wie die hybride Entwicklung oder die Realisierung mithilfe von Crossplattformen, in die Betrachtung einzubinden. Die einzelnen, verschiedenen Varianten werden einander gegenübergestellt und miteinander verglichen. Als Vergleichskriterien dienen sowohl generelle Anforderungen an Apps, wie Performance, Verfügbarkeit, Kosten etc., als auch regulatorische Anforderungen, welche gerade bei der Entwicklung von Pandemie-Apps eine bedeutende Rolle spielen. Für einen anschaulichen Vergleich und einer detaillierten Betrachtung werden unter anderem spezifische Anwendungsfälle von bereits entwickelten und genutzten Apps als Use-Cases herangezogen.
Final soll ein Gesamteindruck bezüglich der Vor- und Nachteile, sowie der spezifischen Möglichkeiten der unterschiedlichen Entwicklungsvarianten entstehen, wodurch auch gewisse Empfehlungen für die Entwicklung von Pandemie-Apps, anhand deren Anforderungen, gegeben werden können.
Zusätzlich wurde ein Privatsphäremodell für die Entwicklung von Apps erarbeitet, um, unabhängig der Entwicklungsvariante, das Vertrauen und die Akzeptanz von möglichen Nutzern für eine solche Pandemie-App zu steigern.

Kurzzusammenfassung

Allein die Betrachtung der hier im Dokument aufgeführten Apps (Corona Health, Corona-Warn-App und CovApp) belegt die starke Variation der einzelnen Pandemie-Apps in Hinsicht auf ihren Zweck und auf deren Datenerhebungen. Die einzelnen Entwicklungsvarianten weisen in unterschiedlichen Bereichen bzw. für verschiedene Aufgaben jeweils Vor- und Nachteile auf. Web-Apps sind in der Regel kostengünstiger und durch die Lauffähigkeit in einem Browser leichter für möglichst viele Nutzer zur Verfügung zu stellen. Native Apps haben ganz klare Vorteile in Bereichen wie Performance und der Verwendung von Sensorik. In Bezug auf die regulatorischen Anforderungen unterscheiden sich der native und der webbasierte Ansatz nur gering, generell können diese Anforderungen mit mehr oder weniger Aufwand erreicht werden, auch wenn dies für die jeweiligen Varianten in bestimmten Bereichen teils einfacher oder teils schwieriger sein kann.
Zusammengefasst ließ sich rasch erkennen, dass bereits bei der Planung einer App genau abgewogen werden muss, welche Aufgaben und Anforderung an jene App gestellt werden. Dadurch lässt sich sicherstellen, dass die meisten Vorteile einer Entwicklungsvariante für die App gewonnen werden. Zukünftig müssen allerdings auch hybride Ansätze und vor allem Crossplattformen berücksichtigt werden, da diese Varianten versuchen die Vorteile der jeweiligen Varianten in sich zu vereinen.

Definitionen/Begriffserklärungen sowie Betrachtung von App-Kategorien:

Native App: Diese App wird spezifisch für ein Betriebssystem entwickelt. Je nach System wird ein entsprechender Programmcode geschrieben, welcher auf die Möglichkeiten und Features des jeweiligen Systems hin ausgerichtet bzw. ausgelegt wird. Die App und ihr Programmcode mit all den inhärenten Möglichkeiten auf das spezifische System hin, laufen damit in einer für sie namensgebenden "nativen Umgebung".

Webbasierte App: Diese App läuft über einen Browser und ist in der Regel auf Basis einer HTML-Seite (plus "css" für Design) geschrieben, verknüpft mit Programmcode, welcher mittels einer gängigen Script- bzw. Programmiersprache (JavaScript, PHP, Ruby, etc.) implementiert wurde . Der Browser eines (jeden) Systems liest über das "Browser-Script" die App ein. Damit kann jedes System, anders als bei einer nativen App, die webbasierte App ausführen, wenn der verwendete Browser nach W3C-Standard arbeitet.

Hybride App: Die Hybrid-App ist eine Mischform zwischen webbasierter – und nativer App und kann damit die meisten Features der jeweiligen Systeme, auf denen sie zur Anwendung kommt, nutzen. Die Hybride App generiert/emuliert, sozusagen in einer Meta-Ausführungsschicht, eine native Umgebung, um spezifische Anforderungen der jeweiligen Systeme bedienen zu können.

Cross(over)plattform: Unter diesem Label firmieren Programme, welche über eine emulierte Ausführungsschicht ihren Programmcode plattformübergreifend ausführen können. Die hybride App, wie auch die webbasierte App, sind dabei mögliche Varianten eines "crossover" auf unterschiedlichen Plattformen bzw. Systemen. Sobald ein Programm in der Lage ist, plattformübergreifend (=crossover) Programme bzw. Apps ausführen zu können und damit auf ein anderes System "portiert" werden kann, fällt es unter die Kategorie Crossplattform.

System/Plattform: Unter einem System oder einer Plattform wird in unserem Fall das Gerät (Smartphone, Tablet aber auch PC) und sein Betriebssystem (MacOS, Android, MS-Derivate, Linux) und den daraus ergebenden Möglichkeiten einer "Programm-Lauffähigkeit" in seiner Gesamtheit verstanden. Wichtig ist dabei die Kompatibilität der einzelnen Komponenten, da z. B. weder MS-Programme auf einem Android-Smartphone/Tablet noch MacOS-Apps auf einem PC ohne weiteres lauffähig sind (d. h. ohne Emulator oder dafür modifizierten Programmcode/Script).

Emulator: Wie zuvor ausgeführt, sind bestimmte Komponenten untereinander nicht ohne weiteres kompatibel, sprich aus sich heraus lauffähig. Um daher beim vorherigen Beispiel zu bleiben, bedarf es einer zusätzlichen Programminstanz um eine native Android-App auf einem Windows Rechner zu betreiben. Eine solche Instanz, welche sozusagen eine natürliche (native) Umgebung für das auszuführende Programm kreiert bzw. simuliert, wird Emulator genannt.

Gegenüberstellung der Varianten

Generelle Vor-/Nachteile

Während *native Apps* relativ aufwendig in der Code-Programmierung (und damit auch teuer) sind, da sie jeweils für ein einziges System geschrieben werden, sind sie aufgrund ihrer spezifischen Erstellung für ein System auch wie keine andere Form einer App in der Lage, die jeweiligen Ressourcen und Möglichkeiten des jeweiligen Systems, für das sie erstellt werden, zu nutzen. Der Nachteil einer *nativen App* liegt in ihrer Exklusivität auf ein System. Eine Portierung auf ein anderes System ist deshalb teuer und zeitaufwändig.

Webbasierte Apps hingegen sind einfacher zu erstellen, da ein Programmcode für einen Webbrowser geschrieben wird, der dann auf den jeweiligen Systemen in Form einer browserbasierten App eingelesen wird. Da dies relativ wenig Aufwand zur Erstellung bedarf, sind diese Apps günstiger zu erstellen, können aber nur die rudimentären „Standardfunktionen“ des jeweiligen Browsers nutzen. Damit sind *webbasierte Apps* in der Regel aber nicht in der Lage, spezielle Ressourcen und Features der jeweiligen Systeme gerade im Hinblick auf Barrierefreiheit zu nutzen, wie dies z. B. bei Mac-Systemen für Gehörlose, Blinde oder Sehbehinderte, sowie für ältere Menschen möglich wäre. Auch die Datensicherheit kann, durch den allgemein gehaltenen Programmcode, nicht auf die vollen Sicherheitsfeatures der jeweiligen Systeme hin erstellt, bzw. ausgenutzt werden.

Die *Hybrid-App* als Mischform zwischen *Web – und nativer App* kann die meisten Features der jeweiligen Systeme nutzen, hat aber in Puncto Datensicherheit und Enduser-Anpassung immer noch leichte Einbußen an Leistungsfähigkeit hinsichtlich den Spektrumsmöglichkeiten einer *nativen App*. Die Performance einer *Hybrid-App* ist, so sie nicht akribisch und aufwändig programmiert wurde und damit teuer und zeitaufwändig ist, bedingt durch ihre emulatorische Natur, nicht mit der einer nativen App vergleichbar, da es aufgrund von Latenzproblemen durch die notwendige Emulation zu Leistungseinbußen, bzw. Ressourcenhunger kommen kann.

Bei der Bezeichnung Crossplattform kann differenziert werden, je nachdem wie der Begriff Crossplattform ausgelegt wird. Zum einen ist die *Cross(over)plattform* entweder browserbasiert (und damit eine Art *Webapp)* oder ein Emulator, welche mittels einer künstlich erstellten Umgebung, für Apps, welche für eine andere Plattform/anderes System geschrieben wurden, ein “natives Enviroment” emuliert. Damit kann durch ein Crossover in der Regel jede App auf allen System zum Laufen gebracht werden. Der Nachteil eines solchen Crossovers ist ein enormer Performanceverlust, da die Emulation große Mengen an Prozessor und RAM-Kapazität für die Erstellung einer solchen künstlichen, nativen Umgebung verbraucht. Des Weiteren kann auch eine *Cross(meta)plattform* nicht zwangsweise alle spezifischen Ressourcen eines Systems ausnutzen, für welches der App-Code ursprünglich geschrieben wurde.

Zum anderen werden heutzutage als Crossplattformen hauptsächlich Frameworks bezeichnet, die es ermöglichen mit nur einer “Codebasis” alle Betriebssysteme (wobei meist nur iOS und Android bedient werden) als “*native App”* abzudecken. Hierbei wird die App demnach lediglich einmal implementiert und aus jener “Codebasis” werden die spezifischen “*nativen*” Apps durch das Framework generiert. Der Vorteil der geringeren Kosten und des geringeren Aufwands durch die zentrale Entwicklung “einer” App liegt auf der Hand. Ebenso können Apps, welche mittels einer solchen Plattform entwickelt wurden generell, gleichermaßen wie nativ entwickelte Apps, auf die volle Sensorik der Endgeräte zugreifen. Ein Nachteil stellen Geräte-/Hardware- und Betriebssystembesonderheiten der einzelnen Hersteller dar. Wenn diese sich in ihren Erreichbarkeiten und Verwendungen stark voneinandern unterscheiden, müssen diese oftmals gesondert, jeweils für das entsprechende Paket, implementiert werden. Das gleiche Problem könnte bei Neuerscheinungen und neuen Hardware- und Softwareeigenschaften auftreten, sofern das benutze Framework noch nicht auf jene Neuheiten angepasst wurde.

Gegenüberstellung der verschiedenen Vorgehensweisen

Betrachtet werden die verschiedenen Vorgehensweisen anhand der Kriterien Performance, Erreichbarkeit, Kosten etc.:

Kriterium	**Native App**	**Web-App**
Verfügbarkeit/Kompatibilität	Apps müssen fast ausschließlich über die gängigen App Stores geladen werden, um genutzt werden zu können. Dadurch für kleine, simple Aufgaben zu viel Aufwand (Suche → Download → Installation → Benutzung). Kompatibel nur mit Zielsystem. Dazu spezielle Funktionalitäten nur mit oder ab bestimmten Betriebssystemversionen verfügbar (vgl. Corona Warn App Bluetooth Low Energy) Icon der App automatisch auf dem Smartphone oder Tablet zu sehen. Wird in den Stores mit ähnlichen Apps angezeigt, bzw. wenn der Nutzer nach bestimmten Begriffen sucht, wodurch die Reichweite gesteigert wird.	Für den Nutzer einfach zu finden. Apps werden meist unbewusst über die Suchfunktion des Browsers gefunden und sind für jegliche Geräte selbst mit unterschiedlichen Betriebssystemen, durch Standardbrowser, gleichermaßen nutzbar. (Vergleich: Suche → Benutzung) Um die Web-App schnell wiederverwenden zu können muss ein Lesezeichen im Browser oder ein Link auf dem Home-Bildschirm des Smartphones oder Tablets erstellt werden. Dieser Vorgang ist für Nutzer eher unbekannt und wird nicht häufig verfolgt.
Installation/Update	Download und Installation meist über App Stores. Update ebenfalls über die Stores. Zu beachten ist, ob der Nutzer automatische Updates zulässt oder nicht. Nutzer haben möglicherweise nicht immer alle die gleiche bzw. aktuellste Version der App.	Kein Download und Installation nötig, da die App über Browser im Web läuft. Nutzer greifen dadurch auch immer auf die gleiche und aktuellste Version zu.
Entwicklungskosten/Aufwand	Kann bis zu doppelt so hohe Entwicklungskosten verursachen wie Web-App, da in den meisten Fällen mindestens zwei Betriebssysteme (iOS und Android) bedient werden müssen. Im Normalfall durch Synergieeffekte jedoch “nur” bei 20-50% Mehrkosten im Vergleich zu einer Web-App-Entwicklung. Aufwand könnte sich vergrößern, sollte keine parallele Programmierung möglich sein.	Relativ geringe Kosten, da nur eine Entwicklung für alle Plattformen benötigt wird. Aufwand bleibt überschaubar und für alle Geräte identisch. Nichtsdestotrotz kann man nicht allzu viel Ersparnis erwarten, da eine App-Entwicklung mehr beinhaltet als die reine Realisierung der App. Benötigt werden Designs, Backend, Testungen, Qualitäts- und Projektmanagement. Hier lassen sich auch bei einer

		Web-App kaum Einsparungen finden.
Performance	Im Normalfall läuft eine Native App deutlich schneller und flüssiger → Hauptmerkmal der Usability. Ressourcen werden besser verwaltet. Spezielle Steuerungen und Gesten können genutzt und programmiert werden. Die Leistungsstärke des Gerätes/Betriebssystems kann bestmöglich ausgenutzt werden. Native UI-Elemente und Guidelines können recht einfach eingebunden werden.	Performance hängt unter anderem vom benutzten Browser ab. Ressourcenverwaltung der Geräte schwieriger. Da die App für viele verschiedene Geräte und Betriebssystemversionen komplett identisch ist, entstehen möglicherweise Performance-Einbußen bei extrem leistungsstarken oder extrem leistungsschwachen Geräten, je nach Implementierung und Zielsetzung der App.
Spezielle Gerätefunktionen	Zugriff auf jegliche Geräte-Hardware (Kamera, Beschleunigungssensoren, Bluetooth usw.) durch Zustimmung des Nutzers möglich. Keine Einschränkungen der Funktionalität durch Umgebung.	Zugriff auf Geräte-Hardware möglicherweise eingeschränkt oder überhaupt nicht vorhanden. Abhängigkeit von genutztem Browser und dessen Funktionalitäten. Möglicherweise abschreckend für Nutzer wenn Berechtigungen für bestimmte Funktionalitäten (zum Beispiel Kamera) für kompletten Browser erteilt werden müssen, obwohl es nur für die WebApp gewünscht ist.
Internetverbindung	Es ist einfach den Großteil der App auch Offline für den Nutzer verfügbar zu machen, da es bis auf die Größe des Gerätespeichers keine Begrenzung gibt, Daten zu speichern.	Offline-Funktionalitäten durch HTML5 zwar möglich aber sehr eingeschränkt. Bei älteren genutzten Technologien nahezu unmöglich. Somit wird eine Internetverbindung fast immer für die Nutzung benötigt.
Bewertungen/Analyse	Apps können in den Stores von allen bewertet werden. Dadurch können sich Nutzer bereits vorab ein Bild der App machen. Weiter entsteht ein Feedback, welches oftmals positive und negative Aspekte der App offenlegt und dadurch Updates und Verbesserungen anstößt. Weiter können Downloadzahlen, Nutzerzahlen, Bewertungsanzahl, Bewertungen an sich etc. genau betrachtet und mittels verschiedener Programme analysiert werden.	Bewertungen sind zwar generell möglich, allerdings gibt es kein übergreifendes Bewertungssystem der Apps wie in den App Stores. Wenn eine Web-App eine Bewertung abfragt, dann ist dies meist nur ein direktes Feedback an die Entwickler. Daher Bewertungsanzeige, wie aus den App Stores bekannt, weniger glaubwürdig für die Nutzer, da diese nicht unbedingt von einer unabhängigen dritten Partei kommen müssen.

Tabelle 1: Gegenüberstellung der verschiedenen Vorgehensweisen

Detaillierte Betrachtung von Anwendungsfällen

Anwendungsfall	**Native App: Corona Health**[1]	**Native App: Corona Warn App (RKI)**[2]	**Web-App: CovApp (=Data4Life)**[3]
Befragungen	ja; physische und psychische Gesundheit	nein, da Ermittlung von Hotspots	ja; Fokus auf Covid-Symptome
Informations-bereitstellung	ja; Feedback, wie Hinweise, Neuigkeiten und Tipps	ja, Nachrichten und Warnungen, bei evtl. Infektionsgefahr	ja, als Entscheidungshilfe bei etwaigen Symptomen und konkrete Handlungs-empfehlungen, sowie Anlaufstellen und Kontakte
GPS-Erfassung	ja; vergröbert auf 11 km	nein	nein
Bluetooth-Tracking	nein	Kontakt-Tracing via Bluetooth	nein
Datenübertragung	Der Datenaustausch zwischen Endgerät und dem Server erfolgt über das Internet, auf Grundlage einer gesicherten SSL-Verbindung, Daten werden auf einem geschützten Server der Universität Würzburg gespeichert	Tracking über das Verbinden mit anderen App-Usern mittels verschlüsseltem Code, um so Infektionsherde zu lokalisieren; keine direkte Standortermittlung sondern Warnung durch positiv getestete User, in deren näheren Umgebung man über einen gewissen Zeitraum war	nein, Daten verbleiben lediglich auf dem Endgerät
Wissenschaftliche Studien	ja 5, Monitoring der psychischen und körperlichen Gesundheit während und nach der Covid-19-Pandemie für Jugendliche und Erwachsene, Stresserkennung für Erwachsene und Compass	nein, primär das Warnen vor Umgang mit Infizierten, Datenaustausch mittels RKI	nein, Zusammenarbeit mit der Charite Berlin, jedoch keine Übermittlung an Dritte, Daten bleiben auf dem Endgerät

[1] Basierend auf: https://www.uni-wuerzburg.de/aktuelles/pressemitteilungen/single/news/corona-per-app-der-wissenschaft-helfen/ (letzter Zugriff 29.04.2021)
https://play.google.com/store/apps/details?id=com.dbis.haugxhaug.coronahealth (letzter Zugriff 29.04.2021)

[2] Basierend auf:
https://play.google.com/store/apps/details?id=de.rki.coronawarnapp (letzter Zugriff 29.04.2021)
https://www.coronawarn.app/assets/documents/cwa-eula-de.pdf (letzter Zugriff 29.04.2021)
https://github.com/corona-warn-app/cwa-app-android (letzter Zugriff 29.04.2021)

[3] Basierend auf:
https://play.google.com/store/apps/details?id=care.data4life.hub (letzter Zugriff 29.04.2021)
https://www.d4l.io/blog/covapp-faq/ (letzter Zugriff 29.04.2021)

	Bevölkerungsumfrage		
Datenschutz/ Anonymität	Anonymisierung der Daten, kein personalisiertes Tracking, keine Rückschlüsse auf einzelne Personen möglich	Anonymisierung durch unpersönliche Anmeldung, Ermittlung nur durch den Austausch von verschlüsselten Codes mit App-Usern in der Nähe	Datenschutz nach ISO 27001 zertifiziert
Tagebücher/ Symptomtracker	ja, jedoch Fokus auf Langzeitauswirkungen, insbesondere der psychischen Gesundheit und psychosozialen Belastungen, Benachrichtigungen für regelmäßige Teilnahme	ja, Symptomtagebuch	ja, Symptomtagebuch
Newsletter	ja	n. n. b.	nein
Technik; Verlinkungen, Extras	Native App; Anonyme Erhebung der Nutzungsdaten von 10 festgelegten (Facebook, WhatsApp etc.) und der 5 meistgenutzten Apps	Native App: Anonyme Herstellung von “Begegnungsdaten” via Code; Risikoeinschätzung; regelmäßiger Onlinebetrieb erforderlich	Webbasierte (xml) Anwendung; Link zum Covid-Fragebogen der Charité[4]; Sourcecode verfügbar auf Github
Bewertung (via GooglePlay)	4 von 5 (25 Kommentare)	3 von 5 (überwiegend positive Kommentare);	keine da Web-App; Data4Life 2,8 von 5 (8 Kommentare)

Tabelle 2: Detaillierte Betrachtung von Anwendungsfällen

Fazit und Empfehlungen für spezifische Anwendungsfälle

Da an die vorgestellten Apps unterschiedliche Anforderungen gestellt und verschiedene Ziele verfolgt werden, unterscheiden sie sich in ihrer Funktionsweise und Datenerhebung.

Die Corona Health App und die Corona-Warn-App sind stark auf Performance-Leistungen und Nutzung von spezifischen Geräte-Sensoren ausgelegt. Ebenso nutzen bzw. benötigen diese teilweise eine längerfristige, lokale Datenspeicherung. So erfasst die Corona Health App parallel zu den Antworten der Fragebögen die GPS-Position der Nutzer und Nutzungsdaten verschiedener Apps (sofern die Berechtigung erteilt ist). Die Corona-Warn-App basiert generell auf der Bluetooth Funktion, welche für das Kontakt-Tracing benutzt wird. Hierbei werden die IDs der Risikobegegnungen lokal auf den Geräten gespeichert.

Die CovApp stellt lediglich einen Fragekatalog bereit und gibt anhand der angegebenen Antworten etwaige Handlungsempfehlungen aus. In Verbindung mit der Data4Life App lässt sich zusätzlich eine Art Symptom-Tagebuch erstellen.

[4] Zugriff auf Fragebogen: https://covapp.charite.de/

Aufgrund der unterschiedlichen Funktionen und Zielsetzungen wurden hier generell die richtigen Entwicklungsvarianten für die jeweilige App gewählt. Als Empfehlung lässt sich demnach festhalten, dass Apps deren Anwendungsfälle auf Performance, guter Usability, persistente lokale Datenhaltung und Hardware-Sensorik beruhen, nativ entwickelt werden sollten. Bei Apps, welche möglichst schnell für möglichst viele Nutzer verfügbar sein sollen, mit Anwendungsfällen geringerer Komplexität ist die webbasierte Entwicklungsvariante vorzuziehen.

Unterschiede bezüglich regulatorischer Anforderungen

<table>
<tr><th>Regulatorische Anforderung</th><th>Native App</th><th>Web-App</th></tr>
<tr><td>Medizinproduktegesetz (MPG), Medizinprodukteverordnung (MPV)</td><td>Nachteil: Entwicklung erfolgt zumeist für mindestens zwei Betriebssysteme (Android und iOS), wodurch zwei Codebasen/Apps dokumentiert und zertifiziert werden müssen.</td><td>Vorteil: Weniger Aufwand, da App in den gängigen Browsern läuft, wodurch nur eine Codebasis entwickelt werden muss. Dadurch muss entsprechend auch nur eine App dokumentiert und zertifiziert werden.</td></tr>
<tr><td>Datenschutz-Grundverordnung (DSGVO)</td><td>Für den Upload der Installationsdateien in die gängigen Stores (Google Play Store und Apple App Store) wird bereits ein konformes Datenschutz Dokument gefordert, ansonsten ist kein Upload möglich.

Evtl. Problem: Stimmt ein (im Entwicklungsland) generell korrektes Datenschutz-Dokument nicht mit den Richtlinien der Stores überein, kann evtl. keine Bereitstellung der App über die Stores erfolgen.</td><td>Kein Upload in App-Store nötig. Dadurch keine Anpassung des Datenschutzes an Richtlinien Dritter. DSGVO-konform immer ausreichend.</td></tr>
<tr><td rowspan="2">Barrierefreiheit (BGG)</td><td colspan="2">Allgemein alle Voraussetzungen gegeben, um Barrierefreiheit in möglichst großer Dimension zu gewährleisten (z. B. Text-zu-Sprache, Spracheingabe usw.)

Generelle Einschränkungen sind durch Betriebssysteme, Browser und Geräte sowohl bei nativer als auch bei webbasierter Entwicklung unabhängig von der Implementierung möglich, da nicht jegliche Funktionalität und Sensorik von allen gleichermaßen unterstützt wird.</td></tr>
<tr><td>Generelle Voraussetzung: zumindest phasenweise funktionierende Internetverbindung.

Alle Geräte zu bedienen beinahe unmöglich, da viele verschiedene (veraltete) Betriebssysteme (Stichwort Huawei).</td><td>Generelle Voraussetzung: Da solche Apps im Browser laufen, ist eine funktionierende Internetverbindung Grundvoraussetzung.</td></tr>
</table>

<table>
<tr><td rowspan="2">Digitales Versorgungsgesetz</td><td colspan="2">Generelle Voraussetzung: Existierende Infrastruktur. Darunter zählt unter anderem die Versorgung der Nutzer mit der nötigen Hardware, um die App nutzen zu können, sowie die entsprechenden Serverstrukturen, um eine reibungslose Bereitstellung der Services zu gewährleisten.</td></tr>
<tr><td>Generelle Voraussetzung: Existierende Infrastruktur. Dazu zählt eine zumindest phasenweise funktionierende Internetverbindung.

Eine Schwierigkeit besteht in der Versorgung aller sich im Umlauf befindlichen Betriebssysteme, da die App jeweils dafür implementiert werden muss.
Ein Vorteil der nativen Entwicklung ist die Ein- bzw. Anbindung von zusätzlicher Hardware wie z. B. Smartwatches, auf denen die App laufen kann/soll.</td><td>Generelle Voraussetzung: Existierende Infrastruktur. Dazu zählt eine funktionierende Internetverbindung, da Web-App in einem Browser läuft.

Es existieren bereits Web-Browser für Smartwatches (z. B. WIB (Wear OS (Android Wear) Internet-Browser)), allerdings sind Web-Apps für Smartwatches noch nicht sehr verbreitet.</td></tr>
<tr><td rowspan="2">eHealth-Gesetz</td><td colspan="2">Betrifft generell beide Entwicklungsvarianten gleichermaßen. Grundvoraussetzung aller eHealth Anwendungen bzw. mHealth Anwendungen (mobile eHealth) ist der sichere, verschlüsselte Datentransfer und die sichere und verschlüsselte Datenspeicherung.</td></tr>
<tr><td>Vorteil der nativen App Entwicklung ist, dass generell auf jegliche Sensorik und Hardware des Gerätes zugegriffen werden kann. Dies würde z. B. auch das Auslesen von Gesundheitsdaten der Gesundheitskarte mittels NFC über ein entsprechend fähiges Smartphone ermöglichen.

Weiter ist wiederum die Anbindung externer Hardware, wie z. B. Smartwatches oder auch Notknöpfe einfacher zu realisieren.</td><td>Sofern der installierte Browser bzw. die geöffnete Web-App auf entsprechende Sensorik des Gerätes zugreifen kann und darf, sind auch mittels einer Web-App links beschriebene Vorgänge generell möglich.</td></tr>
<tr><td>Infektionsschutzgesetz (IfSG)</td><td colspan="2">keine rechtliche Vorgabe → spielt momentan keine wirkliche Rolle</td></tr>
<tr><td>Europäische und internationale Richtlinien und Verordnungen</td><td colspan="2">Europäische oder internationale Richtlinien beschränken native Apps und Web-Apps in ihrer Nutzung generell in gleichem Maße. Kriterium sind die Richtlinien und Verordnungen der jeweiligen Länder, in welchen die App genutzt werden soll. Die Beschränkungen betreffen unter anderem auch hier betrachtete Vorgaben, wie etwa den Datenschutz, die Barrierefreiheit und Medizinprodukteverordnungen.</td></tr>
</table>

	Ein Problem sind eventuelle Richtlinien der Store-Betreiber, auf welchen die App bereitgestellt werden soll. Weichen Apps in ihrer Implementierung von jenen Richtlinien ab, werden jene Apps in der Regel nicht von den Store-Betreibern zugelassen und können somit nicht von dort, von den Nutzern, bezogen werden.	Web-Apps müssen meist keine entsprechende Prüfungen durchlaufen, wodurch jene verhältnismäßig einfach über einen Browser bereitgestellt werden können. Allerdings verringert eben dieser Umstand der Unkontrollierbarkeit gegebenenfalls das Vertrauen der Nutzer in diese Apps.

Tabelle 3: Unterschiede bezüglich regulatorischer Anforderungen

Fazit und Empfehlung

Wie an den vorgestellten Apps zu erkennen ist, unterscheiden sich Pandemie-Apps stark in ihren angestrebten Absichten und in ihrer Art der Datenerhebung. Für die Entwicklung einer Pandemie-App muss nun bereits bei der Planung abgewogen werden, worauf der Fokus der App später liegen soll, da die unterschiedlichen Entwicklungsvarianten jeweils Vor- und Nachteile in verschiedenen Bereichen vorweisen. Grundvoraussetzung für alle Varianten ist die entsprechende Infrastruktur, womit vor allem eine funktionierende Internetverbindung aber auch die entsprechende Hardware gemeint ist. Allerdings lässt sich hier bereits ein kleiner Vorteil der nativen App-Entwicklung erkennen, da diese lediglich eine sporadische Internetverbindung für die Kommunikation mit dem Server benötigt, wohingegen Web-Apps meistens auf eine dauerhafte Verbindung angewiesen sind. Vorteil der Web-App hingegen ist, dass diese auch auf jedem normalen Rechner mit Browser bedient werden kann.
Soll die geplante Pandemie-App in allererster Linie performance-stark und dadurch ein erhöhtes Maß an Usability gewährleisten, sowie Zugriff auf möglichst alle verfügbaren Sensoren der Geräte haben und möglichst externe Hardware, wie Smartwatches, einbinden können, dann ist die native Entwicklungsvariante der webbasierten vorzuziehen.
Soll die geplante App hingegen vor allem möglichst kostengünstig und einfach für beinahe alle potentiellen Nutzer verfügbar sein, dann hat die webbasierte Entwicklung die Vorteile auf ihrer Seite, da dort nur eine Codebasis implementiert werden muss und die App in jedem gängigen Browser lauffähig ist. Dieser Umstand räumt der Web-App auch einen weiteren kleinen Vorteil in Bezug auf die Barrierefreiheit und das Versorgungsgesetz ein, da die App auf jedem Gerät, egal welches Betriebssystem, genutzt werden kann.
In Bezug auf die regulatorischen Maßnahmen bei der Entwicklung von Pandemie-Apps unterscheiden sich native und Web-Apps aber nur marginal. Beide Varianten verbuchen jeweils minimale Vor- und Nachteile für sich. So hat die webbasierte Entwicklung eben erwähnte Vorteile in Sachen Versorgungsgesetz, wohingegen die Native App bezüglich Datenschutz, durch die zusätzliche Kontrolle der App Store Betreiber, etwas besser abschneidet. Jene Sicherheit der App-Kontrolle durch die Stores vergrößert meist das Vertrauen und die Akzeptanz der Nutzer, allerdings können native Apps dadurch auch an ihrer Verbreitung eben durch jene Store-Betreiber blockiert werden, sofern die Richtlinien der entwickelten nativen App den Richtlinien der App Stores nicht ausreichend entsprechen.

Zusammengefasst lässt sich nun zügig erkennen, dass genau betrachtet werden muss, was die Hauptaufgabe der geplanten App sein soll, um möglichst die meisten Vorteile der jeweiligen Entwicklungsvariante für sich zu gewinnen. Nicht zu vergessen ist, dass hier hauptsächlich die native- und webbasierte Entwicklung beleuchtet wurden. In Zukunft werden hybride Entwicklungen und vor allem die Entwicklung mittels Crossplattformen in den Vordergrund treten, da diese versuchen die Vorteile beider Seiten (nativ und webbasiert) in sich zu vereinen. Dass bei der hybriden Variante oftmals noch Einbußen bezüglich der Performance und Usability auftreten, fundiert die Annahme, dass zukünftig Apps vermehrt durch Crossplattformen bzw. Frameworks wie Flutter, React Native, NativeScript, Xamarin etc. entwickelt werden. Diese haben den Vorteil, dass sie mehrere Betriebssysteme mit “nativen Apps” bedienen und dadurch beinahe alle Vorteile von nativen Apps nutzen können und trotzdem nur eine Codebasis implementiert werden muss, was den Hauptvorteil der webbasierten Variante darstellt.

Privatsphäre-Modell

Unabhängig von der Entwicklungsvariante sind Bedenken bezüglich der Privatsphäre von Nutzer*innen schon bei der Entwicklung von Apps zu berücksichtigen. In Vorarbeiten haben wir hierzu ein umfassendes Privatsphäre-Modell entwickelt (Beierle et al. 2018, Beierle et al. 2020, Beierle 2021). Die folgende Tabelle 4 zeigt die neun Maßnahmen des Modells PM-MoDaC (Privacy Model for Mobile Data Collection Applications). Im Folgenden werden wir diese erläutern.

(A)	User Consent
(B)	Let Users View Their Own Data
(C)	Opt-out Option
(D)	Approval by Ethics Commission / Review Board
(E)	Random Identifiers
(F)	Data Anonymization
(G)	Utilize Permission System
(H)	Secured Transfer
(I)	Identifying Individual Users Without Linking to Their Collected Data

Tabelle 4: Privatsphäre-Modell

(A) User Consent (Nutzer*innenzustimmung). Vor dem Installieren der App sollte den Nutzer*innen genau erklärt werden, welche Daten zu welchem Zweck gesammelt werden. Dies wird oft in den Datenschutzerklärungen dargestellt.

(B) Let Users View Their Own Data (Nutzer*innen ihre eigenen Daten ansehen lassen). Wenn der/die Nutzer*in die Möglichkeit hat, seine/ihre eigenen Daten anzusehen, kann er/sie eine bessere Entscheidung darüber treffen, welche er/sie davon teilen möchte.

(C) Opt-out Option (Abmeldeoption). Besonders nach dem Einsehen der eigenen Daten könnte der/die Nutzer*in entscheiden, nicht länger an einer Studie teilnehmen zu wollen. Diese Möglichkeit zur Abmeldung sollte jederzeit gegeben sein.

(D) Approval by Ethics Commission / Review Board (Genehmigung von einer Ethikkommission). Psychologische und medizinische Studien müssen typischerweise vor der Durchführung von einer Ethikkommission genehmigt werden.

(E) Random Identifiers (Zufällige Bezeichner). Wenn eine App gestartet wird, wird als Erstes oft ein Login des/der Nutzer*in erwartet. Dies birgt das Risiko, sensible Daten mit eindeutigen Namen zu verknüpfen. Die Idee von zufälligen Bezeichnern ist es, jede/n Nutzer*in nur über eine zufällige Zeichenfolge zu identifizieren.

(F) Data Anonymization (Datenanonymisierung). Sogenannte Hashfunktionen können beliebige sensible Zeichenfolgen, z. B. Telefonnummern oder Namen, deterministisch und unumkehrbar in andere, für Menschen bedeutungslose Zeichenfolgen abbilden. So ist der originale Text nicht mehr erkennbar. Dadurch, dass die Abbildung deterministisch ist, wird jedoch der gleiche Ursprungstext in den gleichen gehashten Text abgebildet. Zur Steigerung der Datensicherheit sollte diese Form der

Datenanonymisierung bereits beim Sammeln, also vor dem ersten Speichern der Daten angewendet werden.

(G) Utilize Permission System (Berechtigungssystem verwenden). Android und iOS, die beiden einzigen verbreiteten mobilen Betriebssysteme, bieten Softwareentwickler*innen Schnittstellen an, um auf potentiell sensible Daten zuzugreifen, zum Beispiel den Nutzer*innenstandort oder das Mikrofon des Smartphones. Je nach Betriebssystemversion bleibt dem/der Softwareentwickler*in dabei manchmal freigestellt, wann im Appverwendungsprozess eine Berechtigung angefragt wird. Typischerweise sollte die Anfrage möglichst spät stattfinden, also bevor die Daten wirklich für eine Appfunktion benötigt werden. Dabei sollte dem/der Nutzer*in stets erklärt werden, welche Daten wofür genau benötigt werden, dann kann er/sie entscheiden, ob er/sie zustimmen möchte.

(H) Secured Transfer (Gesicherter Datentransport). Die Verbindungen zwischen App und Servern sollte stets gesichert sein, sodass kein/e Dritte*r Zugriff auf die Daten hat.

(I) Identifying Individual Users Without Linking to Their Collected Data (Nutzer*innenidentifikation ohne die Daten mit dem/der Nutzer*in zu verknüpfen). In psychologischen Studien ist es typisch, dass Teilnehmer*innen mit Versuchspersonenstunden, Geld oder Gewinnspielteilnahmen entlohnt werden. Dafür ist es nötig, Nutzer*innen eindeutig zu identifizieren und kontaktieren zu können, was Maßnahme E widersprechen könnte. Um diese Bedenken zu beheben, haben wir einen Prozess entwickelt, der sowohl (1) überprüft, ob ein/e Nutzer*in erfolgreich an einer Studie teilgenommen hat und (2) es ermöglicht, Nutzer*innen zu kontaktieren, ohne die gesammelten Daten mit deren Kontaktdaten zu verknüpfen. Der Prozess besteht aus drei Schritten:

a. **Nutzer*innenregistrierung.** Um teilzunehmen, wird zusätzlich zum reinen Installieren der App eine E-Mail-Adresse angegeben, sodass der/die Nutzer*in kontaktiert werden kann. Die E-Mail-Adresse wird separat von den Daten des/der Nutzer*in gespeichert.
b. **Überprüfen der erfolgreichen Teilnahme.** Die App überprüft, ob der/die Nutzer*in erfolgreich an der Studie teilgenommen hat, z. B. indem überprüft wird, ob ein Fragebogen vollständig oder regelmäßig ausgefüllt wurde. Die App meldet den Status an das Backend des Systems. Jetzt kann das System die Nutzer*innen kontaktieren, die erfolgreich teilgenommen haben.
c. **Generierung von Teilnahmecodes.** Nach erfolgreicher Teilnahme kann der/die Nutzer*in als Bestätigung einen Teilnahmecode vom Backend erhalten. Mit diesem Code kann die erfolgreiche Teilnahme nachgewiesen werden. Diese Codes könnten zum Beispiel für das Vergeben von Versuchspersonenstunden verwendet werden.

Diese neun Maßnahmen aus PM-MoDaC können Entwickler*innen und Forscher*innen schon bei der Erstellung von Apps, die sensible Nutzer*innendaten sammeln, helfen, die Privatsphäre der Nutzer*innen zu schützen.

Referenzen

https://t3n.de/news/5-groessten-irrtuemer-hybrider-1294401/ (letzter Zugriff 29.04.2021)

https://anexia.com/blog/de/native-vs-hybride-apps/ (letzter Zugriff 29.04.2021)

https://www.ionos.de/digitalguide/websites/web-entwicklung/verschiedene-app-formate-was-ist-eine-web-app/ (letzter Zugriff 29.04.2021)

https://www.apple.com/de/accessibility/iphone/ (letzter Zugriff 29.04.2021)

F. Beierle, V. T. Tran, M. Allemand, P. Neff, W. Schlee, T. Probst, R. Pryss, and J. Zimmermann. "Context Data Categories and Privacy Model for Mobile Data Collection Apps". In: *Procedia Computer Science*. The 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC) 134 (2018), pp. 18–25. doi: 10.1016/j.procs.2018.07.139.

F. Beierle, V. T. Tran, M. Allemand, P. Neff, W. Schlee, T. Probst, J. Zimmermann, and R. Pryss. "What Data Are Smartphone Users Willing to Share with Researchers?" In: *Journal of Ambient Intelligence and Humanized Computing* 11 (2020), pp. 2277–2289. doi: 10.1007/s12652-019-01355-6.

F. Beierle. "Integrating Psychoinformatics with Ubiquitous Social Networking: Advanced Mobile-Sensing Concepts and Applications". Springer, 2021 (to appear).

Folgende Universitätskliniken des Netzwerks Universitätsmedizin nehmen am COMPASS-Projekt teil:

Charité – Universitätsmedizin Berlin
Universitätsmedizin Göttingen
Universitätsmedizin Mainz
Universitätsklinikum Würzburg
Uniklinik Köln
Universitätsklinikum Münster
Universitätsklinikum Regensburg
Universitätsklinikum Ulm
Universitätsklinikum Erlangen

Ansprechpartner für weitere Fragen:
COMPASS Koordinierungsstelle
compass@unimedizin-mainz.de

num compass

https://num-compass.science

@CompassNum

netzwerk universitäts medizin

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung