[E: Bewertung & Ausblick]

Ausgewählte Kompromisse

Der Entwurf der CWA geht bewusst Trade-offs ein und balanciert Qualitätsziele aus.

Explizite Freigabe positiver Testergebnisse durch Nutzer/in erforderlich

+ erhöht Vertrauen in die Lösung
- reduziert effektive Warnfunktionalität

Verteilte Anwendung auf dem Backend

+ gut für Datenschutz (Trennung der Daten)
+ verfügbar(er) im Fall von Teilausfällen
- schwieriger zu entwickeln und zu betreiben

Vergleichsweise hohe Kopplung der Microservices

+ einfacher umzusetzen, schneller am Markt
- erschwert unabhängige Entwicklung
- reduziert oder behindert Zuverlässigkeit

Ausliefern der Diagnoseschlüssel über CDN im Batch, Aktualisierung durch Apps in Intervallen

+ spart Ressourcen, vor allen an den Endgeräten
+ robust, erhöht Zuverlässigkeit
- Zeitverzögerung bei Risikoermittlung

Nächste Schritte für die CWA

Auf Basis dieser Lösungsarchitektur sind weitere Features angedacht, konzipiert oder bereits umgesetzt, z.B

- lokales Kontakttagebuch (Nutzung optional)
- Anzeige ausgewählter Kennzahlen zum Infektionsgeschehen (Bereitstellung: RKI)
- Eventregistrierung, Einchecken mit QR-Code, Erkennen sogenannter Cluster
- Integration von Schnelltestergebnissen
- Anzeige eines digitalen Impfnachweises

Weitere Informationen

..zur Corona-Warn-App:

Homepage des Open-Source-Projektes:
www.coronawarn.app/de/
Quelltexte und Dokumentation auf GitHub:
github.com/corona-warn-app/

..zum Anfertigen von Architekturüberblicken:

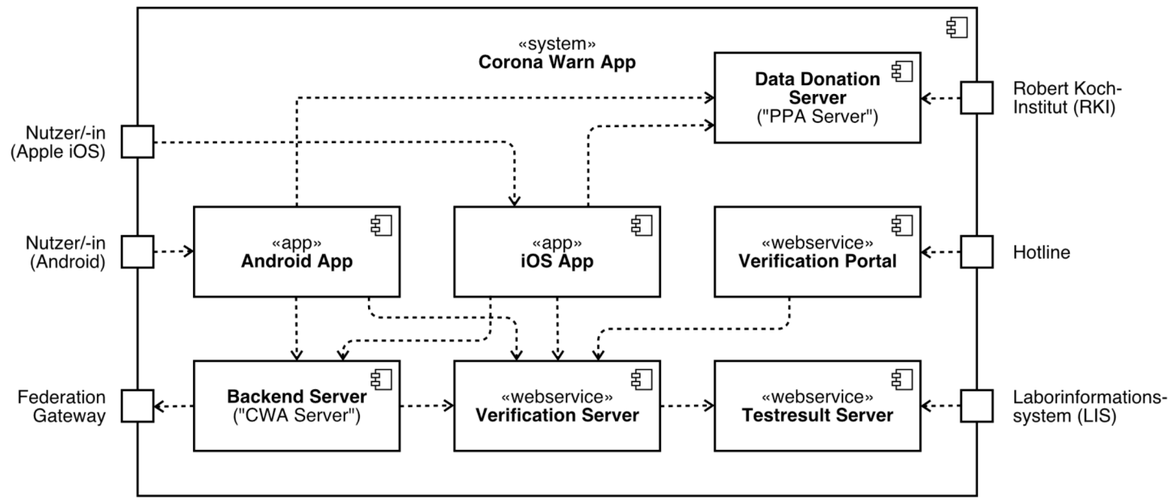
Gesammeltes Material (Videos, E-Books, ...)
www.embarc.de/architektur-ueberblicke/

embarc Software Consulting GmbH

[D: Eintauchen in die Corona-Warn-App]

Zerlegung auf oberster Ebene

Das Diagramm zeigt die wichtigsten Bausteine der CWA gemäss der Struktur des Quelltextes. Die Pfeile bedeuten Abhängigkeiten (A ⇢ B heisst "A verwendet B").



Die GitHub-Repositories liegen unterhalb von github.com/corona-warn-app/...

iOS App – Native App für iPhones. Gibt Testergebnisse frei. Ermittelt Infektionsrisiken anhand von Diagnoseschlüsseln. Spendet Daten.
.../cwa-app-ios/

Backend Server (aka „CWA Server") – Nimmt die Diagnoseschlüssel positiv Getesteter entgegen und teilt sie mit anderen Nutzern über ein CDN. Interagiert mit den Kontaktverfolgungen anderer Länder („Federation").
.../cwa-server/

Verification Server – Sichert ab, dass ein Nutzer der Meldung seines positiven Tests zugestimmt, und dass das Labor tatsächlich positiv getestet hat.
.../cwa-verification-server/

Android App – Native Android App, analog zur iOS App.
.../cwa-app-android/

Data Donation Server – Nimmt Nutzerdaten bei aktivierter Datenspende entgegen und speichert sie, ohne Rückschlüsse auf individuelle Personen zuzulassen. (PPA = Privacy Preserving Analytics).
.../cwa-ppa-server/

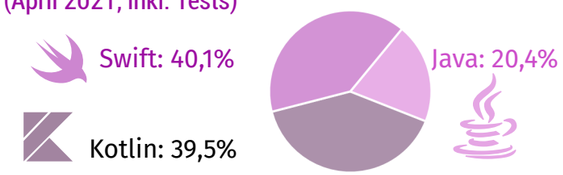
Verification Portal – Ermöglicht die Erzeugung von teleTANs im Verification Server über ein einfaches Browser-Frontend.
.../cwa-verification-portal/

Testresult Server – Empfängt und speichert die Testergebnisse von angeschlossenen Laboren.
.../cwa-testresult-server/

Ausschnitt Technologie-Stack

Die Lösung verwendet diverse Programmiersprachen, Bibliotheken, Frameworks und Middleware.

Quelltext insgesamt 206.560 Lines of Code (100%)
(April 2021, inkl. Tests)



Client (Native Apps)
- iOS App programmiert in Swift
- Android App programmiert in Kotlin
- Persistenz mit SQLite
- Verteilung über Google Play (Android) und App Store (Apple iOS)

Server (Backend)
- Services programmiert in Java mit Spring Boot, Spring Cloud, Spring Data ...
- weitere Open Source Bibliotheken (u.a. Lombok, Guava, Liquibase, Micrometer ...)
- Kommunikation mit REST/OpenAPI und protobuf
- Persistenz mit PostgreSQL

Infrastruktur
- Build mit Maven, Gradle und fastlane
- Docker, Docker Compose (zu Testzwecken), in Produktion auf Kubernetes
- S3 und Content Delivery Network (CDN)

Die deutsche Corona Warn-App

Ein prägnanter Überblick über die Softwarearchitektur der Gesamtlösung - Apps & Backend

Stand: April 2021

Architekturüberblick

embarc Software Consulting GmbH
Hamburg // Wien
info@embarc.de
www.embarc.de

embarc Software Consulting GmbH

[A: Aufgabenstellung]

Mission Statement

Was ist die Corona-Warn-App?

Die Corona-Warn-App (CWA) hilft Infektionsketten des SARS-CoV-2 (COVID-19-Auslöser) in Deutschland nachzuverfolgen und zu unterbrechen.

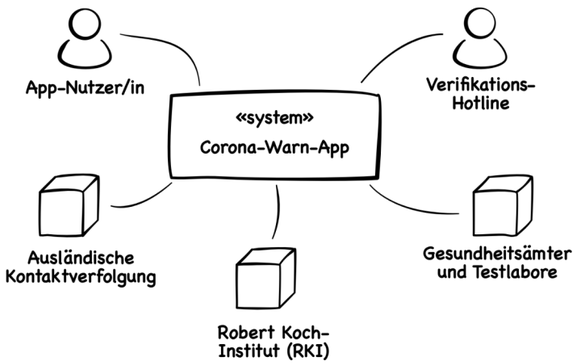
Sie basiert auf Technologien mit einem dezentralisierten Ansatz und informiert Personen, wenn sie mit einer infizierten Person in Kontakt standen.

Transparenz ist von entscheidender Bedeutung, um die Bevölkerung zu schützen und die Akzeptanz zu erhöhen.

www.coronawarn.app/de/

Fachliche Kontextabgrenzung

Das CWA-Gesamtsystem im Zusammenspiel mit den wichtigsten Benutzern und Fremdsystemen.



App-Nutzer/in – Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.

Verifikations-Hotline – Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse ("TeleTan").

Ausländische Kontaktverfolgungen – Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.

Robert Koch-Institut (RKI) – Stellt Content für die App zur Verfügung und bestimmt Parameter für Risiko-Ermittlung. Empfängt Auswertungen, etwa aus der Datenspende.

Gesundheitsämter und Testlabore – Liefern anonymisierte Testergebnisse an das System.

[B: Einflüsse]

Rahmenbedingungen

Ausgewählte Maßgaben an Entwicklung und Betrieb sowie Informationen zum politischen Umfeld.

Technische Vorgaben

- Entwicklung von nativen, mobilen Clients für Android- und iOS-Smartphones
- Verfolgen eines dezentralen Ansatzes für die Datenspeicherung
- Einsatz des Exposure Notification Framework von Google und Apple
- Betrieb der Backend-Komponenten in der Open Telekom Cloud

Organisatorischer Rahmen und Umfeld

- Auftraggeber: Deutsche Bundesregierung
- Entwicklung und Betrieb durch ein Konsortium aus zwei Auftragnehmern (SAP und Deutsche Telekom)
- Start der Entwicklung 04/2020
- enger Zeitrahmen (Apps verfügbar ab 06/2020)
- hoher politischer Druck, viele Parteien involviert (Ministerien, Behörden, RKI ...)
- große Medienaufmerksamkeit
- gewisse Skepsis innerhalb der Bevölkerung

Top-Qualitätsziele

Die vorrangigen Architekturziele der CWA in der Reihenfolge ihrer Wichtigkeit.

Höchster Datenschutz

Der Schutz der personenbezogenen Daten hat oberste Priorität.

Effektive Warnfunktionalität

Die App ist ein effektiver Baustein bei der Pandemie-Bekämpfung.

Attraktive Lösung für App-Nutzer

Die App ist leicht zu installieren sowie intuitiv und effizient zu bedienen.

Hohe Zuverlässigkeit

Die Lösung geht mit Lastspitzen wegen hoher Nutzer- oder Infektionszahlen ebenso souverän um, wie mit Störungen und böswilligen Angriffen.

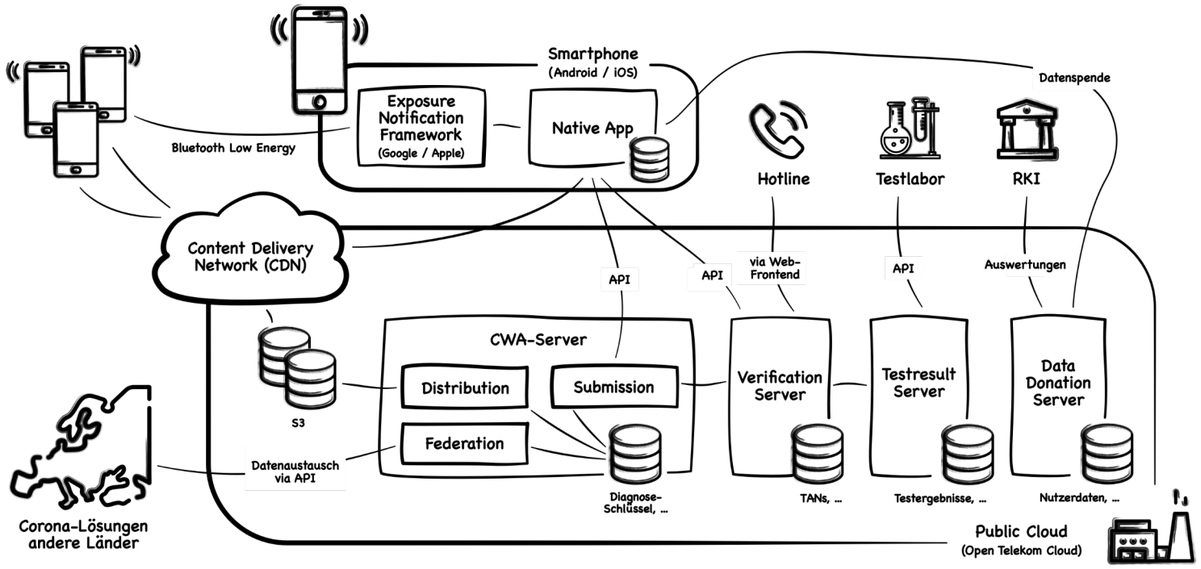
Gute Wartbarkeit

Die Software lässt sich leicht anpassen, wenn z. B. Nutzer/-innen, Politik oder neue Forschungsergebnisse es erfordern.

[C: Lösungsstrategie]

Informelles Überblicksbild

Schematische Darstellung mit wesentlichen Lösungsteilen des CWA-Systems sowie wichtigen Beteiligten und deren Interaktion.



Entscheidende Lösungsansätze

Die aufgelisteten Prinzipien, Muster, Konzepte und Technologien adressieren die jeweiligen Top-Qualitätsziele und begünstigen deren Erreichung.

Höchster Datenschutz

- Dezentraler Ansatz, Speicherung der Daten lokal beim App-Nutzer
- Datensparsamkeit (nur minimale, notwendige Informationen werden erhoben)
- Verschlüsselung aller Bewegungsdaten
- Teilen von Daten nur nach Aufforderung
- Transparente Entwicklung, Quelltexte einsehbar

Effektive Warnfunktionalität

- Digitale Abläufe zur Risikowarnung
- Unterstützung der gängigsten Smartphones (mehr als 90% aller Geräte)
- Verwendung des Exposure Notification Frameworks

Attraktive Lösung für App-Nutzer

- Native Apps für iOS und Android
- Verteilung durch die jeweiligen Stores
- übersichtliche Gestaltung, simple Bedienung
- Geringer Ressourcen-Verbrauch, u.a. durch Bluetooth Low Energy und Datensparsamkeit

Hohe Zuverlässigkeit

- Robuste Lösung, Apps erfassen Begegnungen ohne Verbindung zum Backend
- Backend in der Public Cloud, Orchestrierung mit Kubernetes
- Modularisierung des Backends in unabhängige, skalierbare Services
- Bereitstellung von zu lesenden Daten über Content Delivery Network

Gute Wartbarkeit

- fachlich getriebene Zerlegung der Apps
- Modularisierung des Backends in Microservices mit separater Datenhaltung
- Verwendung verbreiteter Standard- und Open Source-Bibliotheken
- Entwicklung als Open Source, gute Dokumentation
- Tool-gestützte Überwachung der Code-Qualität