eHealth Network

Guidelines on

Technical Specifications for Digital Green Certificates Volume 4

European Digital Green Certificate Applications

Version 1.3

2021-04-21

The eHealth Network is a voluntary network, set up under article 14 of Directive 2011/24/EU. It provides a platform of Member States' competent authorities dealing with eHealth.

The Digital Green Certificate, supporting digital and paper versions, enables EU-wide verification of vaccination, test, and recovery certificates for EU citizens to facilitate safe travel and free movement across the EU. The Digital Green Certificate Application (DGCA) enables issuing, holding, and verification of Digital Green Certificates (DGC).

Adopted by the eHealth Network on 21 April 2021.

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1 Introduction

1.1 Context

DGCA comprises the issuer app, the wallet app, the verifier app, and assorted backends which could be used by member states. Although TSI/SAP will provide reference implementations, the responsibility for duly diligent realization, operation, and maintenance remains with the member states. As member states remain free to develop their own apps, therefore this document is not to be considered as a normative specification. The DGC Gateway (DGCG) is described in a separate document.

1.2 Scope of Document

This document describes the issuer app, wallet app, verifier app, and core functionalities of the required national certificate backends as required by EU-wide DGC verification.

Section 2 outlines the general approach and the underlying assumptions.

Section 3 details the necessary user stories.

Section 4 contains information about the data structures and syntactic conventions.

Section 5 outlines communication flows within the DGCA scheme.

Section 6 shows building block views of the DGCA components.

Lastly, section 7 contains detailed information about recommended interfaces of the DGCA components, section 8 informs about recommended frameworks and technology, and section 9 shows a general deployment perspective.

2 Architecture Overview

2.1 Approach

The Digital Green Certificate Application (DGCA) described in this document consists of three components, roughly corresponding to the three user stories ("issuing, transfer to app, and verify a DGC") detailed in section 3.

2.2 Assumptions

Here are the fundamental assumptions underlying DGCA:

1. Each EU member state provides one or more **national certificate backends** (often just called "backend" below) whose job is DGC signing, TAN generation, TAN validation, public key publication, and DGC validation/revocation inf, etc.
2. Issuing certificates is done via the **issuer app**, which is created and access-controlled by each member state.
3. Exchange of public keys, X509 certificates, and other crypto material between the national backends is facilitated by the **DGC Gateway (DGCG)**, which is described in a companion document and operated by DIGIT[1].
4. Each participating member state is free to provide a **wallet app** that holds DGCs. Nevertheless, cross-national sharing of wallet apps is possible, in case a MS decides to reuse the app of another MS.
5. Each participating MS is free to provide a **verifier app** that can be used to verify DGCs issued by any participating EU country. Access management of the verifier app is up to each member state; cross-country sharing of the same verifier app is possible. Verifier apps must be able to verify DGCs without internet connectivity (offline verification; see section 3.3). The verifier app must neither store nor forward any details of scanned DGCs.
6. Transferring the Green Certificate to the wallet app requires a **second factor**, preferably a perishable TAN, to prevent impersonation and certificate theft.
7. The whole setup—issuer apps, wallet apps, verifier apps, and backends, but without DGCG—is summarily called DGCA.
8. The wallet app could—but need not—be structured such that **selective disclosure** of DGC details is possible. Note that this requires a dedicated backend interface which creates temporary signatures.
9. A note on **key rotation**: National backends are expected to upload new public keys regularly. Moreover, legacy key pairs must remain valid at least as long as the DGCs signed with them, but not much longer. This way, the total amount of crypto material remains within reasonable bounds.

2.3 International interoperability

The Digital Green Certificate technological solutions should seek to ensure interoperability with relevant global initiatives, in particular the World Health Organization (WHO) and the

[1] https://ec.europa.eu/info/departments/informatics_en

International Civil Aviation Organization (ICAO). This may include interoperability features with 2D barcodes following other recognized international standards/formats, for instance by allowing for adequate conversion between certificate formats.

3 User Stories

There are three major user stories to be considered—issuing a DGC, transferring it to the rightful user's app, and verifying it. These user stories are described in more detail below.

Security and trustworthiness of a DGC depends on two things:

(1) The issuing authority is assumed to have undergone a **rigorous onboarding procedure**; otherwise, it would not be able to publish its crypto material via DGCG.

(2) DGCs must be authenticated using **a second factor** (in addition to the correct signature) **to prevent impersonation** (i.e., copying someone else's DGC and pretending it to be one's own).

The attentive reader will note that use cases below are modeled along the issuer-holder-verifier triangle—DGCs are *issued* by a health authority or a delegate thereof; DGCs are *held* by EU citizens in a wallet app and/or on paper; DGCs are *verified* by authorized personnel. Verifiers and issuers are connected by the DGC Gateway (DGCG), which acts as a trust anchor, i.e., trust in DGCG is presumed and cannot itself be proven. Relevant crypto material for verification is provided by the issuers via DGCG to the verifiers, so that issuers will never know who verified whom.

3.1 User Story 1: Issuing a Green Certificate

DGCs will be issued by properly authorized delegates of the national health authorities, e.g., doctors and test center staff, upon three occasions (note that, in practice, these scenarios differ mostly by the expiration date of the DGC):

1. A COVID-19 vaccination
2. A test result
3. Recovery from COVID-19

If one of the above is true, the health authority delegate uses a country-specific issuer application to fill out a DGC form containing[2]:

- Identity data (name, date of birth, etc.)
- Green certificate type (vaccination, test result, or recovery)
- Certificate details (vaccination type, validity, etc.)

Note that the generating issuer app assigns a universally unique ID to the DGC, the DGC Identifier (DGCI). The backend, then, signs the DGC with a private key (there could be more than one, but not less). This makes the DGC verifiable, since the backbend's public keys are published via DGCG. If a TAN is used as a second factor (which is not the case in the first version of DGCA), this TAN is generated during the signing request and returned together with the signed DGC. The TAN, then, is sent to the holder via SMS or email. Alternatively, the TAN

[2] The exact dataset, including technical description, is provided by the eHN.

can be printed and handed out in addition to the digital green certificate or sent directly from the backend to the user via email or SMS.

The holder receives their signed certificate on paper or via email. Together with the TAN (or other second factor), the certificate can be transferred to the wallet app (see next use case) and is then linked to a key pair, which is generated and stored on the user's smart phone. Incidentally, it might seem strange to talk of a printed letter as "digital" (the "D" in DGC), but note that even this letter contains a QR code ready for electronic verification.

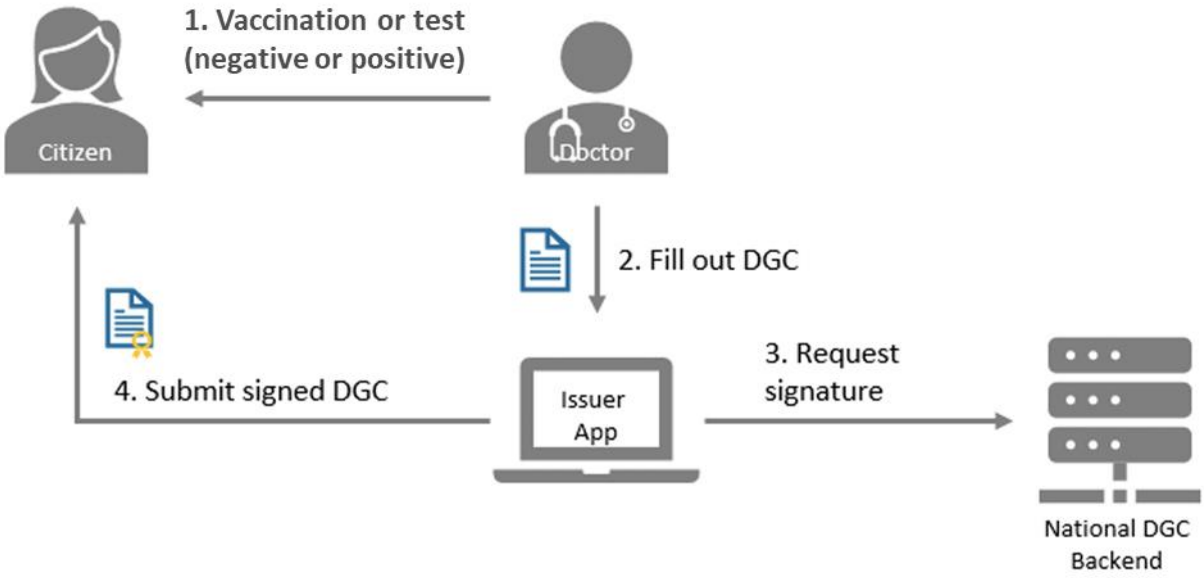

Figure 1: User story 1, issuing a DGC

3.2 User Story 2: Import a Green Certificate in the Wallet App

Initially, the Green Certificate is a piece of paper (or a PDF) containing human-readable and machine-readable information. The machine-readable portion is printed as a QR code. After the user has installed a DGC wallet app, this QR code needs to be scanned. After that, the second factor (e.g., the TAN) needs to be entered into that app. If a TAN is used, it is invalidated after use to prevent reuse. The unique DGCI, the DGC signature, the second factor, and the public key of a newly app-created key pair are then sent to the corresponding backend. Only if the backend accepts this registration and marks the DGCI as registered, the wallet app integrates the DGC. This makes the DGC uniquely identifiable and prevents multiple apps from using the same DGC.

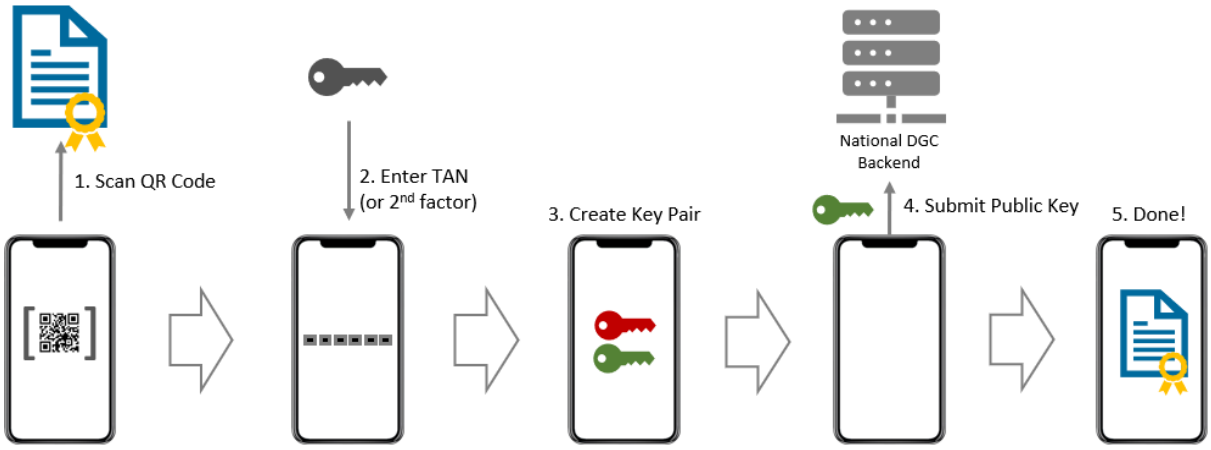

Figure 2: User story 2, transferring a Green Certificate to the wallet app

Optionally you can use a deep link instead of a 2D Code to initiate the certificate import in the wallet app. The deep link can look like:

dgc://example.authority.com?token=ey… & [publickey]

In this case the token is received with the link, and the public key must be replaced by the key of the new generated key pair of the certificate container in the wallet app. The deep link can be delivered by SMS, Email or by presenting another 2D Code for scan.

3.3 User Story 3: Verify a Green Certificate Offline

Offline verification presumes a verifier app that has up-to-date crypto material, most importantly, public keys from the national health authorities. Since these keys change only rarely, daily updates of the verifier app's key store via the backend (which gets fresh keys from the DGCG) are sufficient. In most cases—that is, assuming the public key needed for verification has already been downloaded by the verifier app—verification works without active internet connection. Hence the name, offline verification.

Please note that both the paper DGC and the in-app DGC are viable options for citizens—each contains the QR code needed for verification.

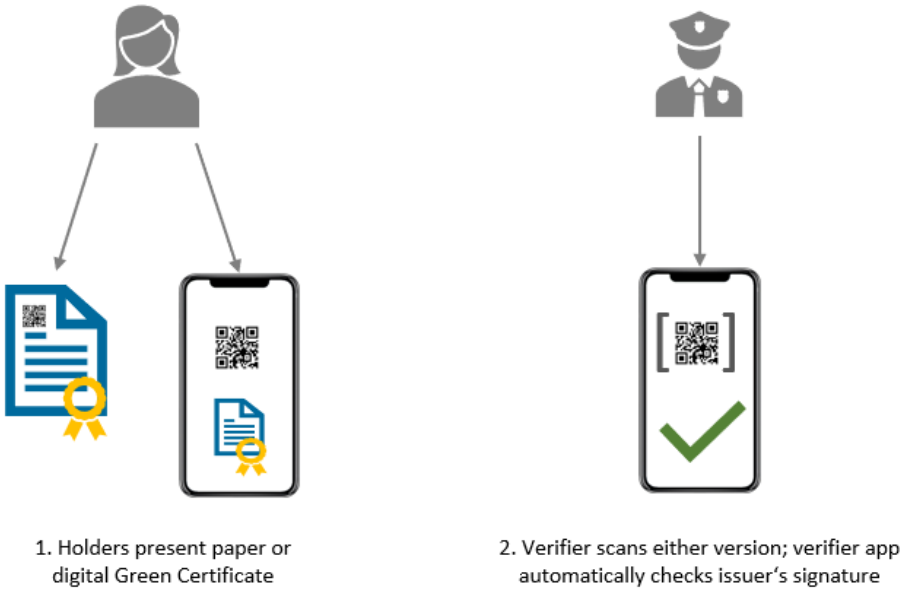

Figure 3: User story 3, offline verification

Note that **a glance at the holder's ID/passport is indispensable** to prevent impersonation; here is the flow in full resolution:

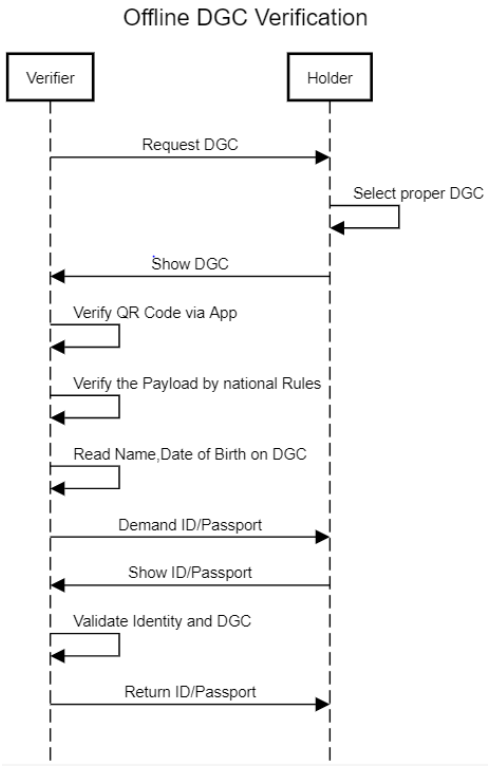

Figure 4: Offline DGC verification, the flow

3.4 User Story 4: Verifying DGC by an airline

[To be completed in a next version of the document.]

4 Data Structures

4.1 Context Documents

The context documents in the backends are downloaded from the DGCG and flatted in single context documents:

```
{
              "version":”HC1”,
               "type":"Vaccination",
               "schema": {},
               "valuesets":[
                {
                 "fieldname":"aut",
                 "values":["AstraZeneca AB", "Biontech Manufacturing GmbH","…"]
                }
              ],
              "mapping":{}
}
```

Figure 5: Flatted context document

4.2 Trust Lists

The DGCA uses no special trust lists. The information is directly fetched from the DGCG and cached. The verifier applications check the signatures of the received certificates.

The trust list provisioning to the verifier apps, is completely free to choose by the member states. Bilateral Exchanges are not forbidden.

Alternatives:

1) A member state can fetch the information from the DGCG per script and can create another representation of the certificates. E.g. building lists of public key parameters, JWK Representations and others.
2) To follow the ICAO philosophy, the certificates / trust anchors can be downloaded and packed to country specific master lists which are provided then to verifiers in different formats.

5 Communication

5.1 TAN Generation

The default second factor for securing DGCs and protecting their transfer to the wallet app is a TAN—a random transaction number that has a limited shelf life, is invalidated immediately after use, and is also invalidated after three (or another fixed number) of incorrect attempts.

Note, first, that the TAN is created in conjunction with the DGC signature by the backend that does the signing and stored together with the Green Certificate's unique identifier, the DGCI. The TAN could be any kind of secure random number, but it is crucial to bear in mind that ordinary citizens must be able to enter it into their smartphone without undue hassle. Therefore, we recommend a 9-digit TAN, using only uppercase letters and numbers, omitting the letters "I" and "O" and the numbers "1" and "0" to avoid confusion. A tenth checksum digit makes sense to capture typing errors on the phone and not using up the small number of retries granted by the backend.

Second, the TAN could be returned in conjunction with the signed DGC (as shown in the flow diagram below) or sent directly to the user's phone. The latter option presumes that the backend knows the user's email address or phone number.

Third, the expiration period of the TAN must be carefully set—too small, and many users will fail to digitize their Green Certificate; too large, and users who don't intend to digitize their Green Certificate will possibly be prey to impersonation attempts and Green Certificate theft, especially if they're gullible and hand out their TAN to fake verifiers. We recommend an expiration period of two hours.

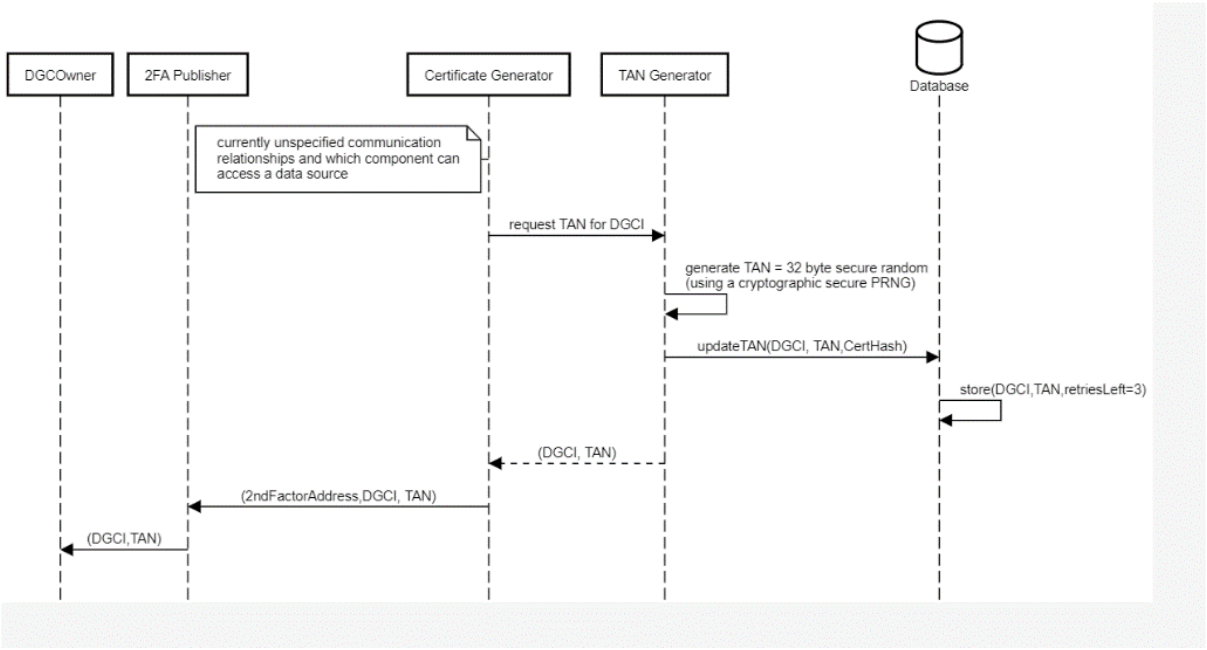

Figure 6: TAN Generation

5.2 TAN Validation

TAN validation is an easy matter—upon scanning a DGC, the wallet app creates a cryptographic key pair. Then, the TAN and the DGCI are signed with the newly created private key and uploaded together with the corresponding public key. The certificate backend checks the signature and verifies whether

1. The DGCI exists
2. There haven't been more than the specified number of TAN validation requests
3. The submitted TAN corresponds to the TAN stored together with the DGCI
4. The stored TAN is not expired

If all these points are positively answered, TAN validation has been successful, the user's public key is stored together with the DGCI, and the corresponding DGC is marked as "registered" (meaning it can't be registered again—a digitized Green Certificate can't be digitized by any other wallet app). Otherwise, an appropriate error code is returned.

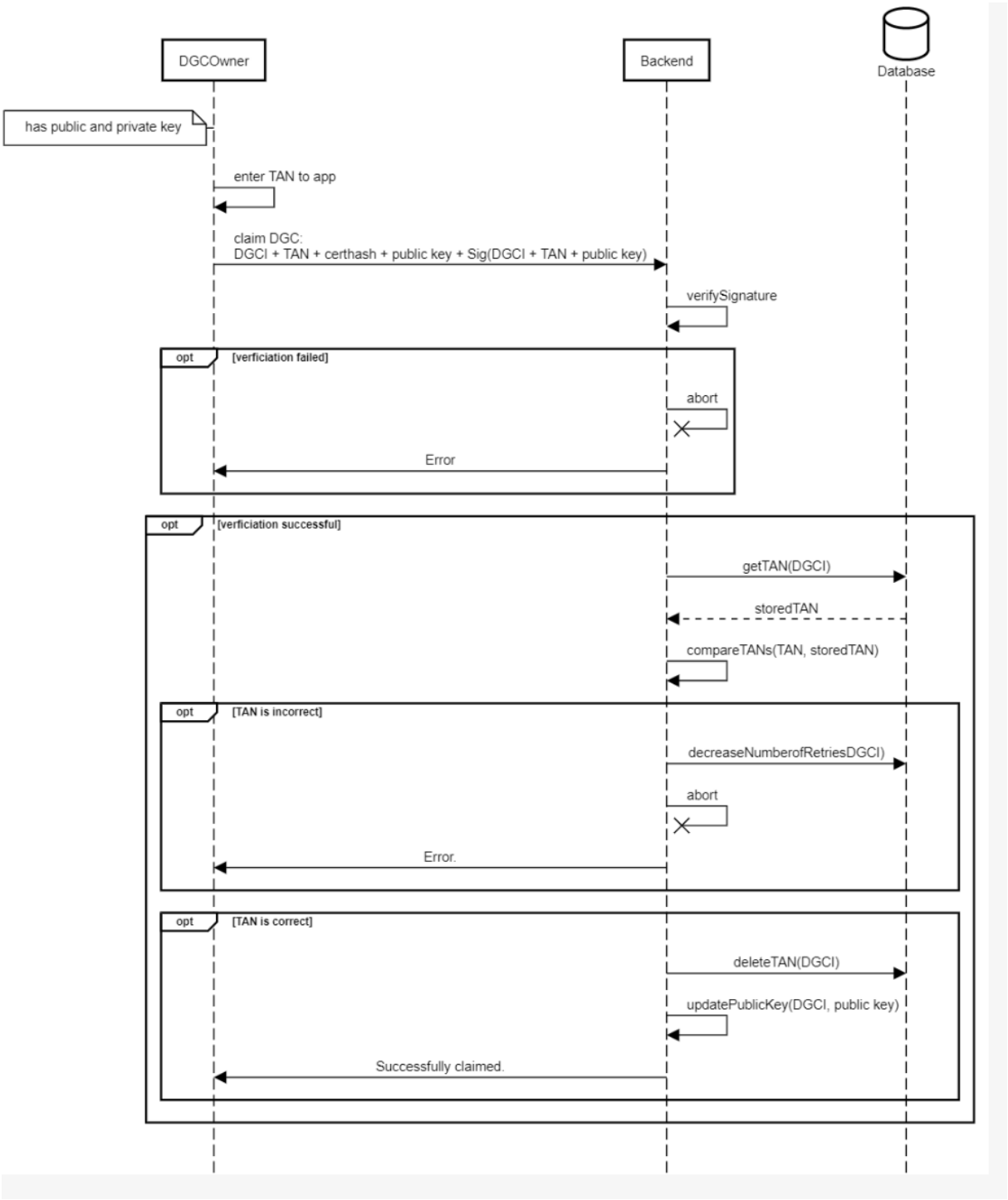

Figure 7: TAN Validation

5.3 Issuing Certificates

Green Certificates will be issued by specifically authorized delegates of the national health authorities of the member states, e.g., doctors, lab personnel, or test center personnel (called “issuer” henceforth). The issuer uses an issuer app (see section 6.2), usually a web application provided and hosted by each member state for the benefit of the aforementioned delegates.

The issuer inserts user data in the issuer app, most notably

- Identity information (name, date of birth)
- Green Certificate type (vaccination, test, recovery status based on a positive test)
- Certificate details (vaccination type, test type, etc.)
- Expiration date

These values are hashed using SHA-256 and then uploaded to the backend, where the “raw” Green Certificate is being equipped with a newly created universally unique identifier—the DGCI—and signed with the newest private key of the national backend. The signed Green Certificate and a newly created TAN are returned (see TAN generation in section 5.1 above) either directly to the user (release 2) or to the issuer app frontend (release 1) and the paper certificate is handed out and/or emailed to the citizen.

The QR Code is created according the specification of the 2D interoperable Barcode.

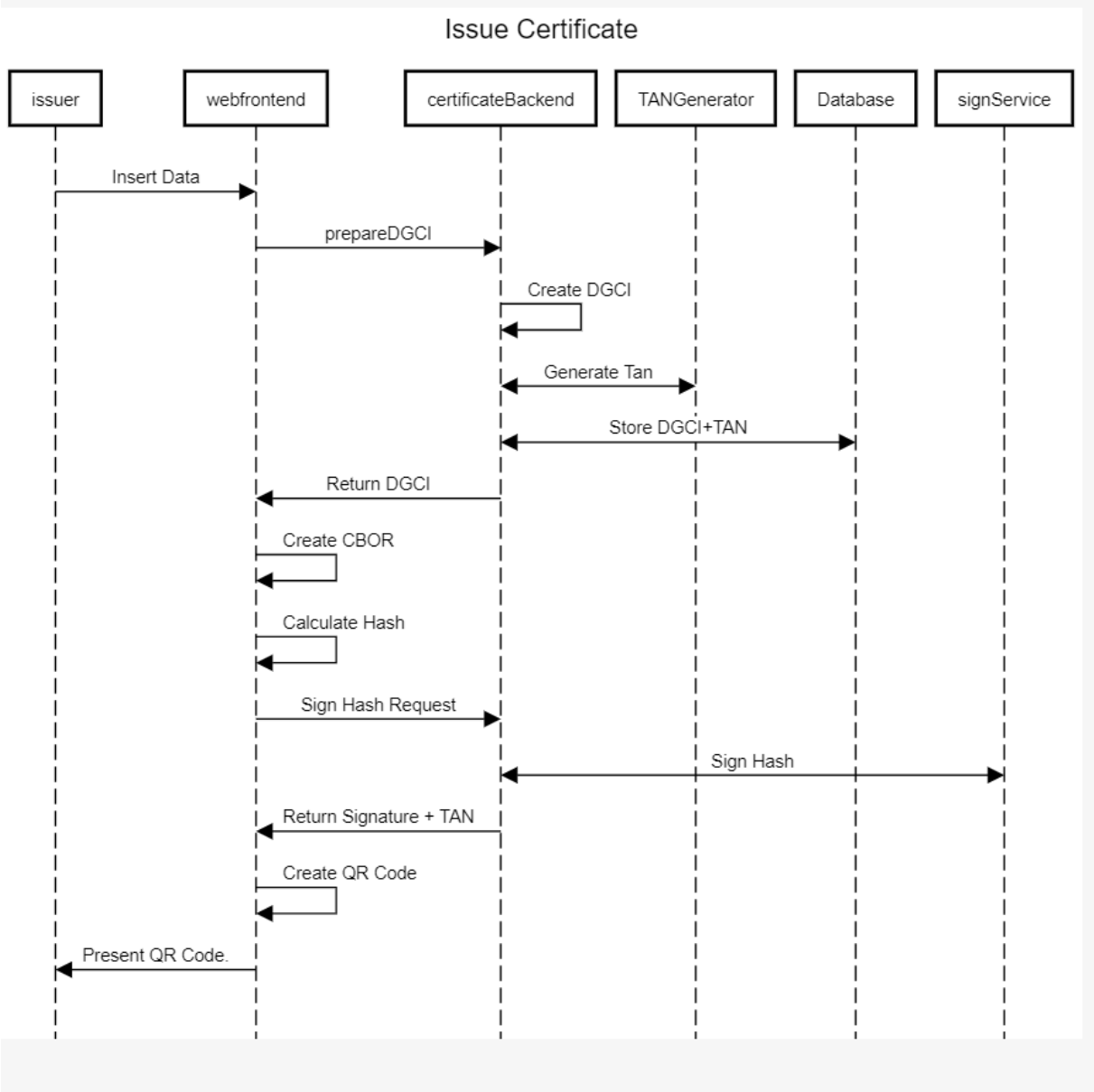

Figure 8: Issuing certificate Flow

5.4 Verifier App Synchronization

To enable offline verification, the verifier app needs to retrieve all public keys from all national backends that it hasn't stored already. After app installation, all keys have to be downloaded in bulk (assuming unlimited key lifetime, monthly key rotation, and 30 participating countries, this amounts to only 1800 public keys after 5 years—less than one megabyte). Since the verifier app's key cache needs to be in sync with the national backends' key cache, this process is called "synchronization." It's recommended to fetch the keys immediately from the DGCG cache to avoid gaps in the synchronization (usage of resume token).

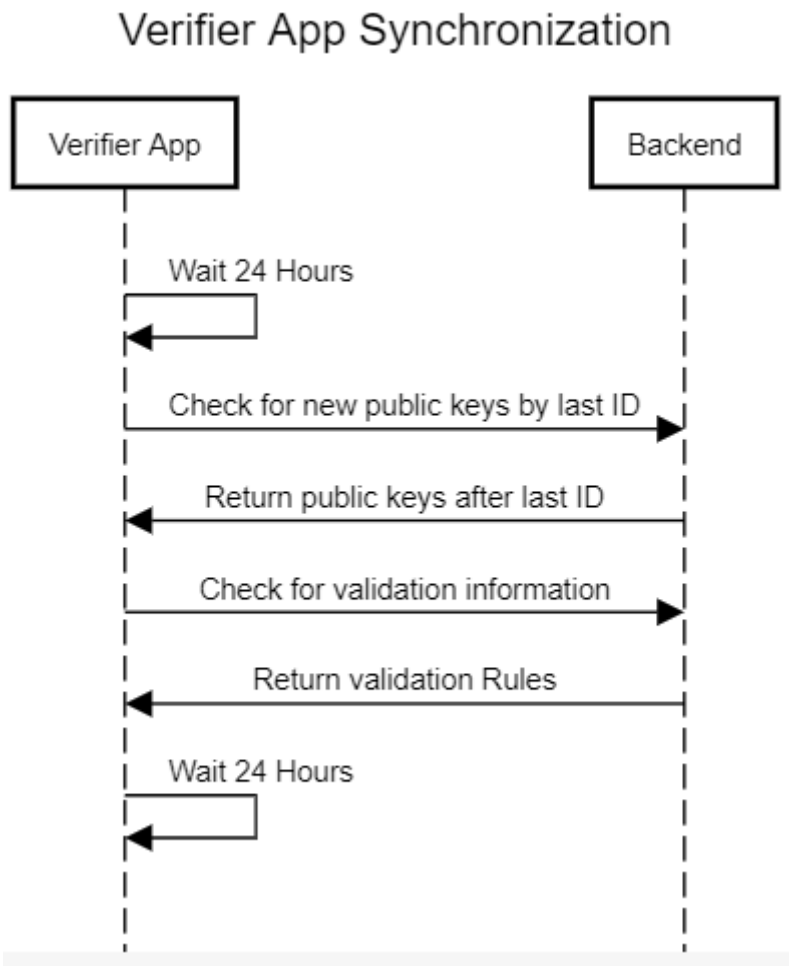

Figure 9: Verifier App Synchronization

6 Building Block Schema

6.1 Overview

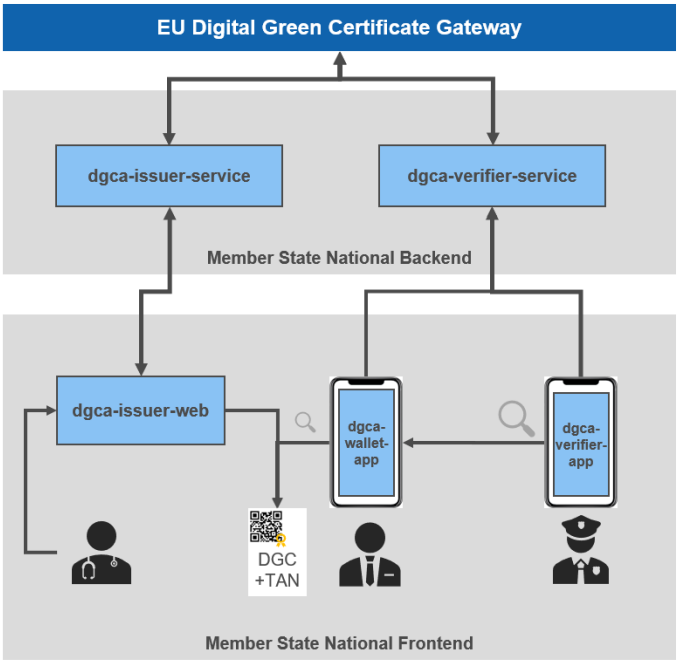

Figure 10: DGCA architecture overview

The DGCA consists of the issuer application (frontend and backend), verifier application (frontend and backend), and the wallet app. The DGCA is provided as a template for member state and will be hosted and managed by each member state. Each application is described in detail in the following section.

This chapter contains just a high-level description of the included application. The detailed design specifications of the applications will be generated during the development.

6.2 Issuer Application

6.2.1 Frontend

The issuer application frontend provides a user interface that is used by the issuer to enter the necessary data (see section 3.1). All personal information is kept local in the issuer frontend application (privacy by design).

The entered personal information is SHA-256 hashed (triggered by the React[3] user interface) in the frontend's crypto component and only those hashes are provided to the issuer service by the signing client for the remote signing process—personal information never leaves the frontend. The signing client uses the signature and compression component to process the

[3] https://reactjs.org/

serialization of the DGC and integrates the signature with COSE. Finally, the signed dataset is used for the generation of the 2D barcode.

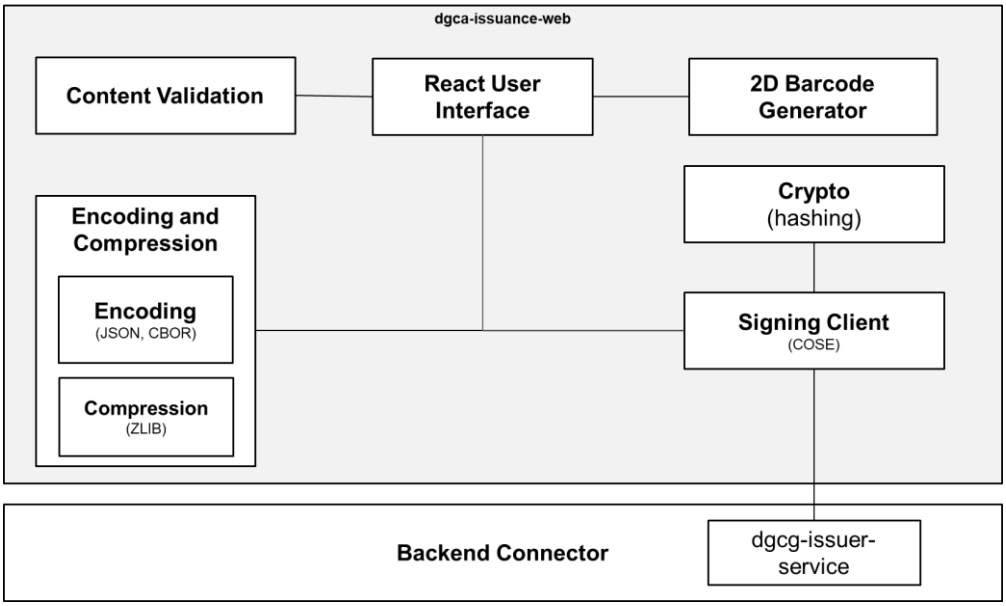

Figure 11: Issuer application frontend building blocks

6.2.2 Backend

The issuer backend is accessed by the issuer frontend application and the respective wallet apps of the same MS. The backend itself publishes its public keys to the DGCG where they can be distributed to other MS. Each MS hosts its own issuer backend. The main function of the backend is to provide services for creating and signing new green certificates. The backend consists of the following building blocks:

- The API gateway manages authentication and communication with the issuer application frontend, DGC Gateway and DGCA wallet apps.
- A Database to store DCGIs and corresponding TANs/public keys of the holder.
- The signing service handles the signing and crypto operations, related to the SHA-256 hash received by the issuer frontend as well as triggering the generation of the corresponding DGCI and TAN. Additionally, it fetches the required DGC metadata from the certificate service and composes the signature response for the frontend consisting of the signed hash, certificate metadata, DGCI, and TAN.
- The DGCI service generates the unique DGCI called by the signing service and stores the DGCI in the database.
- The TAN service generates the TAN and stores it in the database.
- The certificate service is responsible for managing the private and public keys of the backend. It publishes the public keys to the DGGC.

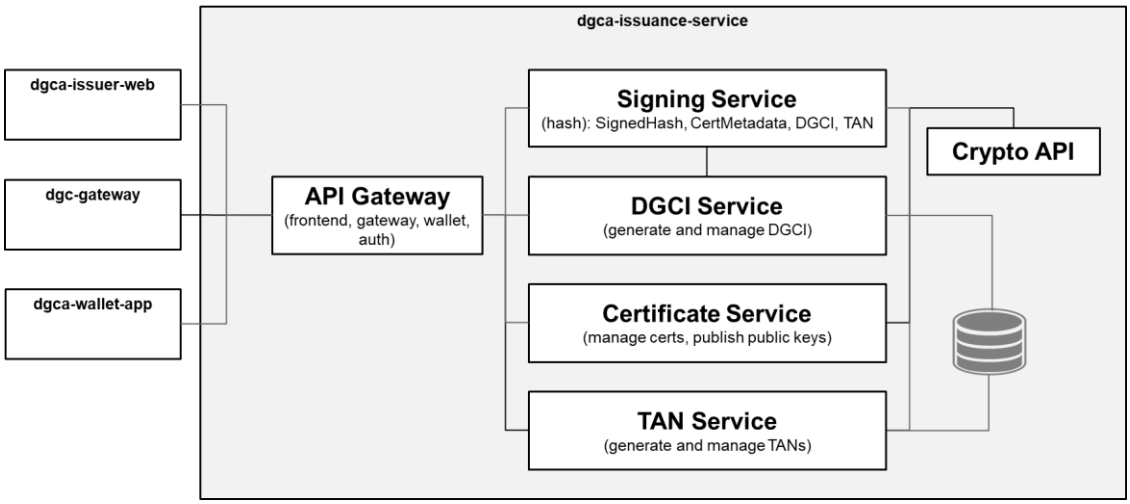

Figure 12: Issuer application backend building blocks

6.3 Verifier App

6.3.1 Frontend

The verifier app frontend provides functionality to scan and verify DGCs. It scans the base45-encoded QR code, extracts the COSE signature, and decodes CBOR back to JSON (see also 6.2.1). It then verifies the signature with the keys provided by the verifier app's backend. The app uses only open-source libraries; all DGCs scanned or processed are ephemeral and will not be stored.

The result of the verification should be shown in such a way that only the minimum required information is displayed to the user of the verifier app in the standard verification workflow. In case of successful verification, the information should be limited to the indication that the authenticity and validity have been verified successfully (GREEN), and minimum personal details necessary to link the certificate to the holder. In case of a failed verification (RED), the app should only display the reason for the fail, including details such as the specific contents or technical details of the certificate that are preventing the successful verification.

Figure 13: Core components of the verifier app

The verifier app realizes a workflow of the following steps during verification:

- Scanning of QR Code[4]
- Base45 Decoding
- COSE Signature Extraction
- CBOR-to-JSON Decoding
- Fetching the Signature's Public Key
- Verification of COSE Signature with Public Key
- Verification of CBOR Content, Comparison with Deny-List*

All codes are scanned with the open-source Zebra Crossing (zXing) Library[5]. The decoding of the content is done by base45[6] decoding and CBOR libraries, with the COSE signature validation by iOS-internal encryption libraries. All scanned green certificates are ephemeral and will not be stored. The *public key store* needs to be protected against tampering and malicious key injection, which is realized via biometric data (e.g., Touch ID/Face ID or similar).

The validation rules are written in JavaScript which are deployed to the app and executed in the code. Manual how to generate validation rules will be delivered during the development.

[4] https://www.qrcode.com/en/about/standards.html
[5] https://github.com/zxing/zxing
[6] https://github.com/ehn-digital-green-development/base45-swift

6.3.2 Backend

The verifier backend basically caches the public keys that are distributed through the DGCG for the MS and provides the Trust List of certificates. It is accessed by the verifier apps to update the key store periodically (see section 3.3).

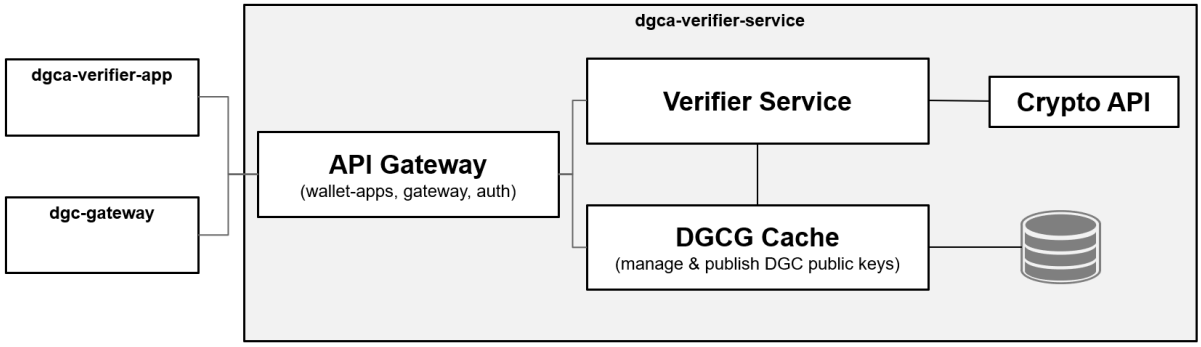

Figure 14: Verifier application backend building blocks

6.4 Wallet App

6.4.1 Frontend

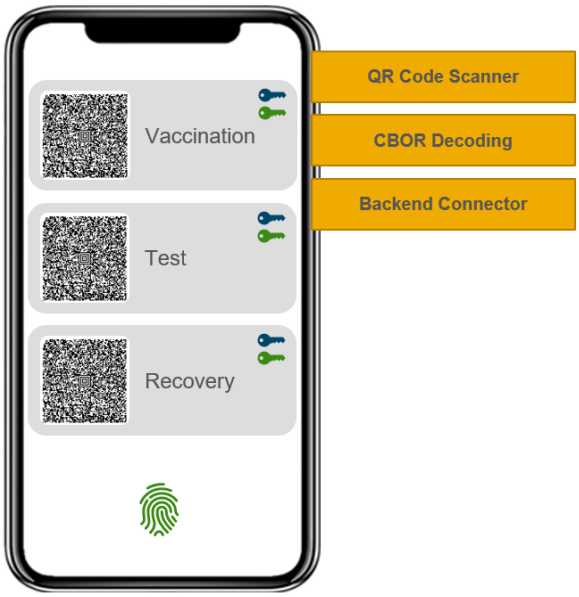

Figure 15: Wallet app frontend components

The wallet application frontend provides a user interface to store and manage personal DGCs directly on the phone. DGCs will be imported by scanning a base45-encoded QR code and decoding CBOR to JSON. Afterwards, it is symmetrically encrypted in the app's sandbox and the symmetric key is stored in the system's keychain. Multiple DGCs can be stored in the app. Access to the app is controlled via biometric data (e. g., Touch ID or Face ID).

The wallet app frontend can display any imported DGC as QR code for scanning and verifying with the verifier app.

The wallet app is developed on the same basis as the verifier app.

6.4.2 Backend

The wallet apps use the issuer application backend as their backend. See section 6.2.2.

7 Interfaces

7.1 Issuer API

Figure 16: API Overview of Issuer Application

7.2 Verifier API

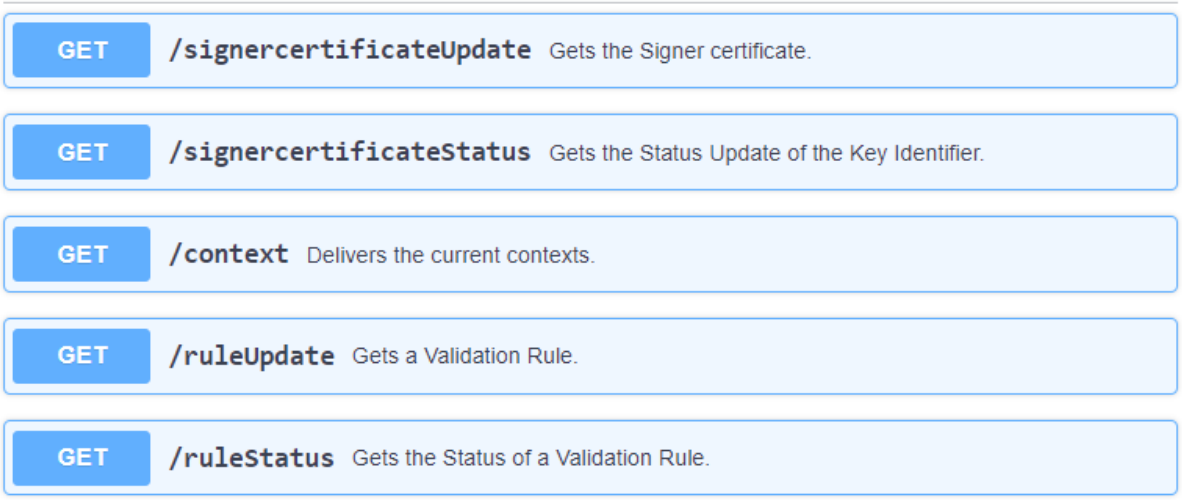

Figure 17: API Overview of Verifier Application

8 Technology Choice

Component	Technology	Core Features
REST API	Java / Spring Boot	Powerful, versatile web framework
Database	Postgres	Open Source Relational Database
Load Balancer	Traefik	Reverse proxy, load balancing, detailed traffic metrics, SSL offloading, Client Auth
Web Server	Tomcat	
Enterprise Platform	Cloud Foundry / k8s	Operating platform for managing the container environment.

Table 1: Technology Choice

9 Deployment

Every deliverable follows the standards of Docker and Kubernetes and can be run on any platform that implements these standards. This leaves it to the operator to freely choose their platform and integrate further services like for example logging, auditing, monitoring or application management.

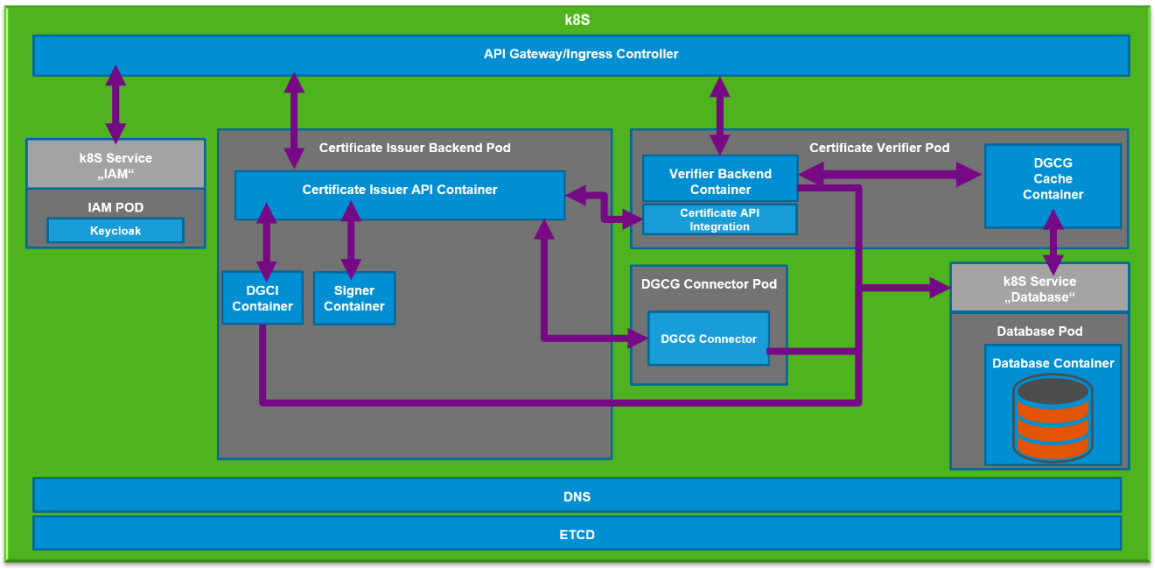

Figure 18: Deployment Example

10 Implementation Roadmap

Feature	Component	Expected Version
Issuer Input Web GUI	Issuer	1.0
Issuer Backend	Issuer	1.0
2D Generation	Issuer	1.0
Signing of DGCs	Issuer	1.0
Green Certificate Registry	Issuer	1.0
TAN Generation/Validation	Issuer	1.0
2nd Factor Publishing	Issuer	TBD
Value Set Provisioning	Issuer	TBD
FHIR Connector	Issuer	TBD
Revocation Support	Issuer	TBD
Mobile App (Android/iOS)	Verifier	1.0
Verifier Backend	Verifier	1.0
Offline 2D Code Verification	Verifier	1.0
Display of Contents	Verifier	1.0
Basic Content Verification (National Verification)	Verifier	1.0
Advanced Content Verification (EU wide Verification)	Verifier	TBD
Revocation Support	Verifier	TBD
Mobile App (Android/iOS)	Wallet	1.0
Secure Green Certificates Import	Wallet	1.0
Location Validity Checkup (Rule based)	Wallet	TBD

Table 2: Roadmap

11 Glossary

Term	Description
2D code / QR code	Two-dimensional barcode
Certificate	Technical Certificate like X509 or a Digital Green Certificate, depending on the context.
Civil identity	Defined by the eHN Minimal Data Set: Person Name (The legal name of the vaccinated person (surname(s) and forename(s) in that order), Person Date of birth
Crypto Material / Cryptographic material	All material, including documents, devices, or equipment that contains cryptographic information and is essential to the encryption, decryption, or authentication of telecommunications
DGC	Digital Green Certificate. Includes also the machine processable part of the according paper document. Currently, it can be differentiated between three different types: vaccination, test, recovery based a on positive test
DGC Gateway / EU Digital Green Certificate Gateway	Exchange of public keys, certificates, and other crypto material between national backends
DGCI	Digital Green Certificate Identifier. Universally unique ID assigned to the DGC when issued by the issuer app
DGC issuance	The act of creating a Digital Green Certificate
Holder / DGC Owner	Person in possession of a Digital Green Certificate
Holder verification / verification	The process of verification to answer the question, if a person who states to be the legal holder of a Digital Green Certificate is the same person who holds the certificate
ICAO	International Civil Aviation Organization is a specialized agency of the United Nations
Issuer	A person or system that works in behalf of an Issuing (health) authority to issue Digital Green Certificates
Issuer App / Issuer application frontend	The application that is used by the issuer to issue Digital Green Certificates. The issuer application frontend provides a user interface that is used by the issuer to enter the necessary data. The application will communicate with the backend / dgca-issuer-service to implement the process of digital green certificates signing. *Component: dgca-issuer-web*
Issuing authority	Institutions named by a member state to act in its will for issuing digital green certificates
JSON	JavaScript Object Notation is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value)
Key pair	Public and private key in the context of asymmetric cryptography

Member states / MS	Member state of the European Union (currently 27) or a third country in scope of the system (e.g. based on an adequacy decision)
National certificate backend / backend	The issuer backend is accessed by the issuer frontend application and the respective wallet apps. The backend itself publishes its public keys to the DGCG where they can be distributed to other MS. Each MS hosts its own issuer backend. DGC signing, TAN generation, TAN validation, public key publication, and digital green certificate validation/revocation inf, etc.
NTP server	The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks.
Offline verification	Process of verification without the need of an active internet connection during the time of verification. Attention: offline verification will make use of an internet connection that downloads necessary crypto material in advance.
Onboarding	The structured process of including and gaining interoperability on a technical and organizational level of a member state to issue and verify digital green certificates
Second factor / 2FA	Authentication method in which a computer user is granted access only after successfully presenting two pieces of evidence (or factors) to an authentication mechanism. E.g., a transaction authentication number.
Service X	Sample service, e.g., in a booking process
SMS	Short Message Service. Text messaging service component
TAN	Transaction authentication number
Trust Anchor	An authoritative entity for which trust is assumed and not derived
Trust Lists / CTL	A predefined list of items signed by a trusted entity
Verifier	Person that verifies DGCs
Verifier App	verifies digital green certificates with the help of the dgca-verifier-service. Consists of a frontend (to be used by the verifier) and a backend (providing trusted key for a member state). *Component: dgca-verifier-app*
Wallet App	Application that holds a digital green certificate and provides a frontend to be used by the holder. *Component: dgca-wallet-app*