Documentación Técnica LACPASS

Introducción

Este documento se presenta como documentación técnica para la implementación de LACPASS, en el cual se detalla cómo opera la solución y los pasos necesarios que deben hacer los países participantes para integrarse a LACPASS.

El Pase de Vacunación de Latinoamérica y el Caribe, LACPASS, es una aplicación de intercambio de información sobre estados de vacunación de los países de América Latina y el Caribe, que permite que personas que hubiesen recibido parte o la totalidad del esquema de vacunación COVID en su país de residencia, al momento de viajar a otro país de la región puedan validar de forma simple y verificable su estado de vacunación en el país de destino, sin necesidad de realizar trámites adicionales como la homologación del certificado local de vacunación.

El proyecto LACPASS es una iniciativa de Red Americana de Cooperación en Salud Electrónica de América Latina y el Caribe (RACSEL), patrocinada por el Banco Interamericano de Desarrollo (BID) y ejecutada por el Centro Nacional de Sistemas de Información de Chile (CENS) por medio de la empresa privada Create de Chile, la cual se adjudicó la licitación para el desarrollo y puesta en marcha de este bien público.

La tecnología detrás de LACPASS se basa en el Digital Green Certificates de la Unión Europea (EU-DGC), este repositorio es un proyecto de código abierto usado en todos los países de la Unión Europea y 24 países fuera de ella. Este pase es multilenguaje y está disponible en Inglés, Español, Francés y Portugues los cuales son de especial interés en esta región. Además permite ser digital y en papel, y posee un código QR verificable a través de las aplicaciones que provee el DGC. Al conectar a los países interesados a LACPASS es posible usar la misma tecnología para conectarse al Digital Green Certificates de la Unión Europea.

El principal objetivo del proyecto LACPASS es conectar de forma segura y verificable la información sobre vacunación individual de los residentes de los países de la región en un sistema uniforme e interoperable que facilite los viajes dentro de la región entregando a las autoridades sanitarias y migratorias de los países una herramienta que le entregue información veráz y oportuna sobre el estado de vacunación de los pasajeros que se encuentran entrando o transitando.

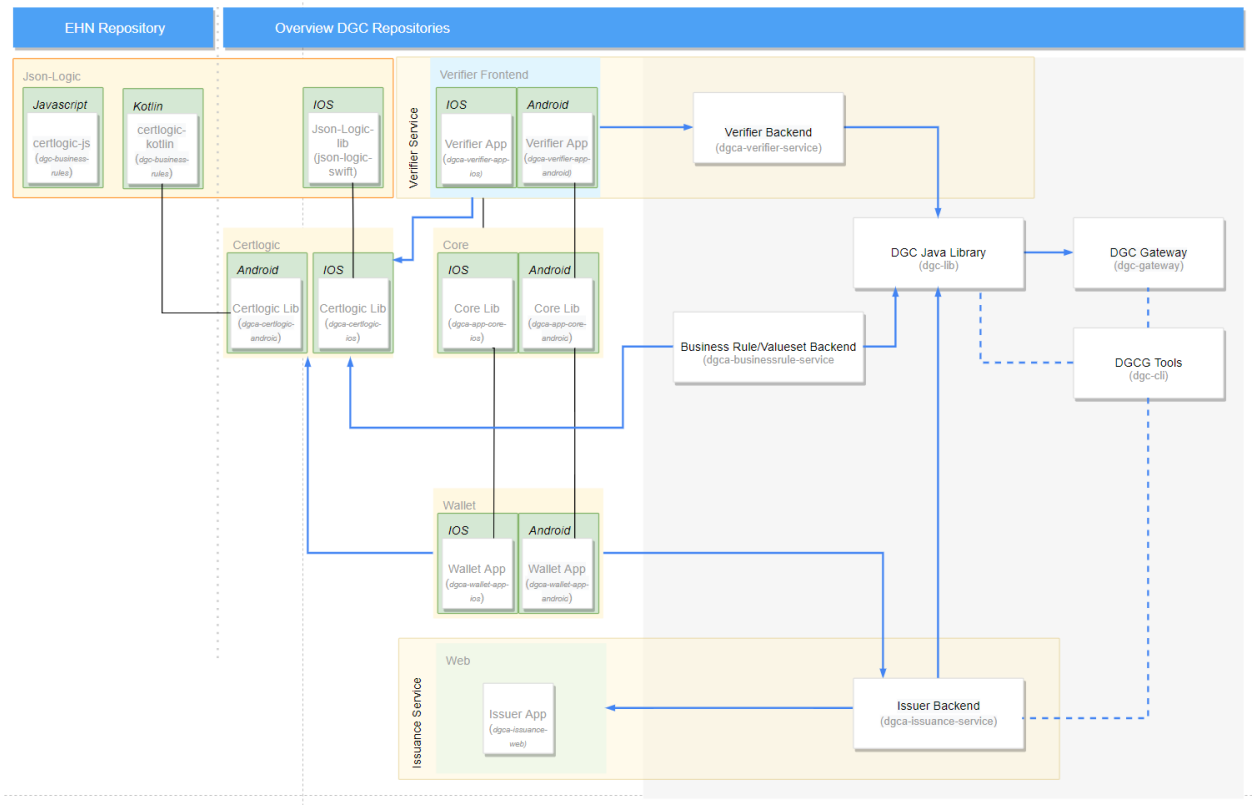
Como objetivo adicional, se busca colaborar con los países de la región para que puedan conectarse de forma simple y fluida a la tecnología del Digital Green Certificates de la Unión Europea. .

Arquitectura

Como se explicó anteriormente la implementación de LACPASS se basa en los proyectos de Digital Green Certificates de la Unión Europea (DGC) y European Health Network (EHN), cuyos repositorios se encuentran en los siguientes enlaces:

- DGC: https://github.com/eu-digital-green-certificates
- EHN: https://github.com/ehn-dcc-development

La DGC provee distintos repositorios para la implementación de interoperabilidad de certificados de vacunación. La interacción de todos estos repositorios está mostrada en el siguiente diagrama.



Funcionalmente los repositorios se pueden dividir en 3 grupos:

Lógica y Sincronización

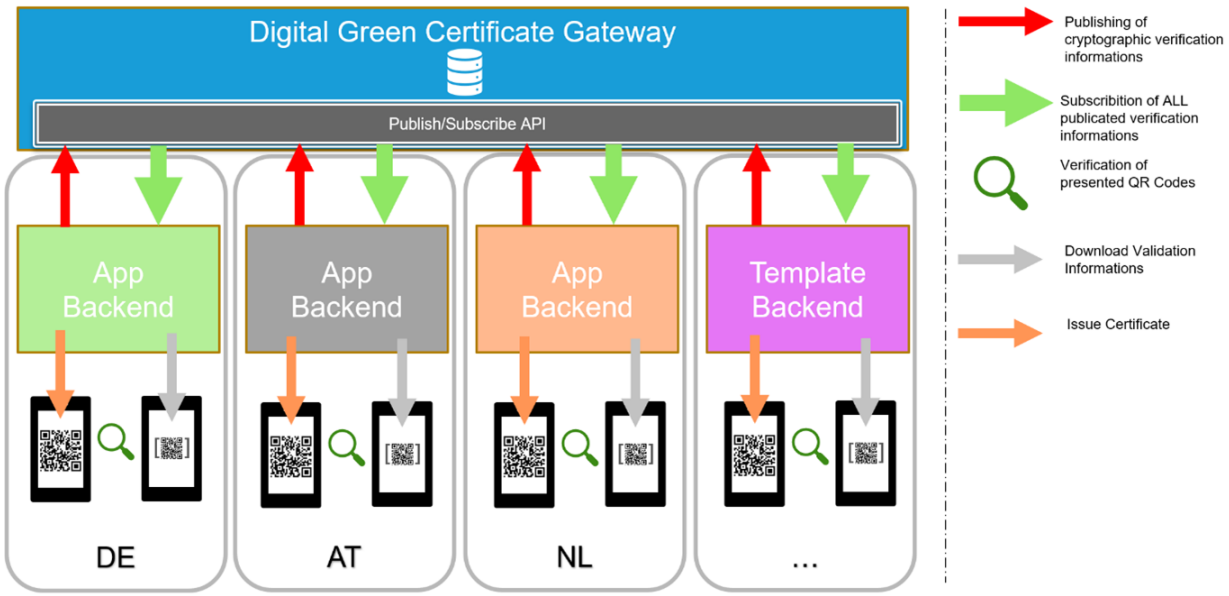
Gateway

El DGC Gateway tiene el propósito de servir de soporte para todo el sistema DGC, provee todos los servicios necesarios para el traspaso seguro de validaciones y verificaciones entre sistemas nacionales. Cada sistema nacional puede implementar su propio DGC Gateway para

obtener la libertad de distribuir las llaves con la tecnología preferida y además para poder manejar sistemas de verificación nacionales.

Adicionalmente si el certificado es generado en un formato estándar correcto, cualquier dispositivo verificador podrá verificar códigos de cualquier país que tenga el formato EU. Esto funciona tanto para el verificador conectado al sistema nacional como los sistemas offline que tengan descargadas las llaves públicas necesarias de antemano.

En el siguiente diagrama se muestra el flujo entre los distintos sistemas nacionales y el DGC Gateway:



Aquí hay un enlace con documentación detallada del DGC Gateway (Documentación)

Business Rule Service

El DGC Business Rule Service es uno de los servicios conectados al DGC Gateway, este servicio provee de las reglas necesarias para poder verificar si un código es o no válido en un sistema nacional. Estas reglas están basadas en las vacunas que posee, los test realizados y el estado de recuperación de la persona validada.

Para generar estas reglas de validación existe un formato más detallado en este enlace (Business Rules Test Data).

Emisión

Issuance Service

El DGC Issuance Service es el sistema backend que provee los servicios tanto de creación como de firma de nuevos certificados (green certificates). Cada país debe levantar este servicio para poder tener los certificados. Para que los certificados puedan ser usados internacionalmente se deben compartir las llaves públicas en el Gateway para que todos los países puedan verificar los certificados. Este servicio es usado por las aplicaciones móviles (Android, iOS) y por la aplicación web.

Issuance Web

El Issuance Web es una aplicación web que provee una interfaz de usuario usada para proveer los datos necesarios en el issuance service. También se pueden generar certificados en esta aplicación.

Verificación

Verifier Service

Para verificar los certificados es necesario tener las llaves públicas del sistema nacional adecuado. El DGC Verifier Service es un servicio backend que se utiliza para gestionar las llaves públicas obtenidas a través del DGCG. Este servicio se utiliza en las aplicaciones móviles para obtener las llaves públicas y verificar los green certificates.

Para verificar los certificados se puede usar tanto el verificador en **iOS** como en **Android**. Ambos repositorios contienen una aplicación muy simple para escanear los códigos QR y una interfaz de verificación y validación de estos.

Seguridad y Llaves de Encriptación

El sistema del DGC usa un sistema de seguridad basado en el paradigma de llaves públicas y privadas que se usan para verificar la autenticidad de las consultas y la firma de los certificados. Algo importante de notar es que estas llaves públicas son verificadas directamente por la aplicación del DGC y no necesariamente siguen las reglas usuales de HTTPS.

Dentro de los repositorios se usan distintos formatos y estándares para el guardado de las llaves, a continuación se describe cada uno de estos formatos:

- **PEM:** Archivo que contiene una llave pública y opcionalmente una llave privada de forma plana. Generalmente sólo se incluye la llave pública.
- **KEY:** Archivo que contiene una llave privada de forma plana. Este archivo **nunca debería ser compartido** con terceros para evitar ataques y vulnerabilidades.

- **P12:** Archivo que contiene una llave pública y opcionalmente una privada de forma encriptada por una contraseña. Normalmente se toma como entrada un archivo PEM para construir un P12.
- **JKS:** Formato similar al P12 que es capaz de ser leído por aplicaciones Java de forma simple.

Implementación

Esta sección describe los pasos que son necesarios realizar para que un nuevo país participante se incluya en LACPASS.

Tecnologías

Los repositorios del "*EU Digital Covid Certificates*" (EUDCC) proveen APIs que están desarrollados en Spring Framework usando Java como lenguaje de programación primario. Las bases de datos utilizadas son Mysql y Postgresql. La aplicación web de emisión de certificados está desarrollada en React. Y las aplicaciones móviles están desarrolladas de forma nativa en Kotlin (Android) y Swift (iOS). Todos los proyectos a excepción de las aplicaciones móviles están disponibles a través de Docker.

Requisitos del Servidor

Se requiere de un servidor el cual alojará los repositorios de los servicios web. Las características de este servidor dependerán del tráfico estimado, pero se recomienda un servidor con al menos 4 vCPU, 8 Gb de RAM y 50 Gb de disco.

El servidor también alojará las llaves de encriptación necesarias para el funcionamiento de las aplicaciones, por esto se recomienda que también cuente con un Hardware security module (HSM) para el manejo de estas llaves.

Pre-requisitos

A continuación se darán los pasos a seguir para levantar cada uno de los repositorios del EUDCC. Pre-requisitos:

- OpenJDK 11
- Maven
- Autenticarse con Github Packages
- Docker (opcional)
- Docker-compsoe (opcional)
- Node 14
- OpenSSL
- DGC-CLI

Para poder instalar las dependencias a través de Maven en los repositorios que utilizan Spring como tecnología, se necesita estar autenticado por Github. Para esto se necesita crear un token de acceso personal, que tenga la opción "*read:packages*" seleccionada. Luego se debe rellenar el archivo de configuración de maven (en linux ubicado en ~/.m2/settings.xml) como el que se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
https://maven.apache.org/xsd/settings-1.0.0.xsd">
  <interactiveMode>false</interactiveMode>
  <servers>
    <server>
      <id>dgc-github</id>
      <username>$USER</username>
      <password>$TOKEN</password>
    </server>
    <server>
      <id>ehd-github</id>
      <username>$USER</username>
      <password>$TOKEN</password>
    </server>
  </servers>
</settings>
```

Gateway

El gateway es utilizado para compartir y verificar información a través de todos los países conectados a él. Por lo que no deberá ser incluido en el backend de cada país, acá se explica cómo levantar un gateway solo con el fin de poder probar la conexión de otros servicios. El repositorio puede ser clonado utilizando:

```
$ git clone https://github.com/eu-digital-green-certificates/dgc-gateway
```

Llaves

Para correrlo de manera local se necesita crear un *TrustAnchor.* El TrustAnchor es usado para firmar entradas en la base de datos. Para crear el TrustAnchor se usa el siguiente comando:

```
$ openssl req -x509 -newkey rsa:4096 -keyout key_ta.pem -out cert_ta.pem
-days 365 -nodes
```

Luego la llave pública se exporta al Keystore de Java utilizando:

```
$ keytool -importcert -alias dgcg_trust_anchor -file cert_ta.pem -keystore
ta.jks -storepass dgcg-p4ssw0rd
```

Donde “cert_ta.pem” es la llave pública y “”dcg-p4ssw0rd” es la clave de la llave. Esta llave “ta.jks” debe ser colocada en una carpeta nombrada “certs”, la cual debe ser creada en la raíz del repositorio.

Base de datos

Este repositorio utiliza una base de datos MySql, si es que no se utiliza docker para construir el proyecto, se necesita instalar y crear una base en MySql.

Configuración

Para configurar variables como por ejemplo el directorio de la llave pública y la conexión a la base de datos, se puede hacer de dos maneras. Si es que se utiliza Docker para ejecutar el proyecto se pueden editar las variables de entorno que se muestran en “docker-compose.yml”, para más detalles sobre este archivo la documentación está disponible en el siguiente link . Si es que no se utiliza docker se puede editar el archivo de configuración de Spring en “~/dgc-gateway/src/main/resources/application.yml”

Ejecutar

Para construir el ejecutable del proyecto, el cual es construido a través de Maven, se utiliza el siguiente comando:

```
$ mvn clean install
```

Si es que se utilizara docker para correr el proyecto se le debe agregar una bandera extra al comando anterior:

```
$ mvn clean install -P docker
```

Esto creará un archivo “jar” en el directorio “~/dgc-gateway/target”. Para correr la aplicación se utiliza:

```
$ java -jar target/dgc-gateway-latest.jar
```

Y si se utiliza Docker, se puede utilizar:

```
$ docker-compose up --build
```

Lo cual subirá la API del gateway junto con una base de datos mysql. Para poder hacer consultas a la API de este gateway, es necesario registrar ciertos certificados que pertenecen al backend de cada país. Estos certificados serán de AUTHENTICATION, UPLOAD y CSCA. Para esto se pueden crear estos certificados con OpenSSL:

```
# AUTHENTICATION
$ openssl req -x509 -newkey rsa:4096 -keyout key_auth.pem -out
cert_auth.pem -days 365 -nodes

# CSCA
$ openssl req -x509 -newkey rsa:4096 -keyout key_csca.pem -out
cert_csca.pem -days 365 -nodes

# UPLOAD
$ openssl req -x509 -newkey rsa:4096 -keyout key_upload.pem -out
cert_upload.pem -days 365 -nodes
```

Estos certificados deben ser firmados por el *TrustAnchor* del gateway (“cert_ta.pem” y “key_ta.pem”), para esto se puede usar el cliente facilitado por el EUDCC. Este se puede bajar en este link. Luego usando este jar, se pueden ejecutar los siguientes comandos:

```
$ java -jar dgc-cli.jar ta sign -c cert_ta.pem -k key_ta.pem -i
cert_auth.pem
$ java -jar dgc-cli.jar ta sign -c cert_ta.pem -k key_ta.pem -i
cert_csca.pem
$ java -jar dgc-cli.jar ta sign -c cert_ta.pem -k key_ta.pem -i
cert_upload.pem
```

En cada uno de estos comandos se entregara un “TrustAnchor Signature”, “Certificate Raw Data”, “Certificate Thumbprint” y “Certificate Country”. Estos valores tienen que ser ingresados en la tabla “trusted_party” de la base de datos del gateway, por lo que se agregaran tres nuevas líneas en esta tabla (por cada uno de los certificados). Esto puede hacerse usando:

```
$ mysql --user=root --password=admin dgc
$ INSERT INTO trusted_party (created_at, country, thumbprint, raw_data, signature,
certificate_type)
SELECT
       NOW() as created_at,
       'CL' as country,
       '{Certificate_Thumbprint}' as thumbprint,
       '{Certificate_Raw_Data}' as raw_data,
```

```
      '{TrustAnchor_Signature}' as signature,
      '{AUTHENTICATION|UPLOAD|CSCA}' as certificate_type;
```

Para probar que los valores fueron ingresados correctamente, se le puede hacer una petición a la API del gateway utilizando el thumbprint de autenticación:

```
$ curl -X GET http://localhost:8080/trustList -H "accept: application/json"
-H "X-SSL-Client-SHA256: $THUMBPRINT" -H "X-SSL-Client-DN: C=$COUNTRY"
```

Lo cual deberá entregar la lista de certificados en la tabla “trusted_parties”.

Business rule

Este repositorio contiene un backend con las reglas de negocio para aceptar/rechazar los estados de los certificados COVID emitidos por los países. El repositorio puede ser clonado utilizando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-businessrule-service
```

LLaves

Este repositorio necesita tres llaves, un trust_anchor, trust_store y key_store. El trust_anchor es el TrustAnchor creado en el gateway, el trust_store puede ser creado utilizando el certificado y llave de autenticación que fueron registradas en el gateway de la siguiente manera :

```
$ openssl pkcs12 -export -in cert_auth.pem -inkey key_auth.pem -name 1 -out
tls_key_store.p12
```

El truststore es creado utilizando el certificado de autenticacion, con el comando:

```
$ openssl pkcs12 -export -in cert_auth.pem -name tls_trust -out
tls_trust_store.p12 -nokeys
```

Base de datos

Este repositorio utiliza una base de datos Postgresql, si es que no se utiliza docker para construir el proyecto, se necesita instalar y crear una base en Postgresql.

Configuración

Para configurar variables como por ejemplo el directorio de las llaves y la conexión a la base de datos, se puede hacer de dos maneras. Si es que se utiliza Docker para correr el proyecto se pueden editar las variables de entorno que se muestran en “docker-compose.yml”, para más detalles sobre este archivo la documentación está disponible en el siguiente link . Si es que no se utiliza docker se puede editar el archivo de configuración de Spring en “~/dgc-gateway/src/main/resources/application.yml”. Las variables mas importantes se muestran a continuación:

```
# Credenciales base de datos
SPRING_DATASOURCE_URL=<CONNECTION_URL>
SPRING_DATASOURCE_USERNAME=<USER>
SPRING_DATASOURCE_PASSWORD=<PASSWORD>

# Gateway endpoint
DGC_GATEWAY_CONNECTOR_ENDPOINT=https://test-dgcg-ws.tech.ec.europa.eu

# Certificados
DGC_GATEWAY_CONNECTOR_TLSTRUSTSTORE_PATH=<PATH>
DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_ALIAS=<ALIAS>
DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_PATH=<PATH>
DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_PASSWORD=<PASSWORD>
DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_ALIAS=<ALIAS>
DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_PATH=<PATH>
DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_PASSWORD=<PASSWORD>
```

Ejecutar

Para construir el ejecutable del proyecto, el cual es construido a través de Maven, se utiliza el siguiente comando:

```
$ mvn clean install
```

Esto creará un archivo “jar” en el directorio “~/dgc-businessrule-service/target”. Para correr la aplicación se utiliza:

```
$ java -jar target/dgc-businessrule-service-latest.jar
```

Y si se utiliza Docker, se puede utilizar:

```
$ docker-compose up --build
```

Reglas

En esta sección se explica brevemente cómo generar un archivo JSON con las reglas de validación de los certificados. Estas reglas determinan si una persona que entra a un país es considerada apta para entrar a este país, las reglas se basan en las vacunas administradas, los test que ha realizado y el estado de recuperación después de contraer COVID. Todas estas reglas deben estar codificadas según los estándares del Digital COVID Certificate.

Para generar las reglas de validación se requiere generar un json con lo siguiente:

- Cumplir la semántica CertLogic
- Tener el esquema estándar (Schema)
- Los campos especificados en “AffectedFields” deben estar contenidos en el esquema DCC. (DCC Schema)

CertLogic es una semántica que extiende la semántica JsonLogic. Estas semánticas usan reglas intuitivas y simples para poder verificar patrones o lógicas dentro de un archivo Json. Usan operadores lógicos como la igualdad (“==”), operadores numéricos, etc. Estos operadores puedes encontrarlos aquí.

Aquí se muestra un ejemplo de cómo se construye un Json con la semántica CertLogic:

```
{
  "<operation id>": [
     <operand 1>,
     <operand 2>,
     // ...
     <operand n>
  ]
}
```

Ahora para generar un archivo con los estándares correctos se debe seguir el esquema correctamente, para esto se deben agregar los siguientes campos:

- **AffectedFields:** Arreglo de reglas que se usarán del payload (QR).
- **Country:** Código ISO del país. (Ej: “CL”).
- **CertificateType:** Tipo del certificado. Los valores válidos son “General”, “Test”, “Vaccination”, “Recovery”. Si por ejemplo la regla busca el tiempo mínimo después de un test de COVID este certificado es del tipo “Recovery”.
- **Description:** Arreglo con la descripción de la regla, aquí se agregan todos los idiomas que se quiera soportar.
- **Engine:** Tipo de semántica usada. (Ej: “CERTLOGIC”)

- **EngineVersion:** Versión de la semántica. Actualmente es la “1.2.2”.
- **Identifier:** Identificador único de la regla. Debe ser el patron "^(GR|VR|TR|RR|IR)-[A-Z]{2}-\\d{4}$". Por ejemplo si la regla es de “Recovery”, el país es Chile y además es la primera regla, el identificador es “RR-CL-0000”.
- **Logic:** Objeto donde se establece la regla. Aquí se usa la semántica para definir la regla.
- **SchemaVersion:** Versión del esquema usado.
- **Type:** Tipo de la regla, puede ser de aceptación (“Acceptance”) o invalidación (“Invalidation”).
- **ValidFrom:** Hasta que fecha esta regla es válida (sin ms y con zona horaria).
- **ValidTo:** Desde que fecha esta regla es válida (sin ms y con zona horaria).
- **Version:** Versión de la regla.

Para entender mejor cómo se genera este archivo se explicara de forma general cómo construir los campos “Logic” y “AffectedFields”. Para el campo “AffectedFields” se debe entender como llega el payload (contenido del QR), el contenido tiene un formato estándar que se puede encontrar en este enlace. El objeto payload debe contener al menos uno de los siguientes campos:

- “v”: Contiene todo lo relacionado con la vacunación (“Vaccination Entry”).
- “t”: Contiene todo lo relacionado con los tests realizados (“Test Entry”).
- “r”: Contiene todo lo relacionado con la recuperación (“Recovery Entry”).

Cada uno de estos campos puede contener atributos específicos a lo que representa, detallaremos en los siguientes puntos que puede contener cada uno.

Vaccination Entry (“v”)

- tg: Enfermedad o agente objetivo.
- vp: Vacuna o profilaxis.
- mp: Medicamento de la vacuna.
- ma: Empresa de marketing autorizada o fabricante.
- dn: Número de la Dosis.
- sd: Total de dosis (Serie de dosis, por ejemplo serían 2 si se requieren dos dosis).
- dt: Fecha de vacunación.
- co: País de vacunación.
- is: Emisor del certificado.
- ci: Identificador único del certificado (UVCI).

Test Entry (“t”)

- tg: Enfermedad o agente objetivo.
- tt: Tipo de test.
- nm: Prueba de ácido nucleico.
- ma: Nombre de la prueba rápida de antígenos y fabricante.
- sc: Fecha/Hora de recolección de la muestra.
- tr: Resultado del examen.
- tc: Centro encargado del examen.
- co: País del examen.
- is: Emisor del certificado.
- ci: Identificador único del certificado (UVCI).

Recovery Entry (“r”)

- tg: Enfermedad o agente objetivo.
- fr: Fecha primer positivo del test de ácido nucleico.
- co: País del examen.
- is: Emisor del certificado.
- df: Fecha desde que el examen es válido.
- du: Fecha hasta cuando es válido el examen.
- ci: Identificador único del certificado (UVCI).

Existe un documento oficial sobre la documentación de este estándar, en este enlace.

Para entender mejor cómo se eligen los valores, tomaremos como ejemplo la regla “La serie de vacunación debe estar completa (por ejemplo, 1/1, 2/2)”. Para este ejemplo los valores de “AffectedFields” serían los siguientes:

```
"AffectedFields": [
  "v.0", // Se necesitan los valores de vacunación.
  "v.0.dn", // Dosis actual.
  "v.0.sd" // Número de dosis total de la serie.
]
```

Ya entendiendo cómo se arma el “AffectedFields” siguiendo con el mismo ejemplo vamos a construir la lógica de la regla: “La pauta de vacunación debe estar completa (por ejemplo, 1/1, 2/2)”. Lo primero que hay que notar es que es una regla de vacunación entonces se usa la parte “v” del payload. Además como se busca comprobar las series de vacunación se usará tanto “dn” y “sd” donde sacaremos la información de la dosis actual y el total de dosis requeridos respectivamente. Entonces para validar el esquema de vacunación completo se debe verificar que ambos valores sean los mismos, como se muestra en en el siguiente esquema:

```
"Logic": {
  "if": [ //  Si es que se cumple lo contenido, se acepta la regla.
    {
      "var": "payload.v.0" // Se explicita de donde obtener los valores.
    },
    {
      "===": [ // Se usa el operador de igualdad exacta.
        {
          "var": "payload.v.0.dn" // Actual Dosis (Número en la serie).
        },
        {
          "var": "payload.v.0.sd" // Número de dosis total de la serie.
        }
    }
  ]
}
```

Este ejemplo fue extraído de las reglas de España aquí. Si necesitas más ejemplos puedes ver los recomendados por la EU (Más ejemplos).

Issuance

Este repositorio contiene un backend con que permite la emisión de certificados de vacunación. El repositorio puede ser clonado utilizando el siguiente comando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-issuance-service
```

LLaves

Este proyecto no requiere crear nuevas llaves, se utilizarán las mismas creadas anteriormente para el business rule. De hecho en vez de copiar las llaves entre los repositorios se recomienda tener un directorio en donde todos los repositorios compartan las llaves y se compartan a través de link simbólicos o volúmenes de docker.

Base de datos

Este repositorio utiliza una base de datos Postgresql, si es que no se utiliza docker para construir el proyecto, se necesita instalar y crear una base en Postgresql.

Configuracion

Al igual que el business rule, la configuración del repositorio se encuentra en el archivo docker-compose.yml, pero también se puede cambiar directamente el archivo "src/main/resources/application.yml" en caso de no utilizar docker.

El servicio de issuance tiene dos formas de ejecución: una de testing y otra conectada a un gateway. Ambas se explican a continuación:

Configuración de Testing: Esta es la que viene por defecto y sirve para probar rápidamente la emisión de certificados sin tener que instalar un gateway. No se requiere mayor configuración para operar en este modo y se utiliza una llave de prueba genérica para la firma de los certificados.

Configuración Producción: Esta configuración se conecta a un gateway y permite interoperar con los demás servicios del DGC. Para acceder a esta configuración hay que cambiar el docker-compose.yml y agregar las siguientes configuraciones al backend

```
backend:
   environment:
      ... # MANTENER LO QUE ESTA Y AGREGAR LO SIGUIENTE
      # EMISION DE CERTIFICADOS
      - ISSUANCE_DGCIPREFIX=URN:UVCI:V1:CL
      - ISSUANCE_KEYSTOREFILE=/app/certs/CL/firmasalud.jks
      - ISSUANCE_KEYSTOREPASSWORD=dgcg-p4ssw0rd
      - ISSUANCE_CERTALIAS=firmador
      - ISSUANCE_PRIVATEKEYPASSWORD=dgcg-p4ssw0rd
      - ISSUANCE_COUNTRYCODE=CL
      - ISSUANCE_EXPIRATION_VACCINATION=365
      - ISSUANCE_EXPIRATION_RECOVERY=365
      - ISSUANCE_EXPIRATION_TEST=60
      # SERVICIOS DISPONIBLES
      - ISSUANCE_ENDPOINTS_FRONTENDISSUING=true
      - ISSUANCE_ENDPOINTS_BACKENDISSUING=true
      - ISSUANCE_ENDPOINTS_TESTTOOLS=true
      - ISSUANCE_ENDPOINTS_WALLET=true
      - ISSUANCE_ENDPOINTS_PUBLISHCERT=true
      - ISSUANCE_ENDPOINTS_DID=true
      # CONFIGURACION DE GATEWAY
      - DGC_GATEWAY_CONNECTOR_ENABLED=true
      - DGC_GATEWAY_CONNECTOR_ENDPOINT=https://lacpass.example.com:3050
      - DGC_GATEWAY_CONNECTOR_PROXY_ENABLED=false
      - DGC_GATEWAY_CONNECTOR_PROXY_HOST=
      - DGC_GATEWAY_CONNECTOR_PROXY_PORT=-1
      - DGC_GATEWAY_CONNECTOR_MAX-CACHE-AGE=300
      -
DGC_GATEWAY_CONNECTOR_TLSTRUSTSTORE_PATH=file:/app/certs/tls_trust_store
.p12
```

```
      - DGC_GATEWAY_CONNECTOR_TLSTRUSTSTORE_PASSWORD=dgcg-p4ssw0rd
      -
DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_PATH=file:/app/certs/tls_key_store.p12
      - DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_PASSWORD=dgcg-p4ssw0rd
      - DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_ALIAS=tls_key
      - DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_PATH=file:/app/certs/ta.jks
      - DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_PASSWORD=dgcg-p4ssw0rd
      - DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_ALIAS=trustanchor
      -
DGC_GATEWAY_CONNECTOR_UPLOADKEYSTORE_PATH=file:/app/certs/CL/upload_key_
store.p12
      - DGC_GATEWAY_CONNECTOR_UPLOADKEYSTORE_ALIAS=upload_key
      - DGC_GATEWAY_CONNECTOR_UPLOADKEYSTORE_PASSWORD=dgcg-p4ssw0rd
```

Asegurarse de reemplazar correctamente las llaves. Más información de esta configuración en [este link](#).

Ejecutar

Para construir el ejecutable del proyecto, el cual es construido a través de Maven, se utiliza el siguiente comando:

```
$ mvn clean package
```

Esto creará un archivo “jar” en el directorio “~/dgc-issuance-service/target”. Para correr la aplicación se utiliza:

```
$ java -jar target/dgc-issuance-service-latest.jar
```

Y si se utiliza Docker, se puede utilizar:

```
$ docker-compose up --build
```

Al finalizar, en el puerto indicado en la configuración se debería levantar el servicio web que emite certificados. Por ejemplo si se usa el puerto 8081 se puede navegar a esta URL: http://localhost:8081/swagger

Cliente Web

Para probar la emisión de certificados, la DGC provee otro repositorio llamado issuance-web. Este corresponde a una aplicación web que consume la API entregada por el issuance-service y permite generar certificados de vacunación. Esta aplicación funciona independiente si se

eligió el modo de Testing o modo productivo en el issuance-service. Para clonar el repositorio se puede ejecutar el siguiente comando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-issuance-web
```

Luego para conectar con las APIs es necesario modificar el archivo docker-compose.yml, o bien cambiar el archivo de configuración de nginx.

```
    - DGCA_ISSUANCE_SERVICE_URL=http://dgc-issuance-service:8081
    - DGCA_BUSINESSRULE_SERVICE_URL=http://dgc-businessrule-service:8082
```

Aquí se debe especificar las URLs del issuance-service y business rule. Algo importante a notar es que este repositorio trae como dependencias estos dos servicios, dado que en esta guía estamos levantando y configurando cada servicio por separado es necesario eliminar estos de la configuración.

Finalmente se puede ejecutar la aplicación web usando el siguiente comando

```
$ docker-compose up --build
```

Verifier

Este repositorio contiene un backend con que permite la verificación de certificados emitidos de vacunación. El repositorio puede ser clonado utilizando el siguiente comando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-verifier-service
```

LLaves
Al igual que el issuance service este repositorio necesita las llaves previamente creadas.

Base de datos
Este repositorio utiliza una base de datos Postgresql, si es que no se utiliza docker para construir el proyecto, se necesita instalar y crear una base en Postgresql.

Configuración
Al igual que el issuance-service, la configuración del repositorio se encuentra en el archivo docker-compose.yml, pero también se puede cambiar directamente el archivo "src/main/resources/application.yml" en caso de no utilizar docker.

Para el verifier-service no existe un modo de testing, por lo que siempre se utiliza con un gateway asociado. Para configurarlo es necesario modificar el archivo de configuración e indicar las rutas al gateway y a las llaves:

```
      - DGC_GATEWAY_CONNECTOR_ENDPOINT=https://dgc-gateway.example.com
      -
DGC_GATEWAY_CONNECTOR_TLSTRUSTSTORE_PATH=file:/ec/prod/app/san/dgc/tls_trus
t_store.p12
      - DGC_GATEWAY_CONNECTOR_TLSTRUSTSTORE_PASSWORD=dgcg-p4ssw0rd
      - DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_ALIAS=1
      -
DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_PATH=file:/ec/prod/app/san/dgc/tls_key_st
ore.p12
      - DGC_GATEWAY_CONNECTOR_TLSKEYSTORE_PASSWORD=dgcg-p4ssw0rd
      - DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_ALIAS=ta
      -
DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_PATH=file:/ec/prod/app/san/dgc/trust_anch
or.jks
      - DGC_GATEWAY_CONNECTOR_TRUSTANCHOR_PASSWORD=dgcg-p4ssw0rd
```

Ejecutar

Al igual que los repositorios anteriores, para construir el ejecutable del proyecto, se utiliza el siguiente comando de Maven:

```
$ mvn clean install
```

Esto creará un archivo “jar” en el directorio “~/dgc-verifier-service/target”. Para correr la aplicación se utiliza:

```
$ java -jar target/dgc-issuance-verifier-latest.jar
```

Y si se utiliza Docker, se puede utilizar:

```
$ docker-compose up --build
```

Al finalizar, en el puerto indicado en la configuración se debería levantar el servicio web que emite certificados. Por ejemplo si se usa el puerto 8082 se puede navegar a esta URL: http://localhost:8082/swagger

Aplicaciones Móviles de Verificación

Para las aplicaciones móviles los repositorios se dividen en las plataformas iOS y Android. Ambas plataformas tienen 4 repositorios divididos por funcionalidades que cumplen cada uno. Para los desarrollos de levantamiento de aplicaciones para verificar los certificados solo se modificarán los repositorios “verifier” y “wallet” según lo que se necesite. Los repositorios de ambas plataformas son las siguientes:

- **App Core:** Este repositorio contiene todos los servicios necesarios para conectarse al DGC Verifier Service y con el DGC Business Rule. También se encarga de firmar los certificados para poder enviarlos de forma segura.
    - iOS: https://github.com/eu-digital-green-certificates/dgca-app-core-ios
    - Android: https://github.com/eu-digital-green-certificates/dgca-app-core-android

- **Verifier:** Este repositorio contiene la aplicación móvil que se encarga de escanear y verificar los certificados usando las llaves públicas, usa el App Core para hacer los llamados pertinentes.
    - iOS: https://github.com/eu-digital-green-certificates/dgca-verifier-app-ios
    - Android: https://github.com/eu-digital-green-certificates/dgca-verifier-app-android

- **Wallet:** Este repositorio provee una interfaz de usuario para poder administrar y guardar los DGCs personales.
    - iOS: https://github.com/eu-digital-green-certificates/dgca-wallet-app-ios
    - Android: https://github.com/eu-digital-green-certificates/dgca-wallet-app-android

- **CertLogic:** Este repositorio contiene el código fuente para poder manejar la semántica CertLogic en las aplicaciones móviles.
    - iOS: https://github.com/eu-digital-green-certificates/dgc-certlogic-ios
    - Android: https://github.com/eu-digital-green-certificates/dgc-certlogic-android

En caso de que se requiera ejemplos de códigos QR, están estos ejemplos oficiales (https://dgc.a-sit.at/ehn/testsuite).

IOS

Los requisitos para los servicios en iOS son:

- Se necesita un Mac o una máquina virtual para correr Xcode.
- Xcode 12.5+ es usada para las compilaciones. Se requiere un sistema operativo macOS 11.0+.
- Para instalarlo en dispositivos físicos, se necesita una cuenta de desarrollador de Apple. Para esto debes enrollar en el programa de desarrollo de apple (Apple Developer Program)

Verifier

Este repositorio contiene la aplicación móvil para verificar certificados a través de iOS. Para poder instalar este proyecto primero debes clonarlo localmente con el siguiente comando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-verifier-app-ios
```

Para poder tener los servicios de conexión y firma de certificados debes tener además el repositorio core en la misma carpeta, puedes usar el mismo comando usado para clonar el repositorio core.

```
<project folder>
|___dgca-app-core-ios
|___dgca-verifier-app-ios
```

Una vez que tengas instalados ambos repositorios, deben modificar el archivo context.jsonc con los valores correctos del sistema nacional. Este archivo se encuentra en la carpeta “context”. Debes rellenar los valores adecuados como se muestra en el siguiente esquema:

```
{
  // Origin in ISO alpha 2 code:
  "origin": "XX",
  "versions": {
    "default": {
      "privacyUrl": "https://<PRIVACY_URL>",
      "context": {
        "url": "https://<URL_ISSUANCE_SERVICE>/context",
        "pubKeys": [<PUBLIC_KEYS>]
      },
      "endpoints": {
        "claim": {
          "url": "https://<URL_ISSUANCE_SERVICE>/dgci/wallet/claim",
          "pubKeys": [<PUBLIC_KEYS>]
        },
        "countryList": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/countrylist",
          "pubKeys": [<PUBLIC_KEYS>]
        },
        "rules": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/rules",
          "pubKeys": [<PUBLIC_KEYS>]

        },
        "valuesets": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/valuesets",
          "pubKeys": [<PUBLIC_KEYS>]
```

```
        }
      }
    },
  }
}
```

Una vez que tienes estos valores, ya puedes correr la aplicación de validación de certificados.

Para modificar el Locale de la app solo debes generar un nuevo archivo locale dentro de la carpeta “Localization/DGCAVerifier”. Copia el archivo en.xloc y modificarlo para cumplir tu localización.

Wallet

Este repositorio contiene la aplicación móvil para guardar y gestionar certificados personales. Para poder instalar este proyecto primero debes clonarlo localmente con el siguiente comando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-wallet-app-ios
```

Para poder tener los servicios de conexión y firma de certificados debes tener además el repositorio core en la misma carpeta, puedes usar el mismo comando usado para clonar el repositorio core.

```
<project folder>
|___dgca-app-core-ios
|___dgca-wallet-app-ios
```

Al igual que el verifier debes modificar el context.jsonc para poder generar el certificado personal. También se puede modificar la localización en el mismo archivo.

Android

Los requisitos para los servicios en Android son:

- Para el desarrollo se recomienda usar Android Studio. La última versión disponible se puede descargar aquí.
- Android SDK version 26+

Verifier y Wallet (Android)

Este repositorio contiene la aplicación móvil para verificar certificados a través de Android. Para poder instalar este proyecto primero debes clonarlo localmente con el siguiente comando:

```
$ git clone
https://github.com/eu-digital-green-certificates/dgca-verifier-app-android
```

Para poder tener los servicios de conexión y firma de certificados debes tener además el repositorio core en la misma carpeta, puedes usar el mismo comando usado para clonar el repositorio core.

```
<project folder>
|___dgca-verifier-app-android
|___dgca-app-core-android
|___dgc-certlogic-android
```

Una vez que tengas instalados los repositorios, deben modificar el archivo verifier-context.jsonc con los valores correctos del sistema nacional. Este archivo se encuentra en la carpeta “app/src/acc/assets”. Debes generar un archivo llamado “config.json” en la misma carpeta y rellenar los valores adecuados como se muestra en el siguiente esquema:

Verifier

```
{
  // Origin in ISO alpha 2 code:
  "origin": "XX",
  "versions": {
    "default": {
      "privacyUrl": "https://<PRIVACY_URL>",
      "context": {
        "url": "https://<URL_VERIFIER_SERVICE>/context",
        "pubKeys": [<PUBLIC_KEYS>]
      },
      "endpoints": {
        "status": {
          "url": "https://<URL_VERIFIER_SERVICE>/signercertificateStatus",
          "pubKeys": [<PUBLIC_KEYS>]
        },
        "update": {
          "url": "https://<URL_VERIFIER_SERVICE>/signercertificateUpdate",
          "pubKeys": [<PUBLIC_KEYS>]

        },
        "countryList": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/countrylist",
          "pubKeys": [<PUBLIC_KEYS>]
        },
```

```
        "rules": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/rules",
          "pubKeys": [<PUBLIC_KEYS>]

        },
        "valuesets": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/valuesets",
          "pubKeys": [<PUBLIC_KEYS>]

        }
      }
    },
  }
}
```

Wallet

```
{
  // Origin in ISO alpha 2 code:
  "origin": "XX",
  "versions": {
    "default": {
      "privacyUrl": "https://<PRIVACY_URL>",
      "context": {
        "url": "https://<URL_ISSUANCE_SERVICE>/context",
        "pubKeys": [<PUBLIC_KEYS>]
      },
      "endpoints": {
        "claim": {
          "url": "https://<URL_ISSUANCE_SERVICE>/dgci/wallet/claim",
          "pubKeys": [
            "lKdU1EbQubxyDDm2q3N8KclZ2C94Num3xXjG0pk+3eI=",
            "r/mIkG3eEpVdm+u/ko/cwxzOMo1bk4TyHIlByibiA5E="
          ]
        },
        "countryList": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/countrylist",
          "pubKeys": [<PUBLIC_KEYS>]
        },
        "rules": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/rules",
          "pubKeys": [<PUBLIC_KEYS>]

        },
        "valuesets": {
          "url": "https://<URL_BUSINESSRULE_SERVICE>/valuesets",
          "pubKeys": [<PUBLIC_KEYS>]

        }
      }
    },
  }
```

```
}
```

En el archivo “app/src/main/java/dgca/verifier/app/android/di/NetworkModule.kt” modificar la variable “BASE_URL” por la url del verifier:

```
const val BASE_URL = "https://<URL_VERIFIER_SERVICE>/"
```

En el caso del wallet cambiarlo por:

```
const val BASE_URL = "https://<URL_ISSUANCE_SERVICE>/"
```

Para correr el proyecto en un emulador android debes ejecutar este comando:

```
$ gradlew -PCONFIG_FILE_NAME="config.json"
```

Preguntas Frecuentes

Esta sección describe las preguntas más frecuentes relacionadas al proyecto

- Si estoy usando el issuance-web, ¿cómo se puede agregar autenticación o alguna medida de restricción de acceso?

La aplicación web de issuance (issuance-web) no cuenta con un sistema de autenticación propio ni ninguna medida de control de acceso. Esto quiere decir que una vez desplegada esta aplicación cualquier usuario que tenga acceso al servidor puede usarla y emitir certificados. Existen distintas estrategias para controlar el acceso dependiendo de la complejidad y cantidad de recursos que posean los países.

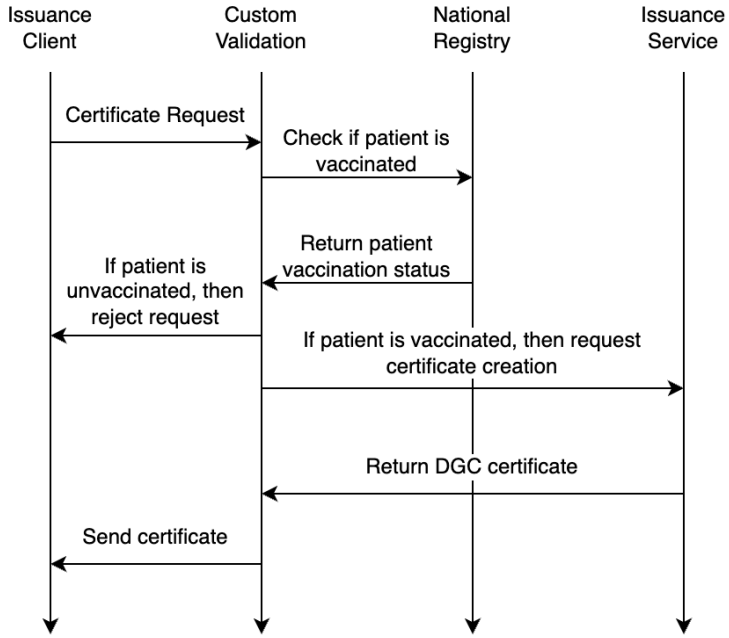
La alternativa más simple de implementar es agregar autenticación HTTP básica al servidor de proxy ya sea nginx o apache. Ambos proveen plugins para agregar este tipo de autenticación, por ejemplo configurando el archivo .htpasswd.

La siguiente opción recomendada es no utilizar el servicio de issuance-web y desarrollar una aplicación propia que utilice la API del issuance-service. De esta forma se tiene control total del desarrollo y se puede agregar autenticación a nivel de la aplicación.

De todas formas se recomienda que la aplicación esté sólo visible para un segmento específico de IPs o ubicaciones para evitar otro tipo de tráfico.

- Si tengo un registro nacional de personas vacunadas, ¿cómo puedo integrar esta información al sistema?

Los países con un Registro Local de Vacunación pueden conectar la información de las personas vacunadas a LACPASS por medio de la API del issuance-service. Lo recomendado es crear un software que se inserte entre el cliente de emisión de certificados (issuance-web o desarrollo propio) y el issuance-service. Este software debería verificar la autenticidad del certificado solicitado a emitir con el registro nacional y continuar con el proceso de emisión o detenerlo en caso que los datos solicitados no coincidan. El siguiente diagrama ejemplifica el funcionamiento.



- ¿Cuáles son los pasos a seguir para poder integrarse con la UE?

El proceso de integración está explicado en el siguiente enlace: Onboarding Checklist.

A modo general el proceso consiste en enviar a la UE las llaves públicas para agregarlas a la base de datos del gateway tal como se explicó en la sección del gateway. Y luego de hacer pruebas de que todo esté funcionando correctamente, se necesita cambiar el endpoint del gateway al endpoint oficial de ellos.

Ellos tienen a disposición 3 ambientes: Test, para las pruebas de integración (este ambiente sólo se inicia cuando se empieza el proceso de Onboarding, antes de eso está apagado). Acceptance, para probar y para que la UE valide que la integración funciona correctamente. Producción: ambiente que contiene los datos reales, una vez integrado a este ambiente se completa el proceso.

- ¿Cómo manejar las personas que se vacunan por primera vez y las personas que ya se encuentran vacunadas?

Es recomendable que los países que tienen implementado un Registro Local de Vacunación lo hayan integrado usando las instrucciones anteriores. De esta forma, la capa de validación propia que se debe desarrollar puede manejar los casos en los cuales las personas ya estuvieron vacunadas o se han vacunado por primera vez o tienen su esquema de vacunación incompleto.