Licencia de Conducir Digital

Licencia de Conducir como Credencial Verificable

¿Qué es una credencial verificable?

Una **credencial verificable** (en inglés *Verifiable Credential* o *VC*) es una representación digital de información que afirma algo sobre una persona, organización o cosa, y que puede ser **verificada criptográficamente** para confirmar su autenticidad y validez.

Las credenciales verificables están basadas en estándares definidos por el **W3C** y permiten emitir, presentar y validar datos de forma segura, privada y descentralizada.

¡Con gusto, Tomás! Aquí tienes una definición clara y técnica de una **credencial verificable**, ideal para incluir en tu documentación:

Ecosistema de una credencial verificable

Las credenciales verificables se basan en un ecosistema compuesto por entidades que desempeñan diferentes **"roles"**. Los roles principales son:

Emisor

Una entidad que crea una Credencial Verificable, compuesta por una serie de afirmaciones relacionadas con su sujeto. Un ejemplo es una universidad que emite credenciales de títulos universitarios o certificados para sus egresados.

Titular

Una entidad que posee una o más Credenciales, y que puede transmitir presentaciones de esas Credenciales Verificables a terceros. Un ejemplo puede ser la persona que "posee" sus propios títulos educativos. Otro ejemplo puede ser una billetera digital que contiene varias credenciales en nombre de alguien.

Verificador

Una entidad que realiza la verificación de una Credencial Verificable para comprobar su validez, consistencia, etc. Un ejemplo puede ser el sistema digital de un empleador que verifica la validez de un título universitario antes de decidir contratar a una persona.

Estructura

Las credenciales verificables siguen un formato estandarizado, normalmente en **JSON-LD**, que facilita la interoperabilidad entre sistemas.

JSON-LD (*JavaScript Object Notation for Linked Data*) es una extensión del formato JSON que permite representar datos con significado semántico, utilizando contextos (`@context`) que definen el significado de cada campo. Esto facilita la interoperabilidad entre sistemas al proporcionar una forma estándar de interpretar la información.

Componentes de una Credencial Verificable

A continuación, se detallan los elementos clave de una credencial verificable, con sus funciones y ejemplos:

`@context`

Este campo define el significado semántico de los términos utilizados en la credencial. Utiliza vocabularios establecidos y permite que los sistemas que reciben la credencial puedan interpretar correctamente su contenido, incluso si no conocen previamente su estructura exacta.

Ejemplo:

```
"@context": [
  "https://www.w3.org/2018/credentials/v1",
  "https://gob.do/contexts/driver-license"
]
```

`type`

El campo type define el tipo de credencial. Toda credencial verificable debe incluir `"VerifiableCredential"` como tipo base, y puede agregar tipos adicionales que indiquen su propósito específico, como `"DriverLicenseCredential"`, `"UniversityDegreeCredential"`, etc.

Ejemplo:

```
"type": ["VerifiableCredential", "DriverLicenseCredential"]
```

issuer

Es el identificador del emisor, es decir, la entidad que firma digitalmente la credencial. Suele representarse mediante un **DID (Decentralized Identifier)** o una URL que puede resolverse para obtener la clave pública del emisor. El verificador necesita este valor para buscar la clave pública del emisor y validar la firma de la credencial. También permite a los sistemas confiar en la fuente, ya que sabrán si la credencial proviene de una entidad oficial.

Ejemplo:

```
"issuer": "https://gob.do/dgeem"
```

issuanceDate

Es la fecha de emisión de la credencial. Debe expresarse en formato ISO 8601, y permite controlar la vigencia de la credencial en combinación con un campo opcional de expiración (`expirationDate` dentro del `credentialSubject`).

Ejemplo:

```
"issuanceDate": "2025-05-26T00:00:00Z"
```

credentialSubject

Contiene la información que se afirma sobre la entidad (persona, empresa, objeto, etc.). Es el contenido central de la credencial. Puede incluir campos como nombre, identificación, rol, atributos, permisos, etc. Siempre debe incluir un campo `id`, que representa al sujeto(normalmente con un DID). Dependiendo del caso de suo, puede contener información personal u organizacional.

Ejemplo:

```
"credentialSubject": {
  "id": "did:example:juanperez",
  "name": "Juan Pérez",
  "licenseNumber": "A1234567",
  "category": "Private vehicle",
  "expirationDate": "2030-05-26"
}
```

proof

El campo `proof` es lo que **convierte un simple documento JSON en una credencial verificable**, ya que contiene la **firma digital** que garantiza que el contenido no ha sido alterado y que fue emitido por una entidad confiable.

Ejemplo:

```
"proof": {
  "type": "Ed25519Signature2018",
  "created": "2025-05-26T00:00:00Z",
  "proofPurpose": "assertionMethod",
  "verificationMethod": "https://gob.do/keys/clave-publica.json",
  "jws": "eyJhbGciOiJFZERTQSJ9...firma..."
}
```

El `proof` se compone de tales componentes:

- `type`: especifica el **algoritmo de firma** o el método criptográfico que se usó para generar la firma de la credencial. Algoritmos que se usan mucho en estos son `"Ed25519Signature2018"`, `"RsaSignature2018"` o `"BbsBlsSignature2020"`.
- `created`: Es la **fecha y hora exacta** en la que se generó la firma digital. Debe estar en formato ISO 8601 (ej. `"2025-05-26T14:23:12Z"`).
- `proofPurpose`: Define **el propósito de la firma**, es decir, **qué intención tiene el emisor al firmar esta credencial**.Los valores típicos para este campo pueden ser:
  - `"assertionMethod"` – el emisor afirma que el contenido es verdadero (el más común en VC).
  - `"authentication"` – usado para firmar pruebas de que alguien es quien dice ser (ej. en autenticación con DIDs).
  - `"capabilityDelegation"` / `"capabilityInvocation"` – usados en sistemas de control de acceso.
- `verificationMethod`: Es un **identificador (generalmente una URL o DID)** que apunta a la **clave pública** que se debe usar para verificar la firma.
- `jws`: Contiene la **firma digital** como una cadena codificada en formato **JWS (JSON Web Signature)**, que es parte del estándar de JOSE (JSON Object Signing and Encryption). Se representa como una cadena Base64 que incluye: un encabezado (con algoritmo y tipo), el contenido firmado (payload), La firma propiamente dicha.

Flujo de Trabajo

Arquitectura del Proyecto

Esta documentación describe la arquitectura técnica para la emisión, gestión y verificación de una Licencia de Conducir Digital en la República Dominicana. El diseño se basa en el modelo de **Identidad Auto-Soberana (SSI)**, utilizando los estándares de **Credenciales Verificables (VCs)** del W3C para garantizar la seguridad, interoperabilidad y el control del ciudadano sobre sus datos.

Actores Principales y sus Roles

En este ecosistema, cada entidad juega un papel crucial:

- **Propietario (Holder):** Es el **ciudadano** titular de la licencia de conducir. Utiliza la aplicación móvil **"Soy Yo RD" (Carpeta Ciudadana)** para recibir, almacenar y presentar su licencia digital de forma segura.
- **Emisor (Issuer):** Es el **Instituto Nacional de Tránsito y Transporte Terrestre (INTRANT)**. Esta entidad es la única autoridad responsable de verificar la identidad del ciudadano, validar el cumplimiento de los requisitos para conducir y emitir la licencia de conducir en formato de Credencial Verificable.
- **Verificador (Verifier):** Son las entidades que necesitan comprobar la validez de la licencia de conducir. Principalmente, serán los agentes de la **Dirección General de Seguridad de Tránsito y Transporte Terrestre (DIGESETT)**, pero también podrían ser otras entidades como compañías de alquiler de vehículos o aseguradoras.
- **Proveedor de Identidad (Identity Provider):** Es **"Cuenta Única"**, el sistema de autenticación centralizado del Estado Dominicano. Se utiliza para que el ciudadano pueda autenticarse de forma segura ante el INTRANT antes de solicitar su licencia digital.
- **Desarrollador y Mantenedor:** La **Oficina Gubernamental de Tecnologías de la Información y Comunicación (OGTIC)** es la responsable del desarrollo y mantenimiento de la wallet "Soy Yo RD" y de la integración de los componentes tecnológicos necesarios.

Componentes Tecnológicos Clave

La arquitectura integra un conjunto de tecnologías de código abierto, principalmente de la plataforma **Inji**, que facilitan la implementación del modelo de VCs.

- **Soy Yo RD (Wallet/Carpeta Ciudadana):** Es la aplicación móvil (wallet) del ciudadano. Basada en **Mimoto**, esta aplicación permite:
  - Generar y gestionar pares de claves criptográficas.

    - Crear y gestionar Identificadores Descentralizados (DIDs).
    - Solicitar la emisión de la licencia al INTRANT.
    - Almacenar de forma segura la Licencia de Conducir Verificable.
    - Crear Presentaciones Verificables (VPs) para compartir la información de la licencia con un verificador, sin ceder el control de la credencial.
- **Inji-Certify (Portal de Emisión):** Es la plataforma que utilizará el **INTRANT** para emitir las licencias. Inji-Certify proporciona una interfaz para:
    - Definir el esquema de la credencial (los campos que contendrá la licencia: nombre, número de licencia, categorías, fecha de expiración, etc.).
    - Conectarse con la base de datos interna del INTRANT para obtener los datos del conductor.
    - Firmar criptográficamente la credencial con la clave privada del INTRANT, convirtiéndola en una Credencial Verificable a prueba de manipulaciones.
    - Enviar la credencial firmada de forma segura al wallet "Soy Yo RD" del ciudadano.
- **Registro de DIDs (DID Registry):** Es un componente fundamental que funciona como un libro de contabilidad descentralizado (puede ser una blockchain o una base de datos distribuida). Su función es almacenar los **DIDs** y sus documentos DID asociados. Un documento DID contiene la clave pública del emisor (INTRANT), permitiendo que cualquier verificador pueda encontrarla para comprobar la autenticidad de la firma en la licencia. La OGTIC gestionará la infraestructura de este registro.
- **Cuenta Única (IdP):** Actúa como el puente de confianza inicial. Antes de que el INTRANT emita la credencial, el ciudadano debe probar quién es. Lo hará autenticándose con su **Cuenta Única** a través de un protocolo estándar como OpenID Connect (OIDC). Una vez autenticado, el sistema del INTRANT asocia la sesión del usuario con su registro en la base de datos de licencias.

Flujo de Emisión y Verificación

Flujo de Emisión de la Licencia Digital

1. **Inicio (Ciudadano):** El ciudadano abre la app **"Soy Yo RD"** y selecciona la opción para solicitar su licencia de conducir digital.
2. **Autenticación (Ciudadano):** La app redirige al ciudadano al portal de **Cuenta Única**. El usuario ingresa sus credenciales para autenticarse.
3. **Autorización (Ciudadano):** Una vez autenticado, Cuenta Única devuelve una prueba de autenticación (un token) a la app "Soy Yo RD".
4. **Solicitud de Credencial (Wallet):** La app "Soy Yo RD" envía una solicitud de emisión al portal **Inji-Certify** del INTRANT. Esta solicitud incluye el token de autenticación y el DID del ciudadano.
5. **Validación de Datos (INTRANT):** Inji-Certify utiliza el token para verificar la identidad del usuario y busca en la base de datos del INTRANT la licencia de conducir asociada a esa persona.
6. **Creación y Firma (INTRANT):** Con los datos confirmados, Inji-Certify genera la Credencial Verificable con el formato estándar W3C, incluyendo los datos de la licencia, el

DID del ciudadano (sujeto de la credencial) y la firma digital del INTRANT (usando su clave privada).
7. **Emisión (INTRANT):** Inji-Certify envía la VC firmada directamente a la app **"Soy Yo RD"** del ciudadano.
8. **Almacenamiento (Wallet):** La app recibe la VC, verifica la firma del INTRANT usando la clave pública obtenida del registro de DIDs y la almacena de forma segura en el dispositivo del ciudadano.

Flujo de Verificación de la Licencia (Ej. Agente de DIGESETT)

1. **Solicitud de Verificación (Verificador):** El agente de la DIGESETT le solicita al conductor que presente su licencia. El agente muestra un código QR en su dispositivo de verificación.
2. **Presentación (Ciudadano):** El ciudadano abre su app **"Soy Yo RD"**, selecciona su licencia de conducir y escanea el código QR del agente.
3. **Creación de Presentación Verificable (Wallet):** La app del ciudadano crea una **Presentación Verificable (VP)**. Esta VP es un paquete que contiene la VC de la licencia y está firmada con la clave privada del ciudadano, demostrando que él es el portador legítimo y que consiente la presentación.
4. **Envío (Wallet):** La VP se envía al dispositivo del agente a través de un canal seguro (ej. Bluetooth, NFC o a través de la red).
5. **Verificación (Verificador):** El dispositivo del agente realiza dos comprobaciones criptográficas de forma automática:
   - **Verifica la firma del ciudadano en la VP:** Confirma que el portador de la wallet es quien dice ser.
   - **Verifica la firma del INTRANT en la VC:** Busca el DID del INTRANT en el registro público, obtiene su clave pública y confirma que la licencia es auténtica y no ha sido alterada desde su emisión.
6. **Resultado:** El dispositivo del verificador muestra el resultado: **"Válido"** o **"Inválido"**, junto con los datos de la licencia (nombre, foto, categoría, estado).

Gestión y Emisión de la Credencial

Emisión de una Nueva Credencial

Este es el proceso completo, paso a paso, que ocurre desde que el ciudadano inicia la solicitud hasta que recibe su credencial.

Paso 1: Inicio de Sesión y Autenticación Fuerte

El ciudadano necesita probar su identidad al sistema de emisión. Esto se logra mediante el flujo de Código de Autorización de OIDC.

1. **Acción del Usuario**: El usuario presiona "Solicitar Licencia Digital" en la wallet `Soy Yo RD`.
2. **Llamada Inicial (Frontend)**: La wallet `Soy Yo RD` genera e invoca una llamada de autorización a `Cuenta Única`. No es una API REST directa, sino una redirección del navegador en la app (o un Custom Chrome Tab / ASWebAuthenticationSession).
   Ejemplo de URL de Autorización (GET):

   ```
   https://auth.cuentaunica.gob.do/auth?
     response_type=code
     &client_id=SOYYO_RD_WALLET_CLIENT_ID
     &scope=openid profile cedula
     &redirect_uri=soyyord://callback
     &state=STATE_STRING_ALEATORIO
     &nonce=NONCE_STRING_ALEATORIO
     &code_challenge=S256_CHALLENGE
     &code_challenge_method=S256
   ```

3. **Autenticación del Usuario**: El usuario se autentica en la interfaz web de `Cuenta Única` (usuario, contraseña, 2FA).
4. **Redirección con Código**: `Cuenta Única` redirige de vuelta a la app `Soy Yo RD` a través de la `redirect_uri` con un código de autorización.
5. **Intercambio del Código por Tokens (Backend-a-Backend)**: La wallet envía el `code` a su propio backend de soporte (o lo hace directamente si es una app confidencial), el cual realiza una llamada **API REST (POST)** al endpoint de token de `Cuenta Única`. Esta comunicación es segura (backend a backend) y evita exponer los tokens en el lado del

cliente.
Llamada a la API de Token (POST):

- **Endpoint:** `https://auth.cuentaunica.gob.do/token`
- **Headers:** `Content-Type: application/x-www-form-urlencoded`
- **Body:**

```
grant_type=authorization_code
&code=CODIGO_RECIBIDO
&redirect_uri=soyyord://callback
&client_id=SOYYO_RD_WALLET_CLIENT_ID
&client_secret=CLIENT_SECRET_DE_LA_APP
&code_verifier=VERIFIER_ORIGINAL
```

6. **Respuesta de la API de Token**: `Cuenta Úunica` responde con un objeto JSON que contiene los tokens de acceso e identidad.
   Formato de Respuesta (JSON):

```
{
 "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6...",
 "token_type": "Bearer",
 "expires_in": 3600,
 "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6..."
}
```

   El `id_token` es un **JSON Web Token (JWT)** que, al ser decodificado, contiene la información verificada del ciudadano, como su número de Cédula.

Paso 2: Solicitud Formal de la Credencial

Con la prueba de identidad (el `id_token` o `access_token`), la wallet ahora puede solicitar la credencial al servidor de emisión del INTRANT.

1. **Llamada a la API de Emisión (POST)**: La wallet `Soy Yo RD` realiza una llamada HTTPS a la API de `Inji-Certify`.
   - **Endpoint:** `https://api.intrant.gob.do/v1/issue/driving-license`
   - **Headers:** `Authorization: Bearer <access_token_de_cuenta_unica>`
   - **Body (JSON):**

```
{
 "holderDid": "did:key:z6Mkt...H73tias"
}
```

   - **Información Relevante:**
     - El `access_token` en el header autoriza la operación y permite a `Inji-Certify` identificar al solicitante (consultando el endpoint `/userinfo` de Cuenta Única o validando el `id_token`).

- El `holderDid` es el Identificador Descentralizado del ciudadano, generado por la wallet. Será el `id` del sujeto (`credentialSubject`) en la VC.

Paso 3: Verificación Interna y Construcción de la VC

El servidor de emisión ahora tiene todo lo necesario para procesar la solicitud.

1. **Validación de Identidad**: El backend de `Inji-Certify` valida el `access_token` con `Cuenta Única`.
2. **Consulta a la Base de Datos**: Utilizando el número de Cédula extraído del token, el sistema realiza una consulta a la base de datos interna del INTRANT para obtener los datos de la licencia del ciudadano.
   - **Mecanismo**: Conexión de base de datos segura (ej. sobre un túnel VPN) con una consulta tipo `SELECT * FROM licencias WHERE cedula = ?`.
3. **Construcción de la VC**: Si se encuentran datos válidos, `Inji-Certify` ensambla la Credencial Verificable.
   Formato de la VC (JSON-LD):

```
{
 "@context": [
  "https://www.w3.org/2018/credentials/v1",
  "https://w3id.org/vdl/v1" // Contexto específico para licencias de conducir
 ],
 "id": "urn:uuid:3978344f-8596-4c3a-a978-8fcaba3903c5",
 "type": ["VerifiableCredential", "Iso18013DriversLicense"],
 "issuer": "did:web:intrant.gob.do", // DID del INTRANT
 "issuanceDate": "2025-10-26T14:23:14Z",
 "expirationDate": "2029-10-26T14:23:14Z",
 "credentialSubject": {
  "id": "did:key:z6Mkt...H73tias", // DID del ciudadano
  "familyName": "Perez",
  "givenName": "Juan",
  "birthDate": "1990-01-15",
  "issuingCountry": "DO",
  "drivingPrivileges": [
   {
    "vehicleCategory": "B",
    "issueDate": "2021-10-26",
    "expiryDate": "2025-10-26"
   }
  ]
 },
 "proof": { // Este bloque se añade después de firmar
```

```
    "type": "Ed25519Signature2020",
    "created": "2025-10-26T14:23:14Z",
    "verificationMethod": "did:web:intrant.gob.do#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z58D3...2k9fV" // La firma digital
  }
}
```

4. **Firma Criptográfica**: El servidor firma el objeto JSON-LD (excluyendo el bloque `proof`) utilizando la clave privada del INTRANT, almacenada en un **Hardware Security Module (HSM)** para máxima seguridad. El resultado es el `proofValue`.

Paso 4: Devolución de la Credencial

El servidor responde a la llamada API del Paso 2.

- **Respuesta de la API de Emisión (Código 200 OK):**
  - **Body (JSON):** El cuerpo de la respuesta es directamente la Credencial Verificable completa, en formato JSON-LD, tal como se describió arriba.

La wallet `Soy Yo RD` recibe este objeto JSON, valida la firma del emisor (resolviendo el `did:web:intrant.gob.do` para obtener la clave pública) y lo almacena de forma segura en el almacenamiento cifrado del dispositivo.

Gestión del Ciclo de Vida: Revocación

La revocación es crítica. No es suficiente con que una credencial tenga una firma válida; debe estar activa.

- **Mecanismo**: **Status List 2021**. El INTRANT mantiene un endpoint público que expone un bitmap (una larga cadena de bits, 0s y 1s). Cada credencial emitida está asociada a un índice en esta lista.
- **Acción de Revocación**: Cuando un oficial del INTRANT revoca una licencia en el sistema, la acción interna es una **llamada API (PUT o PATCH)** al servicio que gestiona la lista de estado para cambiar el bit en el índice correspondiente de `0` (válido) a `1` (revocado).
  - **Endpoint (interno):** `https://api.intrant.gob.do/v1/statuslist/main`
  - **Body (JSON):**

```
{
  "credentialId": "urn:uuid:3978344f-8596-4c3a-a978-8fcaba3903c5",
  "status": "revoked"
}
```

- **API Pública de la Lista de Estado**: Los verificadores acceden a esta lista a través de un endpoint público definido en la propia VC (propiedad `credentialStatus`).

Llamada del Verificador (GET):

- **Endpoint:** `https://api.intrant.gob.do/v1/status/1` (ejemplo)
- **Respuesta:** Un objeto JSON que contiene el bitmap comprimido. El software del verificador lo usa para comprobar el estado de la credencial que está validando.

Esta arquitectura asegura un flujo de emisión robusto, seguro y estandarizado, donde cada componente se comunica a través de APIs bien definidas, utilizando formatos de datos interoperables como JSON, JWT y JSON-LD.

Gestión del Portador

Esta sección describe los procesos técnicos que ocurren dentro de la aplicación `Soy Yo RD` **(la wallet)**, controlada por el ciudadano (el Portador o *Holder*). Cubre desde la creación de la identidad digital hasta la presentación y gestión de la licencia de conducir verificable.

Caso de Uso 1: Onboarding y Creación de la Identidad Digital

Este es el flujo inicial cuando un ciudadano instala y configura la wallet `Soy Yo RD` por primera vez.

1. **Acción del Usuario**: El ciudadano instala y abre la app `Soy Yo RD`.
2. **Proceso Interno (Generación de Claves)**: La wallet invoca las APIs del sistema operativo para generar un par de claves criptográficas asimétricas directamente en el hardware de almacenamiento seguro.
   - **Mecanismo**: Se utiliza un algoritmo de firma eficiente para móviles, como **Ed25519** .
   - **Resultado**: Se obtiene una `clave privada` (que nunca abandona el almacenamiento seguro) y una `clave pública` correspondiente.
3. **Proceso Interno (Creación del DID)**: La wallet utiliza la clave pública recién generada para crear un Identificador Descentralizado (DID) utilizando un método soportado, como `did:key`.
   - **Formato del DID (`did:key`)**: `did:key:z6Mkt...H73tias` (donde la cadena después de `z6Mk` es una representación en Base58-BTC de la clave pública).
   - **Documento DID (Generado localmente)**: La wallet construye en memoria el Documento DID, que asocia el DID con su clave pública y define los métodos de verificación. Este documento no necesita ser publicado en una red para `did:key`.
4. **Proceso Interno (Creación del Respaldo de Seguridad)**: La wallet genera una frase mnemónica de 12 o 24 palabras **(BIP39)** a partir de la entropía utilizada para crear la clave privada maestra.
   - **Acción del Usuario**: Se le solicita al ciudadano que escriba y guarde esta frase en un lugar físico y seguro. Se le advierte que es la única forma de recuperar su identidad si pierde el dispositivo. La app no guarda esta frase.

Post-condición: El ciudadano tiene una identidad digital auto-soberana, representada por un DID y controlada por una clave privada segura en su dispositivo.

Caso de Uso 2: Almacenamiento y Visualización Segura de la Credencial

Una vez que la credencial ha sido emitida por el INTRANT (como se describió en la documentación anterior) y recibida por la wallet:

1. **Recepción y Verificación Inicial**:
   - **Mecanismo**: La wallet recibe la Credencial Verificable (VC) en formato **JSON-LD** a través de la respuesta de la API de emisión.
   - **Proceso Interno**: Antes de guardarla, la wallet realiza una verificación inmediata:
     1. Resuelve el DID del emisor (`issuer`: `did:web:intrant.gob.do`). Esto implica una llamada **API REST (GET)** a `https://intrant.gob.do/.well-known/did.json` para obtener el Documento DID del INTRANT y su clave pública.
     2. Usa la clave pública del INTRANT para verificar la firma (`proofValue`) de la VC.
     3. Si la firma es válida, procede a guardar. Si no, notifica al usuario de un error.
2. **Almacenamiento Seguro**:
   - **Mecanismo**: La wallet almacena la VC (el archivo JSON-LD) en una base de datos local cifrada (ej. SQLite con SQLCipher). La clave de cifrado de esta base de datos está protegida y gestionada por el Almacenamiento Seguro del dispositivo.
   - **Resultado**: Las credenciales no se pueden leer ni extraer si el dispositivo es comprometido o si la aplicación es accedida sin la autorización del usuario (biometría/PIN).
3. **Visualización**:
   - **Acción del Usuario**: El ciudadano navega a su sección de credenciales y toca la licencia.
   - **Proceso Interno**: La app lee el archivo JSON-LD de la base de datos cifrada, lo decodifica y renderiza los campos (`credentialSubject`) en una interfaz de usuario amigable y visualmente representativa de una licencia de conducir.

Caso de Uso 3: Creación y Presentación para Verificación

Este es el flujo técnico para presentar la licencia a un agente de la DIGESETT.

1. **Inicio del Intercambio (Solicitud de Presentación)**:
   - **Acción del Usuario**: El ciudadano presiona "Presentar Licencia" y la cámara de `Soy Yo RD` se activa para escanear un código QR mostrado por el agente verificador.
   - **Formato del QR**: El código QR no contiene datos personales, sino una solicitud de presentación (Presentation Request). Esta solicitud puede seguir el estándar **W3C Presentation Exchange** o estar encapsulada en un protocolo como **OpenID for Verifiable Presentations (OID4VP)**.

Ejemplo de Payload del QR (OID4VP - URL):

```
openid-vc://?request_uri=https://api.digesett.gob.do/request/12345
```

2. **Procesamiento de la Solicitud**:
   - **Mecanismo**: La wallet `Soy Yo RD` realiza una **API REST (GET)** a la `request_uri` obtenida del QR.
   - **Respuesta de la API (JSON)**: El servidor del verificador responde con un objeto JSON que define qué credenciales se solicitan.
     Ejemplo de Presentation Definition (JSON):

```
{
  "id": "definicion_licencia_1",
  "input_descriptors": [
    {
      "id": "licencia_rd_descriptor",
      "name": "Licencia de Conducir Dominicana",
      "schema": [{ "uri": "https://w3id.org/vdl/v1" }],
      "constraints": {
        "fields": [
          {
            "path": ["$.type"],
            "filter": { "type": "string", "const": "Iso18013DriversLicense" }
          },
          {
            "path": ["$.issuer"],
            "filter": { "type": "string", "pattern": "^did:web:intrant.gob.do$" }
          }
        ]
      }
    }
  ]
}
```

     Este JSON le dice a la wallet: "Necesito una credencial que sea del tipo 'Iso18013DriversLicense' y que haya sido emitida por 'did:web:intrant.gob.do'".
3. **Construcción de la Presentación Verificable (VP)**:
   - **Acción del Usuario**: La wallet encuentra la credencial que coincide con la solicitud y le pide al usuario que autorice compartirla (usando biometría o PIN).
   - **Proceso Interno**: Tras la autorización, la wallet construye una **Presentación Verificable (VP)**.
     Formato de la VP (JSON-LD):

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    // Aquí se inserta la VC completa de la licencia de conducir
    { "@context": [...], "type": [...], "issuer": "did:web:intrant.gob.do", ... }
  ],
  "holder": "did:key:z6Mkt...H73tias", // El DID del ciudadano
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2025-06-17T19:30:00Z",
    "verificationMethod": "did:key:z6Mkt...H73tias#z6Mkt...H73tias",
    "proofPurpose": "authentication",
    "challenge": "CHALLENGE_STRING_DE_LA_SOLICITUD", // Previene ataques de repetición
    "proofValue": "z3aD8...k7gRt" // Firma del ciudadano sobre toda la VP
  }
}
```

- **Punto Clave**: La VP es firmada por el **ciudadano** (`holder`) usando su clave privada. Esto prueba dos cosas: que la credencial original no ha sido alterada (por la firma del INTRANT) y que el portador actual controla el DID del sujeto de la credencial (por la firma del ciudadano).

4. **Envío de la VP al Verificador**:
   - **Mecanismo**: La wallet envía la VP completa al servidor del verificador. Esto se puede hacer a través de una **API REST (POST)** al endpoint que se especificó en el protocolo de intercambio o directamente vía Bluetooth/NFC si la conexión es de proximidad.
   - **Endpoint de envío (ejemplo):** `https://api.digesett.gob.do/submit`
   - **Body (JSON)**: El objeto VP completo.

Caso de Uso 4: Gestión de la Wallet y Recuperación

1. **Respaldo (Backup)**:
   - **Mecanismo**: Como se describió en el onboarding, el único método de respaldo es la **frase mnemónica (BIP39)**. Es responsabilidad del ciudadano guardarla. No hay API ni comunicación externa; es un proceso local.
2. **Recuperación (Restore)**:
   - **Acción del Usuario**: En un teléfono nuevo, el ciudadano instala `Soy Yo RD` y elige "Recuperar Wallet".

- **Proceso Interno**:
  1. La aplicación le pide al usuario que ingrese su frase mnemónica de 12/24 palabras.
  2. Usando el algoritmo BIP39, la wallet re-genera la clave privada maestra original a partir de la frase.
  3. A partir de la clave maestra, deriva exactamente el mismo par de claves (privada/pública) y, por lo tanto, el mismo `did:key` que tenía en el dispositivo anterior.
- **Recuperación de Credenciales**: La identidad está restaurada, pero las credenciales no. La wallet está vacía. El ciudadano debe solicitar la re-emisión de cada credencial (como la licencia de conducir) a sus respectivos emisores, repitiendo el flujo de emisión original. El emisor lo tratará como una nueva solicitud de un DID ya conocido.

Esta arquitectura centrada en el portador le da al ciudadano el control total de su identidad y sus datos, utilizando criptografía estándar y protocolos de comunicación abiertos para interactuar de forma segura con emisores y verificadores.

Verificación de la Credencial

Esta sección describe los flujos técnicos y los mecanismos de comunicación para la verificación de la Licencia de Conducir Digital. El proceso está diseñado para ser seguro, rápido y funcional tanto en escenarios con conexión a internet como sin ella.

Protocolo de Comunicación Principal: OID4VP

Para una máxima interoperabilidad y seguridad, el protocolo recomendado es **OpenID for Verifiable Presentations (OID4VP)**. Este estándar define cómo una aplicación de verificador (Relying Party) solicita una Presentación Verificable (VP) de una wallet (Self-Issued OpenID Provider).

Caso de Uso 1: Verificación Estándar en Línea (Online)

Este es el escenario más común, donde el dispositivo del verificador tiene conexión a internet.

1. **Inicio del Flujo (Generación de la Solicitud)**
   - **Acción del Verificador**: El agente abre su aplicación de verificación y presiona "Escanear Licencia".
   - **Mecanismo Técnico**: La aplicación del verificador genera un código QR único para esta transacción específica. El QR contiene una URL que sigue el esquema OID4VP.
     Ejemplo de contenido del QR (URL):

     ```
     openid-vc://?request_uri=https://api.verificador.digesett.gob.do/vp-request/a7b3c9d1-e2f4-4a5b-8c6d-9e0f1a2b3c4d
     ```

2. **Solicitud de Presentación por parte de la Wallet**
   - **Acción del Portador**: El ciudadano utiliza su wallet `Soy Yo RD` para escanear el código QR.
   - **Llamada API (GET)**: La wallet extrae la `request_uri` y realiza una llamada **HTTP GET** a ese endpoint.
     - **Endpoint**: `https://api.verificador.digesett.gob.do/vp-request/a7b3c9d1-e2f4-4a5b-8c6d-9e0f1a2b3c4d`

- **Respuesta de la API (JSON)**: El servidor del verificador responde con una **Solicitud de Presentación (Presentation Request)** en formato JSON. Esta solicitud contiene una **Definición de Presentación (Presentation Definition)** que especifica los requisitos de la credencial.
  Ejemplo de la Respuesta (Presentation Definition):

```
{
 "response_type": "vp_token",
 "client_id": "digesett_verifier_app_01",
 "presentation_definition": {
  "id": "licencia_rd_pd_1",
  "input_descriptors": [{
   "id": "licencia_descriptor",
   "name": "Licencia de Conducir de la República Dominicana",
   "schema": [{ "uri": "https://w3id.org/vdl/v1" }],
   "constraints": {
    "fields": [
     {
      "path": ["$.type"],
      "filter": { "type": "string", "contains": "Iso18013DriversLicense" }
     },
     {
      "path": ["$.issuer"],
      "filter": { "type": "string", "pattern": "^did:web:intrant.gob.do$" }
     }
    ]
   }
  }]
 },
 "nonce": "n-0S6_WzA2Mj" // Nonce para prevenir ataques de repetición
}
```

3. **Construcción y Envío de la Presentación**
   - **Proceso Interno de la Wallet**:
     1. La wallet analiza la `presentation_definition` y busca en su almacenamiento una VC que cumpla con los `constraints`.
     2. Encuentra la licencia del INTRANT, la selecciona y pide autorización al ciudadano (PIN/biometría).
     3. Construye una **Presentación Verificable (VP)**, firmándola con la clave privada del portador e incluyendo el `nonce` de la solicitud para seguridad.
   - **Llamada API (POST)**: La wallet envía la VP al servidor del verificador. El endpoint para el envío suele ser parte de la configuración del protocolo OID4VP o se define en la solicitud.

- **Endpoint**: `https://api.verificador.digesett.gob.do/vp-response`
- **Body (Form-urlencoded)**:

```
vp_token=<LA_VP_COMPLETA_EN_FORMATO_JWT_O_JSON-LD>
&presentation_submission=<DESCRIPTOR_MAP_JSON>
```

4. **Proceso de Verificación Criptográfica (Backend del Verificador)** Al recibir la VP, el sistema del verificador ejecuta una serie de comprobaciones automáticas en milisegundos:
   - **Paso 4.1: Verificar la Firma del Portador (Holder)**
     - **Objetivo**: Confirmar que la presentación fue autorizada por el legítimo propietario de la wallet.
     - **Mecanismo**: Extrae el DID del portador (`holder` en la VP), obtiene su clave pública (del campo `verificationMethod` si es `did:key`, o resolviéndolo) y verifica la firma externa de la VP. Se comprueba también que el `nonce` coincide con el enviado en la solicitud.
   - **Paso 4.2: Verificar la Firma del Emisor (Issuer)**
     - **Objetivo**: Confirmar que la licencia es auténtica, no ha sido alterada y fue emitida por el INTRANT.
     - **Mecanismo**: Extrae la VC de la VP. Resuelve el DID del emisor (`issuer`: `did:web:intrant.gob.do`) realizando una llamada **GET** a `https://intrant.gob.do/.well-known/did.json`. Usa la clave pública obtenida para validar la firma interna de la VC.
   - **Paso 4.3: Verificar el Estado de la Credencial (Revocación)**
     - **Objetivo**: Asegurarse de que la licencia no ha sido revocada por el INTRANT.
     - **Mecanismo**:
       1. Lee el bloque `credentialStatus` dentro de la VC.
       2. Realiza una llamada **HTTP GET** a la URL indicada (ej. `https://api.intrant.gob.do/status/1`).
       3. Recibe una lista de estado (ej. StatusList2021) que contiene un bitmap.
       4. Comprueba el bit en la posición correspondiente al `statusPurpose` y `statusListIndex` de la credencial. Si el bit es `1`, la credencial está revocada.
5. **Resultado Final**
   - La aplicación del verificador recibe una respuesta final de su backend y muestra un resultado claro al agente:
     - **VERDE (VÁLIDO)**: Muestra la foto, nombre y datos de la licencia.
     - **ROJO (INVÁLIDO)**: Muestra el motivo del fallo (ej. "Firma del emisor inválida", "Licencia revocada", "Licencia expirada").

Caso de Uso 2: Verificación sin Conexión (Offline)

Este escenario es crucial para operaciones en áreas rurales o con mala cobertura de red.

1. **Inicio y Solicitud**: El flujo es similar, pero el intercambio de datos debe ser directo entre dispositivos (Peer-to-Peer).
   - **Mecanismo de Intercambio**: En lugar de una `request_uri`, el QR puede contener la **solicitud de presentación completa** en formato JSON. La comunicación entre dispositivos se establece usando **Bluetooth Low Energy (BLE)**, **NFC** o **Wi-Fi Direct**. La especificación **OpenID Connect for Verifiable Presentations (SIOPv2)** detalla estos flujos P2P.
2. **Proceso de Verificación (en la App del Verificador)** La aplicación del verificador debe poder realizar las comprobaciones criptográficas sin conexión a internet.
   - **Paso 2.1: Verificar Firma del Portador (POSIBLE OFFLINE)**
     - Esta verificación es puramente matemática y no requiere red. Se puede completar siempre.
   - **Paso 2.2: Verificar Firma del Emisor (POSIBLE OFFLINE)**
     - Para que esto funcione offline, la aplicación del verificador debe tener **pre-cargada (en caché) la clave pública del INTRANT**. Se puede actualizar periódicamente cuando la app tiene conexión. Así, no necesita resolver el `did:web` en tiempo real.
   - **Paso 2.3: Verificar Estado de Revocación (NO ES POSIBLE OFFLINE)**
     - La comprobación del estado en tiempo real contra una lista en un servidor es imposible sin conexión.
3. **Resultado Final (Offline)** La aplicación del verificador debe presentar un resultado diferenciado:
   - **AMARILLO (PARCIALMENTE VERIFICADO)**: Muestra los datos de la licencia (foto, nombre) y un aviso claro: **"Autenticidad Verificada. Estado de Revocación no pudo ser comprobado. Última sincronización: [Fecha/Hora]"**.
   - **ROJO (INVÁLIDO)**: Si la firma del portador o del emisor falla, el resultado es inválido independientemente del estado de conexión.

Caso de Uso 3: Verificación con Divulgación Selectiva

Un verificador (ej. un bar) solo necesita saber si una persona es mayor de 18 años, no su dirección o nombre completo.

1. **Solicitud de Presentación Específica**: La `presentation_definition` generada por el verificador solicitará una "prueba de edad derivada" en lugar de la credencial completa.
   - **Mecanismo**: Utilizando técnicas como **Zero-Knowledge Proofs (ZKP)** o solicitando una credencial derivada firmada por el portador.
   - **Ejemplo de `presentation_definition` con predicados (si el estándar lo soporta):**

```
"constraints": {
  "fields": [{
    "path": ["$.credentialSubject.birthDate"],
```

```
    "purpose": "Necesitamos verificar que eres mayor de 18 años.",
    "predicate": { // Indica que se necesita una prueba, no el valor
      "type": "date",
      "operator": "<=",
      "value": "2007-06-17" // Fecha de hoy hace 18 años
    }
  }]
}
```

2. **Respuesta de la Wallet**: La wallet `Soy Yo RD` interpreta esta solicitud, realiza el cálculo localmente y genera una presentación que solo afirma: **"El campo `birthDate` en la credencial emitida por el INTRANT es anterior o igual a 2007-06-17: VERDADERO"**. Esta presentación está firmada por el portador y por el emisor (indirectamente a través de la VC original), sin revelar la fecha de nacimiento exacta.
3. **Resultado**: El verificador solo recibe un "Sí" o "No" a la pregunta que hizo, protegiendo al máximo la privacidad del ciudadano.

Gestión del Ciclo de Vida y Seguridad

Esta sección aborda los casos de uso administrativos y de seguridad que son fundamentales para garantizar la integridad, confiabilidad y sostenibilidad a largo plazo del ecosistema de la licencia de conducir digital. Estos procesos son gestionados por personal técnico y de seguridad de la **OGTIC** y del **INTRANT**.

Caso de Uso 1: Gestión del Esquema de la Credencial (Schema)

Este caso de uso se activa cuando los requisitos de negocio o legales de la licencia de conducir cambian (ej. se añade un campo nuevo como "Donante de Órganos").

- **Actor**: Administrador de Credenciales (INTRANT), con aprobación de OGTIC.
- **Trigger**: Decisión administrativa o cambio en la normativa de tránsito.
- **Mecanismo Técnico**:
  1. **Definición de la Nueva Versión**: Se crea una nueva versión del esquema JSON que define la estructura de la VC. Se le asigna un nuevo identificador de versión o URI.
     JSON

     ```
     // old_schema_uri: "https://intrant.gob.do/schemas/licencia/v1.0"
     // new_schema_uri: "https://intrant.gob.do/schemas/licencia/v1.1"
     ```

  2. **Actualización del Portal de Emisión**: El administrador actualiza la configuración en el portal `Inji-Certify` a través de una interfaz administrativa segura.
     - **Llamada API (Interna)**: Se podría ejecutar una llamada `PUT` a un endpoint administrativo.
       - **Endpoint**: `https://admin.inji-certify.intrant.gob.do/api/v1/schemas/driving-license`
       - **Body (JSON)**:
         JSON

```
{
  "schemaUri": "https://intrant.gob.do/schemas/licencia/v1.1",
  "schemaDefinition": { ... nuevo esquema JSON ... },
  "isActive": true
}
```

3. **Plan de Transición**:
   - **Nuevas Emisiones**: Todas las licencias emitidas o renovadas a partir de este punto utilizarán el nuevo esquema v1.1.
   - **Credenciales Antiguas**: Las licencias existentes (v1.0) siguen siendo válidas hasta su fecha de expiración o hasta que sean renovadas. Las aplicaciones de los verificadores deben estar programadas para entender y validar ambas versiones del esquema.
- **Impacto en el Ecosistema**: Requiere coordinación para que las aplicaciones de verificadores se actualicen y puedan interpretar correctamente tanto el esquema antiguo como el nuevo.

Caso de Uso 2: Gestión de Claves Criptográficas del Emisor (INTRANT)

Este es uno de los procesos más críticos para la seguridad y confianza de todo el sistema.

- **Actor**: Oficial de Seguridad (OGTIC/INTRANT).
- **Trigger**:
  - **Rotación Programada**: Política de seguridad que exige cambiar las claves cada 1-2 años.
  - **Compromiso de Clave**: Sospecha o confirmación de que la clave privada ha sido expuesta.
- **Mecanismo Técnico (Rotación Programada)**:
  1. **Generación de Nueva Clave**: Se genera un nuevo par de claves (privada/pública) dentro del **HSM**. La nueva clave privada nunca sale del HSM.
  2. **Actualización del Documento DID**: Se debe actualizar el documento DID del emisor (`did:web:intrant.gob.do`) para reflejar este cambio. El `did.json` se modifica para:
     - Añadir la nueva clave pública en la sección `verificationMethod`.
     - Mantener la clave antigua en la misma sección, pero marcándola como histórica o expirada si es posible.
     - Actualizar las secciones `assertionMethod` y `authentication` para que apunten a la nueva clave.

     Llamada API (para publicar el did.json en el servidor web):
     PUT /var/www/html/.well-known/did.json (Ejemplo de despliegue en servidor web)

3. **Activación de la Nueva Clave**: El portal `Inji-Certify` se configura para usar la nueva clave (`key-2`) para firmar todas las nuevas credenciales.
- **Mecanismo Técnico (Compromiso de Clave de Emergencia)**:
  1. **Revocación Inmediata**: El Documento DID se actualiza inmediatamente para eliminar la clave comprometida de `assertionMethod`. Esto invalida su uso para nuevas firmas.
  2. **Plan de Re-emisión**: Se debe notificar a todos los ciudadanos que necesitan solicitar una re-emisión de su licencia, ya que las antiguas, aunque no expiradas, podrían no ser confiables.
  3. **Actualización de Verificadores**: Las aplicaciones de los verificadores deben forzar una actualización de la caché de la clave del INTRANT para asegurarse de que ya no confían en la clave comprometida.
- **Impacto**: Una rotación bien planificada es transparente para los usuarios. Un compromiso de clave es un incidente de seguridad grave que requiere una comunicación pública clara y una acción rápida de re-emisión.

Caso de Uso 3: Mantenimiento de la Lista de Estado de Credenciales

Garantiza que el mecanismo de revocación funcione de manera eficiente y escalable.

- **Actor**: Administrador del Sistema (OGTIC).
- **Trigger**: La lista de estado actual (bitmap) se está llenando o ha alcanzado un tamaño que afecta el rendimiento.
- **Mecanismo Técnico**:
  1. **Creación de una Nueva Lista**: Se genera una nueva lista de estado vacía.
     - **Llamada API (Interna)**: `POST https://api.intrant.gob.do/v1/statuslists`
     - **Respuesta**: Devuelve el ID y la URL de la nueva lista (ej. `.../status/2`).
  2. **Configuración del Emisor**: El portal `Inji-Certify` se configura para que todas las nuevas credenciales emitidas apunten a esta nueva lista en su propiedad `credentialStatus`.
  3. **Archivado de la Lista Antigua**: La lista antigua (`.../status/1`) se mantiene activa y disponible para que las credenciales ya emitidas que apuntan a ella puedan seguir siendo verificadas, pero ya no se le añaden nuevas revocaciones.
- **Impacto**: Es una tarea de mantenimiento rutinario y transparente para los usuarios finales, pero crucial para el rendimiento y la escalabilidad del sistema de revocación.

Caso de Uso 4: Monitoreo, Auditoría y Alertas del Sistema

Es el proceso continuo de vigilancia para asegurar la salud y seguridad del ecosistema.

- **Actor**: Administrador del Sistema y Oficial de Seguridad (OGTIC).
- **Trigger**: Continuo (24/7).
- **Mecanismo Técnico**:
  1. **Recolección de Logs**: Todos los componentes (servidores de API, HSM, portal de emisión, base de datos) envían sus logs a un sistema **SIEM** centralizado. Los logs deben incluir:
     - Intentos de acceso (exitosos y fallidos).
     - Cada emisión de credencial (quién la autorizó, a quién, cuándo).
     - Cada verificación de estado de revocación.
     - Errores del sistema y métricas de rendimiento (latencia, tasa de errores).
  2. **Creación de Paneles (Dashboards)**: Se configuran paneles en el SIEM o en herramientas como Grafana para visualizar en tiempo real:
     - Disponibilidad (Uptime) de las APIs críticas.
     - Número de credenciales emitidas por día/hora.
     - Latencia promedio de las verificaciones.
     - Mapa de geolocalización de las solicitudes de verificación.
  3. **Configuración de Alertas**: Se definen reglas para generar alertas automáticas (vía email, Slack, PagerDuty) ante eventos anómalos.
     - **Ejemplos de Alertas de Seguridad**:
       - Múltiples intentos de inicio de sesión fallidos para un administrador.
       - Emisión de credenciales fuera del horario laboral normal.
       - Una tasa de error del 5% o más en la API de verificación.
       - Un intento de acceso al HSM desde una IP no autorizada.
- **Impacto**: Permite la detección proactiva de problemas de rendimiento y brechas de seguridad, reduciendo el tiempo de respuesta ante incidentes. Genera un registro de auditoría inmutable crucial para investigaciones forenses.