Documento Técnico de Diseño e Implementación: Integración de Atala Prims(Hyperledger Identus) con el Sistema "Digital Voter ID"

Propuesta: Digital voter ID using Atala Prism (Proof-of-Concept)

Autor: David Tacuri

Date: 4 Agosto 2025

Version: 1.0

1. Introducción al Sistema Digital Voter ID y la Identidad Autosoberana (SSI)

1.1. Propósito y Alcance del Documento

El objetivo principal de este documento es servir como una guía técnica completa para la integración de Hyperledger Identus con el proyecto "Digital Voter ID". Se busca proporcionar una comprensión profunda de la arquitectura, los flujos de trabajo, las tecnologías subyacentes y las consideraciones prácticas para su implementación. Este informe está dirigido a desarrolladores, arquitectos de software y líderes técnicos involucrados en el diseño, desarrollo y despliegue de soluciones de identidad descentralizada.

1.2. Conceptos Fundamentales de Identidad Autosoberana (SSI) y su Relevancia

La Identidad Autosoberana (SSI) es un paradigma de identidad digital que otorga a los individuos **control total sobre su información personal y sus credenciales**. A diferencia de los sistemas de identidad tradicionales, que dependen de autoridades centralizadas, SSI permite a los usuarios decidir cuándo, con quién y qué datos compartir, mejorando la privacidad y reduciendo la dependencia de terceros.

Los componentes clave de SSI incluyen:

- **Identificadores Descentralizados (DIDs)**: Identificadores únicos y controlados por el usuario, que no dependen de una autoridad centralizada para su registro. Un DID se asocia con un Documento DID, que contiene material criptográfico (como claves públicas), métodos de verificación y servicios que permiten a un controlador de DID probar su control sobre el mismo. Los DIDs se resuelven en un Registro de Datos Verificables (VDR).

- **Credenciales Verificables (VCs)**: Afirmaciones digitales firmadas criptográficamente por un emisor, que atestiguan atributos sobre un sujeto. Las VCs están diseñadas para ser criptográficamente seguras, respetar la privacidad y ser verificables por máquinas.

- **Agentes de Identidad (Agents)**: Componentes de software que gestionan DIDs, VCs y la comunicación en el ecosistema SSI. Se distinguen principalmente tres

roles: el **Agente Emisor (Issuer Agent)** que emite VCs, el **Agente Titular (Holder Agent)** que las almacena y presenta, y el **Agente Verificador (Verifier Agent)** que las solicita y valida.

• **Protocolos de Comunicación Segura (DIDComm)**: Permiten el intercambio privado y seguro de mensajes entre los diferentes agentes.

La relevancia de SSI para el proyecto "Digital Voter ID" es fundamental, ya que garantiza la seguridad, escalabilidad, transparencia y descentralización en la gestión de la identidad digital de los votantes, reduciendo el riesgo de suplantación de identidad.

1.3. Visión General del Proyecto "Digital Voter ID" y sus Objetivos

El proyecto "Digital Voter ID" busca transformar los procesos de afiliación política y votación mediante un sistema digital que mejora la transparencia, seguridad y accesibilidad. Propone implementar un sistema de afiliación política basado en SSI, utilizando Atala Prims ahora conocido como Hyperledger Identus y la blockchain de Cardano.

Los **objetivos técnicos específicos** de la integración de Hyperledger Identus son:

• **Implementar la soberanía de los datos**: Permitir que los ciudadanos (Holders) tengan control total sobre su información personal y decidan cuándo y con quién compartir sus credenciales de identidad.

• **Proporcionar identidad digital verificable**: Crear credenciales verificables (VCs) para la identificación digital de votantes, basadas en el estándar W3C, que puedan ser emitidas y verificadas criptográficamente.

• **Utilizar la blockchain de Cardano como VDR**: Aprovechar la inmutabilidad y la naturaleza permisiva de la blockchain de Cardano para registrar y resolver DIDs de emisores, garantizando la confianza y la transparencia.

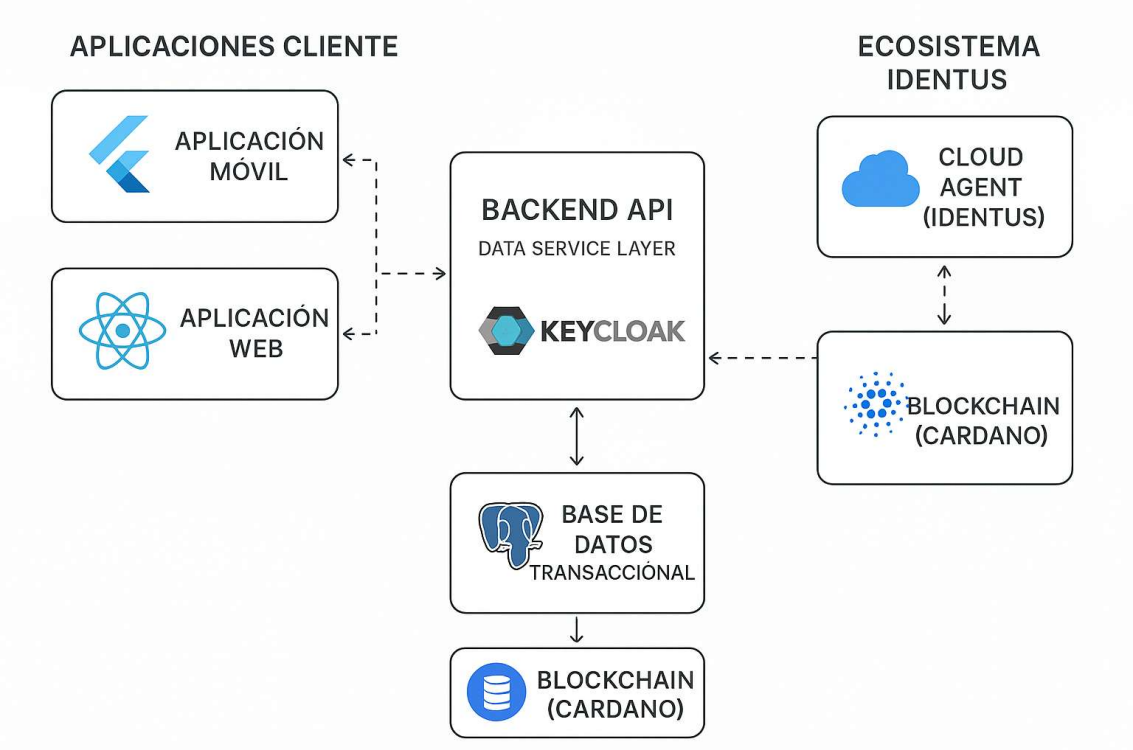
• **Establecer canales de comunicación seguros**: Habilitar la comunicación peer-to-peer segura entre las diferentes partes (emisores, holders, verificadores) utilizando el protocolo DIDComm.

• **Facilitar la gestión de afiliaciones políticas**: Integrar las capacidades de emisión y verificación de credenciales para gestionar las afiliaciones y desvinculaciones de partidos políticos de manera segura y transparente.

2. Arquitectura General del Sistema y Componentes Clave

2.1. Diagrama de Arquitectura General

El sistema "Digital Voter ID" se basa en una arquitectura distribuida que integra componentes centralizados con el ecosistema descentralizado de Hyperledger Identus y Cardano.



Descripción del diagrama: Los componentes principales y sus interacciones se estructuran de la siguiente manera:

- **Aplicaciones Cliente**: Incluyen una **Aplicación Móvil** (desarrollada con Flutter para ciudadanos) y una **Aplicación Web** (desarrollada con React para gestión de organizaciones políticas y administración electoral).

- **Backend API (Data Service Layer)**: Una capa de API Backend que gestiona la lógica de negocio y actúa como un puente entre las aplicaciones cliente, el ecosistema Identus y la base de datos (el backend transaccional con Asp.Net Core)

- **Base de Datos Transaccional (PostgreSQL)**: Para el almacenamiento de datos operativos del sistema (usuarios, afiliaciones, organizaciones, respaldos temporales de credenciales).

• **Sistema de Autenticación (Keycloak)**: Para la gestión de usuarios y roles, proporcionando autenticación y autorización con control basado en roles (RBAC) y soporte multi-inquilino.

• **Ecosistema Identus**: El **Cloud Agent (Identus)** es el componente central de SSI, interactuando con la **Blockchain (Cardano)** como Registro de Datos Verificables (VDR).

Los flujos de alto nivel implican que las aplicaciones cliente interactúan con el API Backend para la lógica de negocio y la autenticación.

El API Backend a su vez se comunica con el Cloud Agent de Identus para todas las operaciones relacionadas con SSI (gestión de DIDs y VCs), y con la Blockchain de Cardano (ya sea a través del Cloud Agent o directamente para ciertas operaciones) para el registro y la verificación de datos inmutables. Keycloak se encarga de la autenticación y autorización de los usuarios para acceder a las funcionalidades de las aplicaciones.

2.2. Componentes del Sistema "Digital Voter ID"

2.2.1. Aplicaciones Cliente (Móvil Flutter y Web React)

Las aplicaciones cliente son los puntos de interacción directos para los usuarios y administradores del sistema.

Aplicación Móvil (Flutter App): Diseñada para el ciudadano.

• **Framework**: Flutter SDK (Dart).

• **Funcionalidades Clave**: Registro de usuario, creación y gestión de la wallet descentralizada y el DID asociado, selección de partido político, visualización de credenciales de afiliación y presentación de las mismas para verificación. También envía alertas instantáneas sobre cambios en el estado de afiliación.

• **Mejores Prácticas de Seguridad**:

- **Almacenamiento Seguro**: Utilizar flutter_secure_storage para almacenar de forma segura claves privadas, DIDs y VCs en el dispositivo del usuario, aprovechando las capacidades de almacenamiento seguro del sistema operativo (Keychain en iOS, Keystore en Android). Se debe evitar el hardcoding de secretos directamente en el código.
- **Autenticación Biométrica**: Integración de librerías como biometric_storage o local_auth para proteger el acceso a datos sensibles y operaciones SSI mediante huella dactilar o reconocimiento facial.
- **Comunicación Segura**: Todas las comunicaciones de red deben realizarse exclusivamente a través de **HTTPS** para garantizar el cifrado de datos en tránsito y prevenir ataques de intermediario.

- **Validación de Entrada**: Implementar validación rigurosa de toda la entrada del usuario tanto en el cliente como en el servidor para prevenir vulnerabilidades.
- **Ofuscación de Código**: El proceso de compilación de Flutter permite ofuscar el código Dart, dificultando la ingeniería inversa.
- **Actualización de Dependencias**: Mantener el SDK de Flutter y todas las dependencias actualizadas para incluir parches de seguridad. La seguridad en la aplicación móvil es crucial, ya que un compromiso de la "wallet" (aplicación móvil) pone en riesgo la soberanía de la identidad del usuario.

Aplicación Web (React Web App): Destinada a usuarios internos y externos, facilita la gestión avanzada de afiliaciones políticas.

- **Framework**: React.js (TypeScript).
- **Funcionalidades Clave**: Interfaz para que las organizaciones políticas emitan credenciales de afiliación, herramientas de auditoría para verificar el estado de las afiliaciones y un panel de administración electoral. Los ciudadanos pueden consultar su estado de registro y acceder a estadísticas.
- **Mejores Prácticas de Seguridad**:
  - **Integración con SDKs de Identus**: El SDK TypeScript de Identus (@hyperledger/identus-sdk) es fundamental para que la aplicación web interactúe con el Cloud Agent para operaciones de DID y VC, incluyendo módulos como Apollo (criptografía), Castor (DIDs), Mercury (DIDComm V2) y Pluto (almacenamiento agnóstico).
  - **Gestión de Estado**: Librerías como Redux, MobX, Recoil o Zustand para gestionar el estado global de la aplicación.
  - **Autenticación**: Integración con Keycloak utilizando librerías como Auth0 React SDK o JWT para manejar la autenticación y protección de rutas.
  - **Protección XSS/CSRF**: React realiza un escape automático de contenido HTML, pero la sanitización manual con sanitize-html es necesaria si se permite la inserción de HTML no confiable. Implementación de tokens CSRF en las peticiones al backend es vital.
  - **Uso de HTTPS**: Todas las comunicaciones con el backend y el Agente Identus deben ser a través de HTTPS.
  - **Minimizar Datos Sensibles en Frontend**: Evitar la serialización de datos sensibles directamente en el frontend y asegurar que la lógica de autorización resida en el backend. La aplicación web debe ser tratada como un "cliente ligero" en términos de seguridad, delegando las operaciones SSI críticas al backend y al Agente Cloud de Identus.

2.2.2. Backend API y Base de Datos Transaccional (PostgreSQL)

Estos componentes forman el núcleo lógico y de persistencia del sistema "Digital Voter ID".

Backend API (Data Service Layer): Actúa como la lógica de negocio central.

- **Funcionalidades**: Validar afiliaciones políticas, generar credenciales verificables, gestionar los procesos de revocación, asociar DIDs con usuarios internos del sistema y servir como punto de integración para la comunicación con el Agente Identus.
- **Frameworks Sugeridos**:
    - **Node.js**: NestJS (TypeScript-first, modular, escalable) o Express.js (minimalista y flexible). La interacción con Identus se puede realizar con el paquete @hyperledger/identus-cloud-agent-client.
    - **ASP.NET Core**: C# perfilando el alto rendimiento, tipado fuerte, integración nativa con Entity Framework Core para PostgreSQL, soporte avanzado de seguridad y API documentada con **Swagger/OpenAPI** para facilitar pruebas e integración con clientes externos.
    - **Java**: Spring Boot (simplifica el desarrollo de aplicaciones Spring, microservicios, REST APIs, JPA). La interacción con Identus se realizaría mediante llamadas HTTP directas a su API REST.

Base de Datos Transaccional (PostgreSQL): Almacena datos internos del sistema como usuarios, afiliaciones, organizaciones y respaldos temporales de credenciales.

- Es crucial que esta base de datos **no almacene las VCs completas ni las claves privadas de los DIDs,** tampoco debe almacenar **información biométrica (huellas, rostros, patrones faciales, etc.),** ya que estos datos deben ser procesados y verificados localmente en el dispositivo del ciudadano y descartados después de la validación. Su función es gestionar los datos transaccionales y los metadatos que enlazan la lógica de negocio tradicional con el ecosistema SSI. Las VCs se almacenan en el Holder Agent (aplicación móvil del ciudadano) y los DIDs se publican en el VDR (blockchain de Cardano).
- **Diseño de Esquema**: Se requerirán tablas para usuarios (con un campo para el DID asociado), organizaciones_politicas, afiliaciones (enlazando usuario y organización), y posiblemente tablas para gestionar el estado de revocación interno o metadatos de VCs.

2.2.3. Sistema de Autenticación y Gestión de Identidades (Keycloak)

Keycloak es una solución de Gestión de Identidad y Acceso (IAM) de código abierto que proporciona autenticación y autorización con control basado en roles (RBAC), gestión de usuarios y soporte multi-inquilino.

- **Integración**: Keycloak puede integrarse con el backend y las aplicaciones cliente a través de protocolos estándar como OpenID Connect (OIDC) y OAuth 2.0. Esto permite Single Sign-On (SSO), federación de usuarios y gestión centralizada de políticas de acceso.
- **SPIs Personalizados (Service Provider Interfaces)**: Keycloak permite la creación de SPIs personalizados. Esto abre la posibilidad de desarrollar un SPI que permita la **autenticación de usuarios utilizando DIDs y Credenciales Verificables**, en lugar de credenciales tradicionales, para el proyecto Digital Voter ID usaremos Authenticator SPI .

2.3. Ecosistema Hyperledger Identus (Atala PRISM)

Hyperledger Identus, construido sobre el robusto protocolo Atala PRISM, es un conjunto integral de herramientas para desarrollar soluciones de identidad descentralizada. Opera como una solución de capa 2 sobre una DLT, utilizando la blockchain como un Registro de Datos Verificables (VDR).

2.3.1. Agente Identus (Cloud Agent): Funcionalidades y Módulos

El **Agente Identus** es el componente central para todas las operaciones de SSI dentro del sistema, actuando como un contenedor que gestiona DIDs, la emisión, posesión y verificación de credenciales. Sus módulos clave, según la estructura del SDK TypeScript, incluyen:

- **Apollo**: Proporciona las operaciones criptográficas necesarias, incluyendo soporte para curvas como Ed25519.
- **Castor**: Gestiona todas las operaciones relacionadas con los DIDs, incluyendo su creación, gestión y resolución.
- **Mercury**: Se encarga del manejo de mensajes DIDComm V2, facilitando la comunicación segura entre agentes.
- **Pluto**: Ofrece una interfaz agnóstica para operaciones de almacenamiento, permitiendo flexibilidad en la persistencia de datos internos del agente.
- **Agent (componente de alto nivel)**: Combina las funcionalidades de los bloques anteriores para proporcionar capacidades básicas de agente de borde (edge agent), incluyendo la implementación de protocolos DIDComm V2.

2.3.2. Protocolo DIDComm V2: Comunicación Segura Peer-to-Peer

DIDComm es el protocolo de comunicación seguro y privado utilizado por los agentes Identus para intercambiar mensajes de credenciales e invitaciones de conexión. Es fundamental para establecer interacciones verificables entre individuos, organizaciones y dispositivos. Un componente clave de DIDComm en Identus es el **Mediator**. El Mediator facilita el enrutamiento de mensajes cifrados entre agentes, especialmente

cuando estos no están en línea directamente (como las aplicaciones móviles). Permite que los mensajes se almacenen y se recojan cuando el Holder se conecte, lo cual es fundamental para la usabilidad y fiabilidad de la aplicación móvil "Digital Voter ID".

2.3.3. Nodo PRISM y su Interacción con la Blockchain de Cardano

El **Prism Node** (o el nodo interno del agente Identus) es el módulo dentro del agente responsable de interactuar directamente con la blockchain de Cardano, que funciona como el **Registro de Datos Verificables (VDR)** del sistema. Cardano se elige como VDR por sus características de **bajos costos de transacción, alta seguridad y compatibilidad con sistemas de identidad autosoberana (SSI)**. Técnicamente, Cardano aporta al proyecto "Digital Voter ID" de varias maneras:

- **Registro Inmutable de Afiliaciones**: La blockchain de Cardano se utiliza para registrar de forma inmutable las afiliaciones de los ciudadanos a organizaciones políticas, garantizando la trazabilidad, transparencia y auditabilidad de cada transacción.
- **Publicación de DIDs**: Cardano sirve como el registro donde los emisores publican sus DIDs, los cuales pueden ser accedidos por cualquier persona para verificar la identidad del emisor.
- **Contratos Inteligentes (Smart Contracts)**: Se pueden emplear contratos inteligentes sobre la blockchain de Cardano (utilizando Aiken) para automatizar validaciones y auditorías relacionadas con el registro de afiliaciones.
- **Mecanismo de Confianza Descentralizado**: La naturaleza pública y sin permisos de Cardano significa que cualquiera puede unirse a la red y verificar la existencia de un DID publicado, reforzando la confianza en el sistema al eliminar la necesidad de intermediarios centralizados para la validación de identidad.
- Para funcionar como VDR, se configuran varios componentes de Cardano:
- **Nodo Cardano**: Un nodo de Cardano se establece para comunicarse directamente con la blockchain de Cardano, siendo el punto de entrada para interactuar con la cadena de bloques.
- **Cardano DB-Sync**: Este componente incluye una base de datos (PostgreSQL) y un proceso que lee toda la información de los bloques de Cardano y la guarda en la base de datos, permitiendo consultas y recuperación de datos de la blockchain de manera eficiente.
- **Cardano Wallet**: Se necesita una billetera Cardano para cubrir las pequeñas tarifas de transacción al publicar DIDs u otras operaciones en la blockchain. Esta billetera es interna para el agente Identus.
- **Node IPC (Inter-Process Communication)**: Una conexión de socket llamada node socket facilita la comunicación entre el Cardano Wallet, el DB-Sync y el

nodo de Cardano, permitiendo la sumisión de transacciones y la obtención de información de los bloques.

2.3.4. Base de Datos Interna de Identus (PostgreSQL) y Vault Server

El Agente Identus utiliza su propia instancia de base de datos PostgreSQL para el almacenamiento interno de datos operativos. Esta base de datos es independiente de Cardano DB-Sync y se utiliza para gestionar DIDs, credenciales y estados de conexión internos del agente. Adicionalmente, el **Vault Server** es un componente opcional, pero crucial para entornos de producción, que se utiliza para el **almacenamiento seguro de información sensible de credenciales, como claves privadas**.

La distinción entre estas bases de datos es importante: la primera almacena el estado operativo del agente (DIDs generados localmente, conexiones DIDComm activas, VCs poseídas), mientras que la segunda es una réplica de los datos públicos de la blockchain.

2.3.5. API Gateway y Swagger UI

El Agente Identus expone sus funcionalidades a través de una **API Gateway (API 6)**, que actúa como un proxy inverso. Esta *gateway* enruta el tráfico a los diferentes subcomponentes del agente Identus y expone las APIs RESTful para la interacción externa.

Swagger UI es una interfaz gráfica que se proporciona junto con la API Gateway para explorar y probar las APIs REST expuestas por el agente Identus. Esta herramienta es invaluable durante las fases de desarrollo y depuración, mitigando la falta de documentación detallada externa.

3. Guía Técnica de Implementación Detallada

3.1. Configuración del Entorno Cardano como Registro de Datos Verificables (VDR)

La implementación del VDR con Identus en Cardano implica el despliegue y configuración de varios componentes de Cardano, siendo **Docker Compose** el método recomendado para simplificar este proceso.

3.1.1. Despliegue de Componentes Cardano (Nodo, DB-Sync, Wallet) con Docker Compose

La forma más sencilla de configurar el entorno Cardano para Identus es utilizando Docker Compose. Este enfoque permite levantar todos los servicios necesarios en contenedores aislados, facilitando la gestión y replicación del entorno.

Pasos Generales para el despliegue:

1. Instalación de Docker y Docker Compose: Asegurarse de tener Docker (v17.06.2-ce o superior) y Docker Compose (v1.14.0 o superior) instalados.

2. Clonación del Repositorio Identus: El repositorio principal de Hyperledger Identus (hyperledger-identus/hyperledger-identus) contiene la configuración de Docker Compose en el directorio identus-docker.

3. Configuración de Variables de Entorno: El agente Identus requiere variables de entorno específicas para su interacción con Cardano. Se deben definir NODE_LEDGER para indicar que Cardano será el VDR y DOCKER_HOST para permitir la comunicación con los servicios de Cardano que se ejecutan en el host de Docker. Además, se deben especificar KEYCLOAK_ENABLED, KEYCLOAK_URL, KEYCLOAK_REALM, KEYCLOAK_CLIENT_ID, y KEYCLOAK_CLIENT_SECRET para la configuración de la autorización con IAM externo.

4. Ejecución de Docker Compose: Utilizar el archivo docker-compose.yml proporcionado para levantar los servicios de PostgreSQL (para DB-Sync), el nodo Cardano, la billetera Cardano (interna al agente Identus) e Icarus (una interfaz gráfica para la billetera).

5. Sincronización del Nodo Cardano: Una vez levantados los servicios, el nodo de Cardano debe sincronizarse con la red (preprod o mainnet). Este proceso puede llevar varias horas, aunque se puede acelerar utilizando instantáneas (snapshots).

- **Verificar la sincronización**: Se puede verificar el estado del nodo usando cardano-cli query tip --testnet-magic 1.
- **Monitoreo del progreso**: Los logs del Docker del nodo (Docker logs) y la consulta del tip pueden mostrar el progreso de sincronización.

6. Configuración y sincronización de Cardano DB Sync con PostgreSQL: Se utiliza cardano-db-sync para sincronizar los datos de la blockchain con una base de datos PostgreSQL, facilitando el acceso y la consulta de datos en tiempo real.

- **Preparar PostgreSQL**: Instalar y ejecutar PostgreSQL. Crear una base de datos y un usuario con permisos. Es conveniente usar un archivo .pgpass.
- **Arrancar Cardano DB Sync**: Se inicia el servicio de sincronización, que leerá bloques desde el génesis de preprod y poblará la base de datos PostgreSQL. El progreso se puede monitorear en los logs.
- **Consulta de datos on-chain vía GraphQL**: Se puede desplegar Cardano GraphQL (que usa Hasura) sobre la base de datos de db-sync para realizar consultas eficientes a la blockchain.

3.1.2. Financiamiento de Wallet Cardano para Tarifas de Transacción

Es esencial que la billetera Cardano asociada al Agente Identus (por ejemplo, prism-wallet-issuer) tenga fondos suficientes (ADA) para cubrir las pequeñas tarifas de transacción (fees) al publicar DIDs u otras operaciones en la blockchain.

Proceso:

1. Creación de la Billetera: La billetera puede crearse utilizando la interfaz de Icarus o mediante comandos de línea de Cardano (CLI).

2. Envío de Fondos de Prueba: En entornos de desarrollo o prueba, se deben enviar fondos de prueba (test ADA) a una dirección asociada a esta billetera, a menudo obtenidos de un faucet de Cardano Testnet.

3. Configuración del Agente Identus: El ID de la billetera Cardano y su frase de acceso (passphrase) deben asignarse a las variables de entorno del agente Identus, junto con una dirección de pago. Esto permite que el agente realice transacciones on-chain de forma autónoma.

3.2. Gestión del Ciclo de Vida de Identificadores Descentralizados (DIDs)

La gestión de DIDs es un aspecto central de cualquier implementación SSI.

3.2.1. Creación y Publicación de DIDs (Formato Largo a Corto)

- **Creación Local**: El proceso comienza con la interacción del DID Controller (backend o aplicación cliente) con el Cloud Agent de Identus. El Agente Identus gestiona las claves privadas y el contenido de la operación de creación, permitiendo al Controller especificar claves públicas y servicios asociados. Los **PRISM DIDs** pueden crearse completamente *offline* sin interactuar con la blockchain.
- **Publicación en Cardano**: Una vez creado localmente, el DID puede publicarse en la blockchain de Cardano. Esta publicación transforma el DID de un "**formato largo**" (long-form DID), que es autocontenido, a un "**formato corto**" (short-form DID), que requiere resolución en la blockchain para obtener su Documento DID completo.
- **Mecanismo**: El agente Identus utiliza la **clave MASTER** del DID para firmar la operación de publicación y la envía a la blockchain de Cardano. Este proceso es **asincrónico y requiere un número específico de bloques de confirmación (actualmente 112)** en la blockchain para que la operación se considere procesada y publicada.
- **Monitoreo**: Es posible monitorear los registros del agente Identus y la propia blockchain para verificar el progreso de la publicación del DID hasta que su estado cambie a PUBLISHED.

3.2.2. Resolución de DIDs y Verificación de Emisores

- La resolución de DIDs es el proceso de obtener el Documento DID asociado a un DID.
- Cualquier cliente puede consultar un nodo PRISM (o un agente Identus) para obtener la información asociada a un DID.

- Los DID Resolvers utilizan la salida de los nodos PRISM para construir el DID Document actual, que contiene las claves públicas y los servicios necesarios para interactuar con el sujeto del DID.
- En la verificación de credenciales, la resolución del DID del emisor en la blockchain es un paso crítico para establecer la confianza en la credencial.

3.2.3. Actualización, Rotación y Desactivación de DIDs

Los DIDs no son estáticos; sus documentos asociados pueden necesitar ser actualizados.

• **Actualización**: Los usuarios pueden actualizar los Documentos DID publicando operaciones de actualización *on-chain*. Esto es necesario para añadir o modificar servicios, o rotar claves criptográficas.

• **Rotación de Claves**: La capacidad de rotar las claves asociadas a un DID es una práctica de seguridad fundamental. Permite reemplazar una clave comprometida con una nueva sin cambiar el DID, mitigando el riesgo proactivamente.

• **Desactivación**: Un DID puede ser desactivado si el sujeto ya no desea utilizarlo o si se detecta un compromiso grave. La desactivación se realiza publicando una operación de desactivación *on-chain*. Una vez que el DID es desactivado, todo el contenido del Documento DID se elimina.

Tabla: Operaciones DID y su Implementación con Identus (APIs/SDKs)

Operación DID	Componente Identus/Cardano	API/SDK Relevante	Descripción Técnica Breve	Estado en Blockchain
Creación	Cloud Agent (módulo Castor)	API REST del Cloud Agent, SDKs Identus (TypeScript Castor.createDID())	El DID Controller interactúa con el Cloud Agent para generar un DID y sus claves localmente.	Local, no *on-chain* hasta la publicación.
Publicación	Cloud Agent (Prism Node), Cardano VDR	API REST del Cloud Agent, SDKs Identus (TypeScript Castor.publishDID())	El agente Identus firma la operación de publicación con la clave MASTER y la envía a	*On-chain*, requiere 112 bloques de confirmación.

			Cardano. Convierte DID de formato largo a corto.	
Resolución	Prism Node, Cardano DB-Sync	DID Resolvers, API REST del Cloud Agent (para resolución local/cacheada)	Cualquier cliente o agente consulta un nodo PRISM para obtener el Documento DID. Cardano DB-Sync acelera las consultas *on-chain*.	*On-chain* (para el Documento DID publicado) y *off-chain* (para resolución eficiente).
Actualización	Cloud Agent (Prism Node), Cardano VDR	API REST del Cloud Agent, SDKs Identus (TypeScript Castor.updateDID())	Se publican operaciones de actualización *on-chain* para modificar el Documento DID (ej. añadir/eliminar claves, servicios).	*On-chain*, requiere confirmación.
Rotación de Claves	Cloud Agent (Prism Node), Cardano VDR	API REST del Cloud Agent, SDKs Identus (TypeScript Castor.rotateKey())	Una forma específica de actualización donde se reemplazan las claves criptográficas asociadas al DID.	*On-chain*, requiere confirmación.
Desactivación	Cloud Agent (Prism Node), Cardano VDR	API REST del Cloud Agent, SDKs Identus (TypeScript Castor.deactivateDID())	Se publica una operación *on-chain* para marcar un DID	*On-chain*, requiere confirmación.

			como desactivado, impidiendo su uso futuro.	

3.3. Implementación del Ciclo de Vida de Credenciales Verificables (VCs)

Las Credenciales Verificables son el medio por el cual las afirmaciones sobre la identidad de un votante se emiten, almacenan, presentan y verifican en el sistema.

3.3.1. Emisión de Credenciales de Afiliación (Claims, Firma Digital)

- **Flujo**: El proceso de emisión comienza cuando un ciudadano selecciona un partido político en la aplicación móvil. El backend valida que no existan afiliaciones previas. El Agente Emisor genera una credencial verificable (VC) que contiene los "claims" (atributos) del usuario, como nombre, fecha de afiliación, partido político y un hash de consentimiento.
- **Firma Digital**: La VC es firmada criptográficamente por el Agente Emisor para garantizar su autenticidad e integridad. Una vez firmada, la VC se envía al Agente Titular (la wallet del ciudadano) utilizando el **protocolo DIDComm**, asegurando una transmisión segura y privada.
- **Estándares**: Los "claims" dentro de la VC deben adherirse al modelo de datos de Credenciales Verificables del W3C.

3.3.2. Almacenamiento Seguro de VCs en el Agente Titular (Holder)

El Agente Titular, que reside en la aplicación móvil del ciudadano, es el responsable de almacenar de forma segura las VCs recibidas. Este almacenamiento es crítico para la soberanía del ciudadano sobre su identidad.

- **Mecanismo**: En la aplicación móvil Flutter, esto implica el uso de la librería flutter_secure_storage para almacenar las VCs cifradas directamente en el dispositivo, aprovechando las capacidades de almacenamiento seguro del sistema operativo, como Keychain en iOS y Keystore en Android.

3.3.3. Verificación de Credenciales por Terceros (Firma, Revocación, DID Emisor)

La verificación es el proceso mediante el cual una entidad (Verificador) valida la autenticidad y validez de una VC presentada por un Holder.

- **Flujo**: Una entidad verificadora (por ejemplo, un organismo electoral) solicita la validación de la afiliación de un ciudadano. El usuario, a través de su aplicación móvil, aprueba compartir su VC. El Agente Verificador de Identus realiza las siguientes comprobaciones:
  1. **Validación de la Firma Digital:** Verifica que la VC no ha sido alterada y que fue firmada por el Agente Emisor legítimo.

2. **Estado de Revocación:** Consulta el estado de revocación de la VC para asegurarse de que no ha sido invalidada.
3. **Resolución del DID del Emisor: Resuelve** el DID del emisor en la blockchain de Cardano para obtener su Documento DID y verificar la clave pública utilizada para la firma.

3.3.4. Revocación de Credenciales (StatusList2021)

La **revocación de credenciales** es un mecanismo esencial para invalidar VCs que ya no son válidas, ya sea por fraude, cambio de estado (como una des-afiliación) o expiración.

- Si se detecta fraude o el usuario desea des-afiliarse, la VC se revoca utilizando un **mecanismo de lista de estado (StatusList2021)**, un estándar del W3C.
- **Proceso**: El backend actualiza su estado interno para reflejar la revocación, y el Agente Emisor publica la revocación en la blockchain de Cardano. Los verificadores que consulten esa VC encontrarán que está marcada como revocada.
    - Cada VC revocable incluye un campo credentialStatus que apunta a una Status List Credential pública, que a su vez contiene un *bitstring* codificado. Cada posición en el *bitstring* representa el estado de una VC específica (1 para revocado, 0 para válido).
    - **Proceso de verificación (lado Verificador)**: El verificador resuelve la StatusListCredential, verifica su autenticidad e integridad (firma), decodifica el *bitstring* y consulta el bit correspondiente a la VC para determinar su estado.
    - **Proceso de revocación (lado Emisor)**: Solo el emisor puede revocar una credencial. El agente emisor actualiza el bit correspondiente en la StatusListCredential y la publicación se propaga de forma pasiva cuando los verificadores consultan el recurso actualizado.

Tabla: Ciclo de Vida de VC y Métodos de Implementación (APIs/SDKs)

Operación VC	Componente Identus/Aplicación	API/SDK Relevante	Descripción Técnica Breve	Estándares/Protocolos
Emisión	Agente Emisor (Cloud Agent), Backend API	API REST del Cloud Agent, SDKs Identus (TypeScript Agent.issueVC(), Mercury para DIDComm)	El Agente Emisor genera y firma criptográficamente una VC con los *claims* del Holder y la	W3C Verifiable Credentials, DIDComm V2.

			envía vía DIDComm.	
Almacenamiento (Holder)	Agente Titular (Aplicación Móvil)	flutter_secure_storage (Flutter), Pluto module (SDK TS)	La VC se almacena de forma segura en la *wallet* del Holder en el dispositivo del usuario.	Almacenamiento seguro del SO.
Presentación (Holder)	Agente Titular (Aplicación Móvil)	SDKs Identus (TypeScript Agent.presentVC(), Mercury para DIDComm)	El Holder crea una Presentación Verificable (VP) a partir de una o más VCs y la firma con su clave DID para presentarla a un Verificador.	W3C Verifiable Presentations, DIDComm V2.
Verificación	Agente Verificador (Cloud Agent), Backend API	API REST del Cloud Agent, SDKs Identus (TypeScript Agent.verifyVC())	El Verificador valida la firma de la VC/VP, resuelve el DID del emisor en el VDR y comprueba el estado de revocación.	W3C Verifiable Credentials, W3C DIDs.
Revocación	Agente Emisor (Cloud Agent), Backend API	API REST del Cloud Agent (para gestión de revocación)	El Agente Emisor publica un estado de revocación para una VC específica en el VDR, invalidándola, utilizando el	StatusList2021 (W3C).

			mecanismo StatusList2021.	

3.4. Integración de Componentes de Aplicación Específicos

La integración efectiva de los diferentes componentes del sistema "Digital Voter ID" es fundamental para su funcionalidad y rendimiento.

3.4.1. Aplicación Móvil Flutter: Uso de SDKs Identus, Almacenamiento Seguro, Autenticación Biométrica

La aplicación móvil Flutter es la interfaz principal para el ciudadano (Holder).

- **Interacción con Identus**: Para la creación de DIDs, la gestión de la *wallet* y la emisión/presentación de VCs, la aplicación Flutter interactuará con el Agente Identus. Esto se logrará utilizando el **SDK Kotlin Multiplatform (KMP) para Android y el SDK Swift para iOS de Hyperledger Identus**.
- **Almacenamiento Seguro**: Se implementará flutter_secure_storage para proteger las claves privadas del DID del usuario y las credenciales verificables en el dispositivo.
- **Autenticación Biométrica**: La integración de local_auth o biometric_storage permitirá a los usuarios autenticarse con huella dactilar o reconocimiento facial antes de acceder a datos sensibles o realizar operaciones criptográficas con sus DIDs/VCs.

3.4.2. Aplicación Web React: Uso de SDKs Identus (TypeScript), Gestión de Estado, Mejores Prácticas de Seguridad Frontend (XSS, CSRF)

La aplicación web React servirá como interfaz para los administradores y las organizaciones emisoras/verificadoras.

- **Interacción con Identus**: La aplicación web React interactuará con el Cloud Agent de Identus principalmente a través del Backend API. Para funcionalidades directas de frontend, el **SDK TypeScript de Identus** (@hyperledger/identus-sdk) puede ser utilizado.
- **Gestión de Estado**: Se recomienda el uso de librerías de gestión de estado como Redux Toolkit, MobX, Recoil o Zustand.
- **Seguridad Frontend**: Implementación de protección XSS y CSRF.

3.4.3. Backend API (Node.js/Python/Java): Selección de Frameworks (NestJS/FastAPI/Spring Boot), Interacción con la API del Agente Identus

El Backend API es el orquestador central del sistema "Digital Voter ID".

- **Opciones de Frameworks**: NestJS (TypeScript) es adecuados para microservicios centrados en API de alto rendimiento.

- **Interacción con Identus**: Para backends en Node.js (TypeScript/JavaScript), se recomienda utilizar el paquete @hyperledger/identus-cloud-agent-client. Para Python y Java, la interacción se realizaría mediante llamadas HTTP directas a su API REST, debido a la ausencia de SDKs oficiales en estos lenguajes en los recursos investigados.

3.4.4. Base de Datos Transaccional PostgreSQL: Diseño de Esquema para Usuarios, Afiliaciones, DIDs y VCs

La base de datos PostgreSQL se utilizará para almacenar la lógica de negocio tradicional del sistema "Digital Voter ID", **no los DIDs o VCs en sí mismos**, sino sus metadatos y las relaciones con los usuarios y afiliaciones.

- **Consideraciones de Diseño de Esquema**: Incluye tablas para Usuarios, DIDs, Organizaciones_Politicas, Afiliaciones, y opcionalmente Credential_Schemas.

3.4.5. Integración de Keycloak: Autenticación y Autorización Basada en Roles, Posibles SPIs Personalizados para Autenticación DID

Keycloak es fundamental para la autenticación y autorización de los usuarios y organizaciones.

  - **Autenticación Tradicional y Autorización Basada en Roles**: Keycloak gestionará la autenticación de los usuarios y el control basado en roles para la autorización.
  - **Extensión con SPIs Personalizados para Autenticación DID**: Se puede desarrollar un AuthenticatorFactory y un Authenticator SPI personalizado en Java para permitir la autenticación de usuarios presentando una Presentación Verificable (VP) firmada con su DID, en lugar de un nombre de usuario y contraseña tradicional.

Flujo Propuesto para Autenticación DID con Keycloak:

1. Inicio de Sesión y Desafío DID.
2. Generación de VP por la *wallet* SSI del usuario.
3. Envío de VP a Keycloak.
4. Verificación de VP por el SPI personalizado, incluyendo validación de firma, resolución del DID del presentador y comprobación de revocación.
5. Autenticación y Emisión de Token JWT estándar por Keycloak.

4. Consideraciones de Desarrollo y Mejores Prácticas

4.1. Configuración Detallada del Agente Identus (Variables de Entorno, Wallet, DIDComm)

La **configuración correcta del Agente Identus** es fundamental para su operación.

- **Variables de Entorno**: Configurar NODE_LEDGER para Cardano VDR y DOCKER_HOST para comunicación entre servicios Docker.
- **Wallet Cardano**: Asignar el ID de la billetera Cardano y su frase de acceso a variables de entorno del agente Identus, junto con una dirección de pago.
- **DIDComm**: Configurar el protocolo DIDComm para comunicación segura, incluyendo mediadores si los agentes no están siempre en línea.
- **Modo de Desarrollo**: Habilitarlo durante fases de desarrollo y prueba, pero **deshabilitarlo estrictamente en entornos de producción** debido a implicaciones de seguridad.

4.2. Interacción con la API del Agente Identus (Ejemplos de Uso de Swagger UI y SDKs)

La interacción programática con el Agente Identus es fundamental para el backend y las aplicaciones cliente.

- **Swagger UI**: Es una herramienta esencial para explorar *endpoints*, comprender modelos de datos y probar interacciones de la API directamente, especialmente útil dada la inaccesibilidad de documentación detallada en línea.
- **Uso de SDKs**: Los SDKs de Identus (TypeScript, Kotlin, Swift) son la forma preferida de interactuar programáticamente con el Cloud Agent, simplificando la gestión de DIDs, VCs y la comunicación DIDComm.

4.3. Estrategias de Escalabilidad y Optimización de Rendimiento

La escalabilidad y el rendimiento son consideraciones críticas para un sistema de identidad nacional.

- **Microservicios**: Adoptar una arquitectura de microservicios permite el escalado independiente de cada servicio.
- **Servicios Stateless**: Diseñar los servicios backend para que sean sin estado facilita el escalado horizontal.
- **Bases de Datos**: Implementar estrategias de escalado de base de datos como *sharding*, réplicas de lectura, o bases de datos NoSQL.
- **Procesamiento Asíncrono y Colas de Mensajes (opcional para la prueba de concepto)**: Utilizar colas de mensajes (Kafka, RabbitMQ) para manejar tareas que no requieren una respuesta inmediata (ej. publicación de DIDs).
- **Balanceadores de Carga y API Gateways (opcional para prueba de concepto)**: Implementar balanceadores de carga para distribuir el tráfico a múltiples instancias del backend y del Agente Identus.
- **Caching (opcional para prueba de concepto)**: Implementar estrategias de caché para datos frecuentemente accedidos.

4.4. Seguridad de Claves Privadas y Gestión de Semillas

La **seguridad de las claves privadas y la gestión de las frases semilla** son aspectos críticos para la integridad y la confianza en un sistema SSI.

- **Almacenamiento Seguro de Claves MASTER**: La clave MASTER del DID es fundamental para firmar operaciones y debe ser protegida con el máximo nivel de seguridad. En producción, se recomienda el uso de un Vault Server o un Módulo de Seguridad de Hardware (HSM).
- **Gestión de Frases Semilla (Seed Phrases)**: Para los Holder Agents (aplicaciones móviles), la gestión de las frases semilla es vital. La pérdida de una frase semilla puede significar la pérdida irreversible del acceso a la identidad digital. Es crucial implementar mecanismos de **recuperación de identidad robustos** (por ejemplo, recuperación social, computación multipartita) y educar exhaustivamente a los usuarios sobre la importancia de almacenar estas frases de forma segura.
- **Rotación de Claves**: Implementar mecanismos para la rotación periódica de las claves de los DIDs, según lo permitido por el método PRISM DID.
- **Monitoreo Continuo**: La seguridad de las claves privadas, la gestión de semillas, la rotación de claves y el monitoreo de *logs* del sistema y de la blockchain exige una operación activa y continua.

5. Análisis Crítico del Sistema

5.1. Ventajas Detalladas de la Integración SSI con Identus y Cardano

La integración de Hyperledger Identus con Cardano para el sistema "Digital Voter ID" ofrece una serie de ventajas significativas:

- **Soberanía del Ciudadano**: Control completo sobre su identidad y datos personales, eliminando la dependencia de entidades centralizadas.
- **Auditoría Transparente e Inmutable**: Todas las operaciones críticas se registran de forma inmutable en la blockchain de Cardano, garantizando trazabilidad y auditabilidad pública.
- **Seguridad Criptográfica Avanzada**: Uso de tecnologías criptográficas de vanguardia, incluyendo curvas como Ed25519, DIDComm, JWT y firma digital.
- **Escalabilidad Modular y Multi-inquilino**: Identus está diseñado con una arquitectura modular que permite configurar múltiples *wallets* y *tenants*, facilitando la adopción y escalando la solución eficientemente.
- **Interoperabilidad Global**: El sistema se basa en estándares abiertos y ampliamente aceptados (W3C DID, VC-JWT, DIDComm V2).

- **Respaldo de Cardano**: La robustez, seguridad y naturaleza descentralizada de Cardano como Registro de Datos Verificables proporciona una base sólida de confianza para todo el sistema.

5.2. Desventajas y Desafíos Técnicos Identificados

A pesar de sus ventajas, la implementación de un sistema SSI con Identus y Cardano presenta desafíos técnicos y operativos:

- **Complejidad Tecnológica**: Requiere conocimiento especializado en blockchain, criptografía, contenedores Docker y arquitectura distribuida.
- **Latencia de Blockchain**: La confirmación de transacciones en Cardano puede demorar entre 3 y 10 minutos (112 bloques), afectando la experiencia del usuario.
- **Barrera de Acceso Digital**: El sistema requiere que los ciudadanos tengan acceso a dispositivos inteligentes y conexión a internet estable, lo que puede generar una brecha digital.
- **Curva de Aprendizaje de los Agentes**: La interacción y los flujos entre los diferentes roles de agentes (emisor, titular, verificador) son novedosos y requieren capacitación inicial.
- **Dependencia de Servicios de Terceros**: Aunque descentralizado, aún puede depender de la disponibilidad y el rendimiento de ciertos servicios (nodos de Cardano, APIs del Agente Identus).
- **Requiere Supervisión Continua**: La gestión de la seguridad de claves privadas, frases semilla, rotación de claves y monitoreo constante exige operación activa de personal especializado.
- **Inaccesibilidad de Documentación Detallada**: Una observación recurrente es la dificultad para encontrar guías detalladas y ejemplos de código para varios componentes clave de Hyperledger Identus, lo que aumenta la curva de aprendizaje y el tiempo de implementación.

5.3. Riesgos Potenciales y Estrategias de Mitigación

A continuación, se presenta la tabla detallada de riesgos identificados y sus estrategias de mitigación, conforme a la arquitectura propuesta para el sistema "Digital Voter ID". Esta tabla expande los riesgos previamente mencionados, proporcionando estrategias de mitigación más específicas y técnicas.

Tabla: Riesgos Identificados y Estrategias de Mitigación

Riesgo	Descripción	Impacto Potencial	Estrategia de Mitigación

Desincronización de nodo Cardano	El nodo Cardano local o del Agente Identus pierde la sincronización con la red principal.	Fallo en la publicación y validación de DIDs y VCs, interrupción de servicios.	Monitoreo proactivo del estado de sincronización del nodo. Implementar mecanismos de fallback a nodos públicos confiables o a servicios de resolución de DIDs externos. Utilizar instantáneas para acelerar la recuperación.
Pérdida de frase semilla (Holder)	Los ciudadanos pierden la frase semilla de su wallet SSI.	Pérdida irreversible del acceso a su identidad digital y a sus credenciales.	Implementar mecanismos de recuperación de identidad (e.g., recuperación social multifirma, custodia compartida de claves). Educación exhaustiva al usuario sobre la importancia y el manejo seguro de la frase semilla.
Compromiso de claves privadas (Emisor/Verificador)	Las claves privadas de los Agentes Emisor o Verificador son robadas o comprometidas.	Emisión fraudulenta de VCs, revocación maliciosa, suplantación de identidad.	Almacenamiento de claves en Vault Server o HSM en producción. Implementación de rotación periódica de claves de DIDs. Monitoreo de actividad de claves y alertas ante patrones anómalos.
Ataques a la API del Agente Identus	Vulnerabilidades en la API REST del Agente Identus o en el API Gateway.	Acceso no autorizado a operaciones de DID/VC, manipulación de datos.	Implementar autenticación robusta (API Keys, OAuth 2.0) y autorización granular para la API. Uso de WAF (Web Application Firewall). Auditorías de seguridad regulares y pruebas de penetración.

Latencia de confirmación de Cardano	El tiempo prolongado para la confirmación de transacciones en Cardano afecta la experiencia del usuario (UX).	Frustración del usuario, procesos de negocio lentos.	Diseñar flujos de usuario asíncronos con feedback visual claro. Implementar mecanismos de caché para datos de lectura frecuentes. Optimizar la lógica de negocio para no depender de confirmaciones inmediatas para todas las operaciones.
Inaccesibilidad de documentación técnica	Dificultad para encontrar guías detalladas y ejemplos de código para la implementación.	Aumento del tiempo de desarrollo, mayor curva de aprendizaje, posibles errores de implementación.	Fomentar la exploración activa de Swagger UI del Agente Identus. Participación en la comunidad Hyperledger Identus (Discord, foros). Contribución a la documentación de código abierto.
Dependencia del Mediator DIDComm	Fallos o indisponibilidad del servicio Mediator de Identus.	Interrupción de la comunicación DIDComm entre agentes (especialmente móviles).	Despliegue del Mediator con alta disponibilidad y redundancia. Monitoreo del estado del Mediator. Considerar múltiples Mediators o un diseño que permita la comunicación directa cuando sea posible.
Ataques de suplantación de identidad (en aplicaciones)	Fraude de identidad a nivel de aplicación (ej. si la app móvil es comprometida).	Acceso no autorizado a funciones del sistema, uso fraudulento de credenciales.	Implementar autenticación biométrica fuerte. Uso de flutter_secure_storage para proteger datos sensibles en el dispositivo. Ofuscación de código y detección de manipulación de la aplicación.

Vulnerabilidades en el Backend API	Fallos de seguridad en la lógica de negocio o en la interacción con Identus/Keycloak.	Exposición de datos, manipulación de afiliaciones, acceso no autorizado.	Adherencia estricta a las mejores prácticas de seguridad para el framework elegido (ej. validación de entrada, protección XSS/CSRF, gestión segura de tokens). Auditorías de código y pruebas de penetración.
Problemas de interoperabilidad	Dificultad para integrar con otros sistemas SSI o versiones futuras de estándares.	Limitación de la adopción, necesidad de re-desarrollo.	Adherencia estricta a los estándares W3C (DID, VC, DIDComm V2). Participación en la comunidad de estándares. Pruebas de interoperabilidad continuas.

6. Pruebas y Validación

6.1. Estrategia de Pruebas

6.1.1. Pruebas Unitarias para Componentes Críticos

Se realizarán pruebas unitarias para componentes clave como el servicio de emisión de credenciales, verificando la lógica de negocio y la generación de *hashes* de consentimiento.

6.1.2. Pruebas de Integración para Flujos Clave

Se realizarán pruebas de integración para validar el flujo completo de registro de identidad y emisión de credenciales, incluyendo:

1. Registro del ciudadano.
2. Solicitud de afiliación.
3. Emisión de credencial.
4. Verificación de credencial.
5. Revocación de credencial.
6. Verificación de estado revocado.

6.2. Validación de Conformidad con Estándares

6.2.1. Conformidad con W3C DID

Validación técnica de conformidad para los DIDs, verificando:

1. Estructura DID (did:prism:<namespace>:<unique-id>).
2. Resolución DID y existencia del DID Document.
3. Presencia de métodos de autenticación.
4. Formato y tipo de claves (ej. Ed25519VerificationKey2018).

6.2.2. Conformidad con W3C Verifiable Credentials

Validación de conformidad para credenciales:

1. Verificación del contexto W3C Verifiable Credentials.
2. Presencia del tipo VerifiableCredential.
3. Presencia de credentialSubject.
4. Verificación del esquema político (PoliticalAffiliationCredential).
5. Estructura de credentialSubject y campos requeridos.