Universidad Europea VALENCIA

GRADO EN CIENCIA DE DATOS

Sistema de Trazabilidad Segura e Inviolable mediante Blockchain y Tecnología RFID para la Protección de Activos

Presentado por:
Dmitriy Kirichenko
Dirigido por:
Jesus Friginal Lopez

CURSO ACADÉMICO 2024-2025

Universidad Europea VALENCIA

Índice

Universidad Europea VALENCIA

Resumen

En un contexto global donde los bienes personales y comerciales se ven cada vez más afectados por delitos como el robo o la falsificación, surge la necesidad de desarrollar soluciones tecnológicas capaces de brindar protección efectiva. Este trabajo propone una solución integral denominada InLock, basada en la combinación de etiquetas RFID de tipo write-once embebidas en productos físicos, la tecnología NFC accesible desde dispositivos móviles comunes, y una infraestructura blockchain descentralizada implementada mediante una arquitectura DAG (Directed Acyclic Graph).

El sistema permite asociar cada producto físico a un identificador único e irreversible, almacenado y verificado mediante blockchain. Esto proporciona un registro inmutable y transparente que asegura la autenticidad, propiedad y trazabilidad completa del bien a lo largo de toda su vida útil. A través de una aplicación móvil multiplataforma desarrollada con Kotlin Multiplatform, cualquier usuario puede realizar operaciones esenciales como el escaneo físico del producto, verificar al instante su autenticidad, comprobar quién es su propietario actual, e incluso detectar si el bien ha sido denunciado como robado.

InLock posibilita también la transferencia segura de la propiedad mediante un proceso doble de verificación física (NFC) y digital (códigos QR), reforzando así la seguridad y la confianza entre usuarios. A nivel técnico, incorpora mecanismos avanzados de seguridad que incluyen autenticación robusta basada en Firebase (con JWT y gestión de roles), protección ante clonación o manipulación de las etiquetas RFID, y una lógica descentralizada para asegurar la coherencia y resiliencia de la información registrada en blockchain.

Además, la plataforma introduce un mecanismo de incentivos mediante la generación de tokens digitales vinculados a la posesión legítima y continua de activos físicos verificados. De esta manera, se promueve una economía digital segura, descentralizada y basada en la confianza.

Finalmente, InLock contribuye directamente a mejorar la eficiencia de las fuerzas de seguridad, facilitando la identificación precisa e inmediata de bienes recuperados, y ayudando a combatir el comercio ilegal de productos robados y falsificados, que representa aproximadamente el 3,3% del comercio mundial (OECD/EUIPO, 2019). El proyecto también se alinea con los Objetivos de Desarrollo Sostenible (ODS), especialmente con aquellos que buscan fortalecer la innovación tecnológica (ODS 9) y promover instituciones sólidas, justas y transparentes (ODS 16).

1. Introducción

1.1 Contexto y justificación del Proyecto

La protección de bienes personales y comerciales ha adquirido una relevancia crítica en los últimos años, impulsada por el aumento sostenido de los delitos contra la propiedad.

Más allá del perjuicio económico evidente, los robos exponen a las personas a situaciones de riesgo físico, incluyendo agresiones y daños que afectan gravemente su integridad y bienestar. Este impacto social, a menudo subestimado, evidencia la urgencia de sistemas que no solo resguarden los activos, sino también la seguridad y dignidad de sus legítimos propietarios.

En paralelo, la falsificación de productos continúa su expansión global, afectando tanto a marcas como a consumidores. Según la OCDE y la EUIPO (2019), este fenómeno representa cerca del 3,3 % del comercio mundial. Los productos falsos no solo generan pérdidas económicas, sino que además comprometen la salud, la seguridad y los derechos del consumidor. Esta problemática se ve agravada por la ausencia de mecanismos tecnológicos accesibles para verificar la autenticidad y trazabilidad de los bienes físicos.

Este proyecto plantea una solución tecnológica innovadora que permite vincular activos reales a registros digitales inmutables, mediante el uso combinado de etiquetas RFID de escritura única - integradas discretamente en los productos - y una infraestructura blockchain basada en arquitectura DAG (Directed Acyclic Graph). Esta configuración posibilita la creación de un registro descentralizado, resistente a manipulaciones, que certifica de forma confiable la existencia, autenticidad y propiedad de cada objeto.

Además de contribuir a la seguridad ciudadana, el sistema favorece la labor de las fuerzas de seguridad, facilitando la identificación de bienes recuperados y agilizando los procedimientos judiciales asociados. De esta manera, no solo se protege al individuo, sino que también se fortalece la respuesta institucional frente a delitos patrimoniales.

Finalmente, este trabajo se alinea con los Objetivos de Desarrollo Sostenible (ODS) promovidos por Naciones Unidas, en particular el ODS 9, orientado al fomento de la innovación tecnológica, y el ODS 16, enfocado en la construcción de instituciones más justas, efectivas y transparentes (ONU, 2015).

1.2 Motivación y relevancia

El desarrollo de nuevas tecnologías aplicadas a la seguridad y trazabilidad de bienes físicos no solo responde a una necesidad social urgente, sino que también representa una oportunidad para avanzar en la consolidación de entornos urbanos más seguros e inteligentes. En contextos donde la posesión de un bien no puede demostrarse fácilmente, se generan conflictos legales, pérdida de confianza en los mecanismos institucionales y una mayor carga para los sistemas judiciales.

Por otra parte, el mercado global avanza hacia una digitalización cada vez más profunda. En ese marco, el sistema propuesto introduce una capa de confianza adicional sobre el mundo físico, posibilitando que los bienes materiales formen parte de ecosistemas digitales más amplios, como plataformas de comercio electrónico, entornos de trazabilidad logística o sistemas de certificación inteligente.

Uno de los elementos más motivadores del proyecto es su capacidad para empoderar al usuario final: no solo protegiendo su propiedad, sino otorgándole la capacidad de certificarla, transferirla y, eventualmente, obtener una retribución económica mediante un sistema de tokens digitales asociados a la posesión legítima de bienes. Este modelo de incentivo, aunque no constituye el foco central del proyecto, representa un factor diferencial que podría favorecer la adopción masiva de tecnologías de autenticación descentralizadas.

Además, se abren líneas de innovación futura en aspectos como la integración con tecnologías IoT, la interoperabilidad con servicios públicos de verificación (como policía o aduanas), o la escalabilidad del modelo a través de redes federadas.

En síntesis, este proyecto no solo aborda un problema de seguridad, sino que explora una visión de futuro donde lo físico y lo digital convergen, generando valor económico, confianza institucional y herramientas para una sociedad más protegida y tecnológicamente empoderada.

1.2.1 Contribución a los Objetivos de Desarrollo Sostenible

El proyecto no solo atiende necesidades técnicas de trazabilidad y protección de activos, sino que también contribuye estratégicamente al cumplimiento de varios Objetivos de Desarrollo Sostenible (ODS) definidos en la Agenda 2030 de las Naciones Unidas (ONU, 2015).

En orden de relevancia para este proyecto, se impacta positivamente en los siguientes ODS:

- ODS 16: Paz, justicia e instituciones sólidas - El fortalecimiento de los mecanismos de protección de activos, la prevención de robos y falsificaciones, y el apoyo a las fuerzas de seguridad mediante sistemas de trazabilidad inviolables, contribuyen a crear sociedades más justas, pacíficas y transparentes.

- ODS 9: Industria, innovación e infraestructura - La implementación de tecnologías como RFID write-once, blockchain DAG y comunicación móvil NFC impulsa la construcción de infraestructuras digitales resilientes, sostenibles e innovadoras, fundamentales para una economía basada en activos protegidos y confiables.

- ODS 12: Producción y consumo responsables - El sistema combate de manera directa la falsificación y facilita el acceso de los consumidores a productos genuinos, promoviendo prácticas de producción más éticas y un consumo consciente y responsable.

- ODS 8: Trabajo decente y crecimiento económico - Al crear nuevas oportunidades en industrias de certificación, tecnología de trazabilidad y gestión digital de activos, el proyecto contribuye al crecimiento económico sostenible y la generación de empleos de calidad en sectores emergentes.

Con estas acciones, el proyecto no solo innova en el ámbito técnico, sino que también promueve un impacto social positivo y coherente con los principios de sostenibilidad global.

Nota sobre el uso de herramientas de IA

Durante la elaboración del presente Trabajo Fin de Grado se ha hecho uso puntual de herramientas de inteligencia artificial generativa, concretamente ChatGPT (OpenAI) y Claude (Anthropic), con el objetivo de mejorar la redacción, la coherencia estilística, la corrección gramatical y la organización lógica del contenido. ChatGPT ha sido empleado exclusivamente como asistente de apoyo lingüístico y estructural, mientras que Claude se utilizó específicamente para asistir en tareas de depuración de código (debugging) y generación automatizada de estructuras de logging, actuando siempre como soporte técnico complementario.

En ningún caso estas herramientas han sustituido el trabajo intelectual, técnico o conceptual desarrollado por el autor. Todas las decisiones relacionadas con el diseño arquitectónico del sistema, la implementación técnica, la elección de tecnologías, la construcción de la lógica de negocio, el desarrollo del código fuente, así como la definición y validación del modelo de trazabilidad, han sido llevadas a cabo de forma íntegra y original por el autor.

El contenido del documento, las ideas que lo sustentan y la estructura final reflejan un proceso personal de investigación, desarrollo aplicado y resolución de problemas, guiado por los criterios del tutor académico y las directrices establecidas por la Universidad Europea de Valencia.

La inclusión de esta nota responde a un compromiso explícito con la transparencia y la ética académica, reconociendo el uso de herramientas emergentes como parte del ecosistema de apoyo, pero sin que ello afecte a la autoría, autonomía ni profundidad del trabajo desarrollado.

2. Objetivos

Diseñar y desarrollar un sistema multiplataforma de trazabilidad segura e inviolable para bienes físicos, mediante la integración de etiquetas RFID de tipo write-once, tecnología de lectura NFC desde dispositivos móviles, y una infraestructura de registro descentralizado basada en una blockchain con arquitectura DAG (Directed Acyclic Graph).

El sistema tiene como objetivo principal permitir el registro inmutable, la validación física, la verificación pública y la transferencia segura de activos físicos, garantizando en todo momento la autenticidad, integridad y propiedad legítima de cada bien. Se ha implementado una aplicación móvil multiplataforma para realizar estas operaciones, una API intermedia segura para gestionar las solicitudes, y un backend basado en DAG que almacena los eventos de forma estructurada y criptográficamente protegida.

Como parte del diseño, se plantea un modelo de incentivos tokenizado, mediante el cual los propietarios legítimos de activos podrán generar tokens en función de su participación y posesión verificable, sentando así las bases para una economía digital distribuida basada en bienes físicos reales.

La solución incluye mecanismos avanzados de seguridad como verificación cruzada entre blockchain y Firestore, validación física mediante lectura NFC, protección contra transferencias fraudulentas, autenticación Firebase por roles, y replicación distribuida de datos relevantes en dispositivos móviles. El sistema está diseñado para ser escalable, accesible y energéticamente eficiente, y se alinea con los principios de innovación segura, trazabilidad distribuida y apertura tecnológica.

Numero	Objetivo específico
1	Diseñar una arquitectura de identificación física robusta utilizando etiquetas RFID de tipo WORM (write-once, read-many), discretamente embebidas en los productos durante su fabricación, y compatibles con el estándar NFC para su lectura directa desde dispositivos móviles.
2	Implementar una aplicación móvil multiplataforma mediante Kotlin Multiplatform y Compose UI, que permita el escaneo de etiquetas NFC, visualización de datos, verificación de propiedad y gestión de transferencia de activos, con control de acceso por rol (usuario, fabricante, administrador).
3	Diseñar y desarrollar un sistema blockchain funcional basado en una estructura DAG (Directed Acyclic Graph), prototipado en Python para facilitar su validación inicial y depuración, pero con la previsión de ser reimplementado en Golang en versiones avanzadas, dada su mayor eficiencia en operaciones concurrentes y tolerancia a carga. Esta blockchain DAG registrará eventos como el alta, transferencia, staking y denuncias de robo, asegurando integridad, trazabilidad y escalabilidad sin minería intensiva.
4	Implementar una API REST ligera en Flask (Prototipo en Python) como puente de comunicación entre la aplicación cliente y el motor DAG, actuando como capa intermedia que reciba, procese y devuelva peticiones de registro, consulta, validación y sincronización, sin intervenir en la lógica interna de la blockchain, asegurando una arquitectura desacoplada y mantenible.
5	Diseñar e implementar un sistema de reporte de activos robados, de modo que la marca de "bien robado" quede registrada de forma permanente en la blockchain y sea públicamente visible a cualquier usuario que interactúe con el activo. Este mecanismo deberá integrarse en la app móvil y reflejarse en las respuestas del backend.

6	Diseñar un mecanismo de validación distribuida de transacciones que se ejecute en segundo plano desde el dispositivo móvil de los usuarios, sin requerir intervención directa ni consumo significativo de recursos. Esta participación descentralizada permitirá validar bloques en la red DAG a través de nodos lógicos autónomos embebidos en la propia red de clientes, fortaleciendo la robustez de la red sin recurrir a infraestructura externa ni minería intensiva.
7	Diseñar el uso de bases de datos locales SQLDelight integradas en cada dispositivo móvil, con el objetivo de almacenar fragmentos relevantes de la información blockchain (como registros de propiedad, historial resumido o sincronización pendiente). Esta arquitectura permitirá el acceso offline eficiente y, al mismo tiempo, actuará como mecanismo de seguridad estructural, ya que la replicación del registro entre miles o millones de dispositivos introduce un nivel de redundancia distribuida que hace estadísticamente inviable cualquier intento de manipulación coordinada o eliminación masiva de datos, reforzando así la resiliencia de la red frente a ataques o fallos catastróficos.
8	Implementar un sistema de incentivos mediante generación de tokens digitales ligados a la posesión legítima de bienes físicos certificados. Este sistema se basará en esquemas tipo Proof of Stake y participación continua, vinculando identidad, autenticidad y permanencia en la red.
9	Evaluar la escalabilidad, consistencia y tolerancia a fallos del sistema mediante pruebas técnicas controladas que simulen escenarios reales como transferencias concurrentes, desconexión y reconexión asincrónica, y validación cruzada entre blockchain y datos locales.
10	Documentar de forma exhaustiva toda la arquitectura técnica del sistema, incluyendo diagramas de flujo, interacciones API, diseño modular de clases, estructura de nodos DAG, y posibilidades de evolución futura como la integración con smart contracts, wallets criptográficas o servicios públicos de verificación.

3. Marco Teórico

3.1 Trazabilidad y autenticación de bienes físicos

La trazabilidad de activos consiste en el proceso de registrar, verificar y seguir el ciclo de vida de un bien desde su origen hasta su destino final, incluyendo los eventos de transferencia, modificación o pérdida. Este concepto, tradicionalmente asociado al ámbito logístico y sanitario, ha cobrado una nueva relevancia en contextos donde la protección de la propiedad y la autenticidad resultan críticas.

En la economía actual, cada vez más interconectada y vulnerable a la falsificación, el contar con mecanismos fiables de trazabilidad se ha convertido en un factor diferencial tanto para fabricantes como para consumidores. Según UNIDO (2021), más del 65 % de los robos de bienes de consumo de alta gama podrían haberse resuelto de forma más eficiente si existiesen registros verificables de la propiedad a lo largo del tiempo. La trazabilidad, además, permite establecer la legitimidad de un bien sin depender exclusivamente de documentación física o terceros.

3.2 Tecnologías de Identificación: RFID y NFC

La Identificación por Radiofrecuencia (RFID) representa una evolución respecto a métodos tradicionales como códigos de barras o etiquetas visibles. Un sistema RFID consta de una etiqueta pasiva o activa que almacena un identificador único y un lector que interpreta esta información a distancia, incluso sin línea visual directa. Esta capacidad permite integrar la identificación dentro de los objetos, haciéndola prácticamente invisible y resistente a manipulaciones.

Particularmente relevantes en este contexto son las etiquetas WORM (write-once, read-many), cuyo contenido no puede ser reescrito tras su programación inicial, lo que elimina la posibilidad de alteración fraudulenta. Según Allied Market Research (2022), el mercado global de RFID alcanzará los 35.600 millones de USD en 2031, impulsado por su aplicación creciente en sectores de seguridad, retail y trazabilidad industrial.

Complementariamente, el uso de la tecnología NFC (Near Field Communication) - un subconjunto de RFID compatible con la mayoría de los smartphones— permite que los usuarios finales accedan directamente a la información embebida en los objetos sin equipos especializados. Esto abre la puerta a sistemas de verificación distribuidos y accesibles, donde cualquier ciudadano pueda validar un bien con su propio teléfono móvil.

3.3 Infraestructuras Distribuidas: Blockchain

Blockchain es una tecnología de almacenamiento de información basada en registros distribuidos, seguros y resistentes a manipulaciones. Cada bloque en la cadena contiene datos, una marca de tiempo y una referencia criptográfica al bloque anterior, formando una secuencia cronológica inmutable. Esta propiedad de inmutabilidad es especialmente útil en contextos de trazabilidad, donde garantizar que un evento no pueda ser modificado o eliminado resulta fundamental (Nakamoto, 2008).

Si bien la cadena de bloques tradicional se basa en una estructura secuencial, este modelo presenta ciertas limitaciones técnicas. Entre ellas se encuentran los cuellos de botella en el procesamiento de transacciones, el consumo energético elevado - especialmente en esquemas de consenso como Proof of Work (PoW) - y la centralización de la validación en grandes pools de minería (McKinsey & Company, 2022).

Para aplicaciones donde la frecuencia de registro es elevada o la escalabilidad es prioritaria, resulta más adecuado adoptar modelos estructurales alternativos como los DAG (Directed Acyclic Graphs).

3.4 DAG como evolución técnica de la blockchain

A diferencia del modelo lineal tradicional, un DAG organiza los registros en una estructura en grafo acíclico dirigido, donde cada nuevo nodo hace referencia a uno o varios anteriores. Esta arquitectura elimina la necesidad de bloques globales y permite que múltiples transacciones se verifiquen de forma paralela, reduciendo la latencia y mejorando la eficiencia energética. En lugar de mineros, los propios nodos que se añaden contribuyen a la validación de los anteriores, lo que descentraliza la autoridad sin sacrificar seguridad.

Además, un DAG es más flexible para entornos con muchas interacciones simultáneas, como podría ocurrir en un sistema de trazabilidad con múltiples usuarios realizando escaneos, consultas o transferencias en paralelo.

3.5 Avances y desafíos en soluciones existentes

En los últimos años, han surgido múltiples iniciativas que buscan aplicar estas tecnologías a la trazabilidad de bienes. VeChain, por ejemplo, es una plataforma que combina blockchain con sensores RFID para certificar autenticidad en sectores como el lujo o la alimentación. Sin embargo, requiere hardware especializado y no está orientada a la participación del consumidor final (VeChain Foundation, 2022). Chronicled, por su parte, integra blockchain y dispositivos IoT para verificar medicamentos, aunque su enfoque se limita al entorno empresarial (Chronicled, 2021).

Otros modelos, como Pi Network, se basan en la validación ligera de identidades mediante smartphones y redes de confianza, pero sin anclaje físico ni aplicación directa a bienes reales (Pi Network, 2022). A pesar de su popularidad, carecen de mecanismos robustos de verificación material.

La mayoría de estos sistemas operan sobre arquitecturas de cadena lineal, lo que limita su escalabilidad técnica y los expone a posibles problemas de saturación. Además, en muchos casos, el control del sistema recae en actores centralizados, lo que va en contra del principio de descentralización participativa que debería guiar los sistemas de trazabilidad en la era digital (Deloitte, 2021).

3.6 Análisis comparativo de tecnologías

Para evaluar la solidez y viabilidad del sistema propuesto, es necesario comparar su desempeño frente a otras tecnologías comunes utilizadas actualmente para la trazabilidad y autenticación de activos. A continuación, se presenta una tabla (Tabla 1) comparativa entre distintos enfoques, considerando factores como seguridad, posibilidad de manipulación, accesibilidad, coste, escalabilidad y compatibilidad con dispositivos móviles.

Tecnología	Seguridad	Manipulación de datos	Accesibilidad	Coste	Escalabilidad	Adaptación móvil
Número de serie físico	Baja	Fácil de copiar	Alta	Muy bajo	Alta	Total
Código QR	Baja	Fácilmente falsificable o copiable	Alta	Muy bajo	Alta	Total (cámara)
Hologramas de autenticidad	Media	Moderadamente reproducibles	Media	Medio-alto	Baja	Nula
Verificación centralizada online	Media	Vulnerable a perdida de integridad y alteraciones. Fallos de servidor.	Alta	Medio	Media	Total
RFID tradicional	Media	Puede ser reprogramado o clonado si no está protegido	Moderada	Medio	Alta	Limitada (requiere hardware externo)
RFID WORM + Blockchain DAG (Propuesta)	Muy alta	Inalterable tras registro inicial	Alta (lectura vía NFC)	Medio	Muy alta (estructura DAG)	Total (NFC en móviles)

Tabla 1 – Comparación de technologias

Análisis detallado:

- Número de serie físico: Aunque es común y barato, no ofrece garantías de autenticidad. Puede ser fácilmente replicado o falsificado sin dejar trazabilidad digital.
- Código QR: Ampliamente adoptado por su bajo coste y facilidad de implementación, pero extremadamente vulnerable a la suplantación, clonación o sustitución maliciosa.
- Hologramas de autenticidad: Aunque visualmente difíciles de replicar, no permiten trazabilidad digital ni verificación autónoma, y su reproducción ha mejorado con el avance tecnológico.
- Verificación online centralizada: Depende de servidores únicos que pueden ser comprometidos. La falta de descentralización reduce su resiliencia.
- RFID tradicional: Aunque útil, su capacidad de reescritura lo hace vulnerable sin protección criptográfica adicional. Requiere además hardware especializado para su lectura.
- RFID WORM + Blockchain DAG (propuesta): Combina identificación única e inviolable con trazabilidad descentralizada y eficiencia estructural. El uso de etiquetas write-once elimina la posibilidad de reprogramación, mientras que el registro en DAG asegura la resistencia a manipulación o pérdida de datos. La compatibilidad NFC permite el uso de smartphones como lectores universales. Además, la arquitectura DAG ofrece ventajas significativas en términos de escalabilidad y eficiencia energética, reduciendo congestión y latencia (McKinsey & Company, 2022).

Esta solución híbrida permite no solo verificar el origen y la autenticidad de los bienes, sino también registrar, transferir y auditar su propiedad en tiempo real, sin requerir confianza en terceros o servidores centralizados. Supone, por tanto, un salto cualitativo respecto a tecnologías tradicionales tanto en seguridad como en transparencia y eficiencia.

4. Metodología

4.1 Descripción general de la arquitectura

El sistema planteado está diseñado para proteger activos físicos de alto valor mediante la integración de identificación automática, blockchain descentralizada y aplicaciones móviles de acceso universal. Su objetivo principal es reforzar la seguridad de los bienes frente a robos y falsificaciones, facilitando al mismo tiempo su trazabilidad y la validación pública de su propiedad.

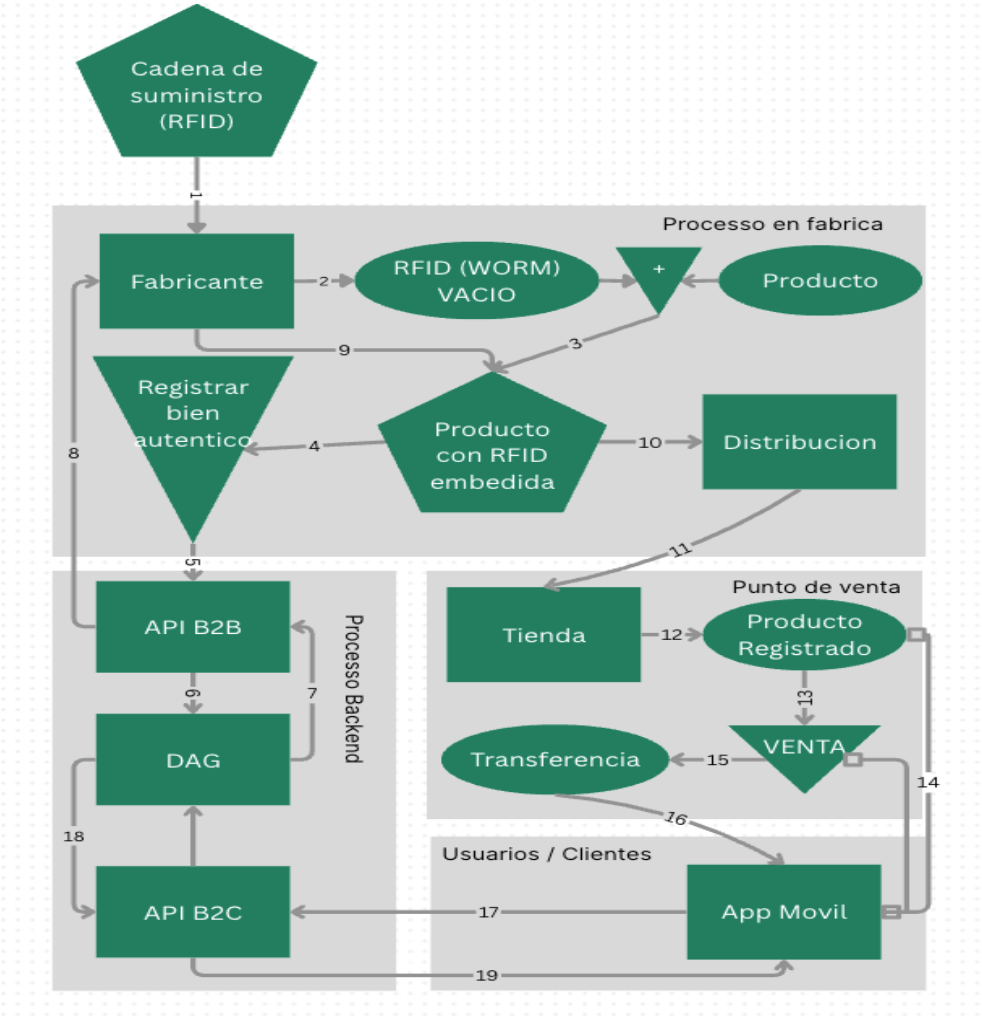


Imagen 1 – Flujo básico de arquitectura

Arriba, se presenta un diagrama (Imagen 1) simplificado que resume el flujo completo del sistema desde la etapa de fabricación hasta la interacción con el usuario final. Esta representación tiene carácter introductorio y servirá como marco de referencia para los apartados posteriores donde se explicarán los procesos con mayor nivel de detalle.

Descripción paso a paso del flujo principal:

1. Cadena de suministro (RFID): se obtiene el lote de etiquetas RFID WORM (Write Once Read Many), listas para ser utilizadas.
2. Fabricante: recibe las etiquetas vacías.
3. Proceso en fábrica: el fabricante embebe la etiqueta RFID dentro del producto físico.
4. Producto con RFID: el bien ya contiene una identidad única integrada en su estructura.
5. Registro de bien auténtico: el fabricante inicia el proceso de registro digital desde la aplicación móvil o interfaz administrativa.
6. API B2B: la solicitud es enviada al backend mediante una API destinada a fabricantes (Business to Business).
7. Blockchain DAG: la API procesa el evento y registra el nodo correspondiente en la red DAG, con los datos del activo, su creador y la marca temporal.
8. Confirmación del registro: el sistema responde al fabricante con la validación del registro.
9. Producto validado: el producto queda marcado como “auténtico” y listo para salir al mercado.
10. Distribución: se envía el bien a los canales de distribución.
11. Punto de venta: el producto llega a tiendas o minoristas.
12. Producto registrado: permanece listo para ser adquirido por un nuevo propietario.
13. Venta: cuando se produce la venta, se inicia una solicitud de transferencia de propiedad.
14. Validación e inicio de transferencia: el vendedor y el comprador validan el cambio mediante la app y el escaneo del RFID. (El comprador puede validar que compra el producto autentico.)
15. Nueva propiedad: la transacción es registrada como “transfer” en la DAG.
16. Usuario final: el nuevo propietario ya figura en el sistema como titular del bien.
17. App móvil: permite a usuarios escanear, verificar, denunciar o transferir activos desde el teléfono.
18. API B2C: las acciones de los usuarios se gestionan a través de una API orientada al cliente (Business to Consumer).
19. Interacción bidireccional: la app móvil y la red DAG se mantienen sincronizadas mediante operaciones seguras y auditables.

4.1.1 Bien físico con etiqueta RFID write-once

Cada bien físico será dotado durante su proceso de fabricación con una etiqueta RFID de alta frecuencia (HF, 13,56 MHz), seleccionada específicamente por su compatibilidad con la tecnología NFC presente en la mayoría de los teléfonos móviles actuales. Estas etiquetas serán de tipo WORM (Write Once, Read Many), es decir, diseñadas para ser programadas una única vez de forma irreversible, lo que impide su sobrescritura o modificación posterior.

Esta característica garantiza la inmutabilidad del identificador vinculado al producto, lo cual es esencial para prevenir suplantaciones o falsificaciones. El contenido grabado en la etiqueta incluirá:

- Un identificador único (UID) generado de forma criptográficamente segura.
- Una referencia hash del fabricante, que permite validar el origen del producto sin exponer datos sensibles.
- Opcionalmente, un timestamp de inicialización, útil para verificar la antigüedad del activo.

La etiqueta será embebida físicamente en una zona interna del producto, no visible externamente, y adherida de forma estructural para que cualquier intento de extracción o alteración implique la destrucción parcial del bien. Esta decisión de diseño incrementa de forma considerable la seguridad física del identificador, dificultando su clonación o reutilización no autorizada.

Gracias a su formato pasivo y de bajo consumo, estas etiquetas no requieren batería, tienen una vida útil prolongada, y pueden ser leídas con un simple acercamiento de un smartphone, sin necesidad de equipamiento especializado. Este enfoque permite que cualquier usuario pueda verificar la legitimidad del producto desde su propio dispositivo, sin intermediarios.

4.1.2 App móvil de escaneo y gestión de activos

El núcleo de interacción del usuario con el sistema estará basado en una aplicación móvil multiplataforma, desarrollada en Kotlin con Compose Multiplatform, compatible tanto con dispositivos Android como iOS. Esta app funcionará como herramienta universal para verificar, gestionar y proteger la propiedad de activos físicos registrados en la red blockchain.

Entre sus funcionalidades principales se encuentran:

- Lectura de etiquetas RFID mediante NFC: utilizando el chip NFC integrado en la mayoría de los smartphones actuales, la app puede leer en tiempo real el identificador único embebido en el producto.
- Consulta de información pública del bien: al escanear el producto, el usuario podrá acceder a datos verificados como el historial de propiedad, la marca de fabricación y el estado actual del activo (válido, robado, transferido, etc.).
- Verificación de autenticidad y propiedad: la app realiza una validación directa con la red blockchain para confirmar si el bien corresponde efectivamente al usuario o si presenta alguna irregularidad.
- Gestión de transferencias de propiedad: en caso de venta o cesión, tanto el antiguo como el nuevo propietario deberán validar el proceso mediante la app. Este flujo genera una nueva transacción firmada digitalmente que se registra en la arquitectura DAG.
- Participación en el staking de bienes: los usuarios que posean activos legítimos podrán participar en un sistema de incentivos mediante staking, generando tokens digitales como recompensa por mantener bienes registrados y activos en la red.

Además, la aplicación está diseñada para ofrecer un modo de verificación de terceros, permitiendo que compradores, autoridades o agentes externos puedan comprobar rápidamente la validez y procedencia del producto simplemente escaneando su etiqueta RFID.

Esta solución combina usabilidad, accesibilidad y seguridad, permitiendo que cualquier persona con un teléfono móvil moderno pueda interactuar con el sistema sin conocimientos técnicos avanzados ni necesidad de dispositivos adicionales.

4.1.3 Backend API segura

El sistema contará con una API REST segura desarrollada utilizando el framework Ktor, una tecnología ligera y asincrónica basada en Kotlin, ideal para aplicaciones modernas y escalables. Esta API estará contenida en un entorno Docker, lo cual permite un despliegue eficiente, reproducible y seguro tanto en servidores dedicados como en plataformas de nube como Firebase, AWS o Google Cloud.

La función principal de esta API será servir de puente entre la aplicación móvil del usuario y el núcleo de la infraestructura blockchain, actuando como capa lógica de control, validación y orquestación de eventos del sistema.

Funciones clave del backend:

- Validación de autenticidad de solicitudes: toda interacción entre el cliente y el sistema será evaluada mediante verificaciones criptográficas y autenticación de sesión, evitando operaciones no autorizadas o maliciosas.
- Control de acceso y permisos por roles: el backend aplicará un modelo RBAC (Role-Based Access Control), diferenciando entre usuarios finales, fabricantes y administradores. Este modelo asegura que cada actor tenga acceso solo a las funciones correspondientes a su rol.
- Gestión de staking, generación de tokens y transferencias de propiedad: los flujos críticos del sistema, como registrar una propiedad, realizar un staking o cambiar de propietario, serán coordinados por esta API. El backend verificará las condiciones necesarias (por ejemplo, que el usuario sea el propietario actual del activo), antes de comunicar la operación a la red DAG.
- Protección contra ataques y manipulación de datos: se implementarán estrategias para mitigar riesgos comunes como inyección de datos, ataques de repetición (replay attacks), suplantación de identidad (spoofing) y denegación de servicio (DoS). Esto se logrará mediante validaciones, filtros, y políticas de timeout y rate-limiting.
- Conexión con Firebase para autenticación y metadatos: se utilizará Firebase Authentication como proveedor de identidad, permitiendo gestionar sesiones de usuarios con seguridad y escalabilidad. Asimismo, Firestore será empleado como almacenamiento auxiliar para metadatos no críticos, como perfiles, configuración del usuario o índices rápidos de búsqueda, mientras que los datos sensibles permanecerán protegidos dentro del ecosistema blockchain.

Este backend está diseñado para funcionar como una capa de seguridad robusta, pero también como motor de lógica de negocio que garantice integridad, coherencia y fiabilidad en las operaciones del sistema. Su diseño desacoplado permite futuras ampliaciones, como la integración de nuevas tecnologías de autenticación, microservicios adicionales o servicios analíticos en tiempo real.

4.1.4 Blockchain DAG para trazabilidad y staking

El núcleo de integridad del sistema estará sostenido por una blockchain no lineal basada en arquitectura DAG (Directed Acyclic Graph). A diferencia de las cadenas de bloques tradicionales, que se organizan en secuencia (bloque tras bloque), el modelo DAG permite que las transacciones se entrelacen de forma distribuida y asincrónica, sin necesidad de esperar a que se "mine" un bloque completo.

Este sistema está implementado como prototipo en lenguaje Python, pero su diseño final está proyectado para desarrollarse en Golang, una tecnología más eficiente para el manejo de concurrencia, redes y procesamiento intensivo a nivel de sistema.

Cada nodo en el DAG representa un evento validado en la vida del activo (por ejemplo, su creación, transferencia o staking), y se enlaza a uno o dos nodos anteriores mediante referencias criptográficas. Esto garantiza que toda modificación esté respaldada por la historia previa, creando una estructura inalterable y auditada.

Cada nodo de registro incluye:

- Identificador RFID único, asociado físicamente al producto.
- Metadatos descriptivos del bien, tales como:
    - Fabricante (en forma de hash o firma digital).
    - Modelo.
    - Peso.
    - Color.
    - Material.
    - Fecha de fabricación.
- Historial de eventos, incluyendo:
    - Registro inicial de la propiedad (realizado por el fabricante).
    - Transferencias posteriores entre usuarios.
    - Activaciones y desactivaciones de staking por parte del poseedor legítimo.
    - Cambios de estado, como denuncias de robo o recuperación del activo.

Características técnicas destacadas del DAG:

- Alta escalabilidad: el DAG permite registrar múltiples eventos simultáneamente sin cuellos de botella, ya que cada nuevo nodo solo necesita validar una o dos transacciones previas (referencias).
- Bajo consumo energético: al no depender de minería intensiva como en el modelo Proof of Work (PoW), el DAG reduce de forma drástica la necesidad de cómputo, favoreciendo su uso incluso en dispositivos móviles.
- Validación distribuida (PoS/PoA): el sistema implementa un modelo híbrido de validación basado en:
    - Prueba de Participación (PoS): donde los usuarios con más bienes registrados tienen más peso al validar nuevos eventos.
    - Prueba de Autoridad (PoA): donde ciertas entidades confiables (como fabricantes verificados) tienen capacidad para validar eventos clave.
- Ledger descentralizado y sincronizado: la información de la DAG se replica parcialmente en miles de nodos ligeros (usuarios) mediante una base de datos local embebida (por ejemplo, SQLite/SQLDelight), actuando como libro mayor distribuido (distributed ledger). Esta descentralización hace que alterar la historia de propiedad sea prácticamente imposible, ya que requeriría modificar simultáneamente una enorme cantidad de copias en diferentes dispositivos.
- Integridad asegurada: cada nodo contiene una firma digital y un hash criptográfico, de forma que cualquier modificación maliciosa rompe automáticamente la estructura. La integridad completa del sistema puede verificarse mediante algoritmos internos de chequeo que comparan hashes actuales con los esperados.

Este enfoque no solo permite mantener un historial verificable y robusto de cada activo, sino que también democratiza el almacenamiento y la validación, repartiendo la carga entre todos los participantes del sistema, lo que lo hace resistente a fallos, manipulaciones y censura.

4.1.5 Diseño de nodos validadores y participación distribuida

La robustez y descentralización del sistema propuesto se ve reforzada por la incorporación de nodos validadores autónomos y de bajo consumo, los cuales serán ejecutados de forma distribuida desde los dispositivos móviles de los usuarios. Este enfoque permite democratizar la validación de transacciones sin necesidad de infraestructura pesada, eliminando intermediarios y favoreciendo la escalabilidad.

Tipología de validadores:

Se propone un modelo mixto de validación descentralizada compuesto por:

- PoS (Proof of Stake): cualquier usuario con un número significativo de activos legítimos registrados podrá convertirse en validador, contribuyendo a la red al permitir nuevas transacciones. El peso de validación estará vinculado al número y valor de los bienes bajo su posesión.
- PoA (Proof of Authority): fabricantes verificados y entidades confiables actuarán como validadores de alto nivel para eventos críticos (como el registro inicial de un producto), garantizando la legitimidad de los datos desde el origen.

Implementación técnica:

- Cada dispositivo móvil funcionará como un nodo ligero, conteniendo una réplica parcial de la DAG a través de una base de datos local persistente (como SQLDelight), donde se almacena el subconjunto de datos relevantes al usuario (sus activos, historial de propiedad, nodos cercanos, etc.).
- Estos nodos móviles operarán en segundo plano, validando nuevos eventos relacionados con su contexto. Por ejemplo, si un usuario posee cinco activos y otro nodo inicia una transferencia con uno de ellos como referencia, el dispositivo ejecutará una validación rápida antes de aceptar o rechazar el nuevo nodo.
- Las operaciones se diseñarán para tener un consumo energético mínimo, funcionando solo bajo eventos relevantes y mediante uso de corutinasKotlin con Dispatchers.IO y withTimeout, garantizando eficiencia sin impacto notable sobre la batería ni la experiencia del usuario.

Beneficios de la participación distribuida:

- Reducción de dependencia de servidores centrales: al delegar validaciones a los propios usuarios, el sistema es más resistente a caídas o censura.
- Mayor seguridad: la replicación parcial de la DAG hace que cualquier intento de alterar los datos requiera manipular simultáneamente miles de nodos, lo cual es estadísticamente inviable.
- Participación incentivada: los validadores recibirán recompensas en forma de tokens digitales por su contribución a la red, generando un ecosistema autosostenido, donde la tenencia legítima de bienes no solo otorga derechos, sino también beneficios económicos.

Este modelo innovador descentraliza la verificación sin sacrificar seguridad ni eficiencia, sentando las bases de una red sólida, participativa y resistente a fallos o ataques.

4.1.6 Seguridad y resistencia

El sistema propuesto ha sido concebido desde su origen con un enfoque defensivo y resiliente, integrando múltiples capas de seguridad tanto en el plano físico como en el digital. A continuación, se describen los principales mecanismos implementados o previstos en el diseño de la arquitectura:

Inmutabilidad estructural mediante DAG blockchain

La base de datos de eventos está estructurada como un grafo acíclico dirigido (DAG), donde cada nodo representa un evento firmado criptográficamente. Este enfoque garantiza que una vez registrado, ningún dato pueda ser modificado sin invalidar toda la cadena de referencias posteriores, proporcionando una inmutabilidad nativa sin necesidad de minería intensiva. El sistema también incluye verificación periódica de integridad, comprobando hashes internos y referencias cruzadas.

Identificación robusta con RFID de tipo write-once

Los bienes físicos estarán equipados con etiquetas RFID write-once, programadas únicamente en su origen por el fabricante y posteriormente bloqueadas a nivel físico. Esto impide la sobrescritura o clonación posterior del ID. Además, la firma digital de la fabricante asociada a cada producto actúa como prueba de origen, y es validada contra la red blockchain durante cada operación.

Autenticación y control de acceso con Firebase

Todas las operaciones que requieran permisos elevados - como transferencias, staking o registros de nuevos activos - estarán protegidas mediante un sistema de autenticación Firebase Authentication, que ofrece protección contra accesos no autorizados mediante:

- Gestión de sesiones seguras
- Tokens JWT firmados y renovables
- Inicio de sesión con correo/contraseña o métodos anónimos temporales
- Verificación de identidad por roles (usuario, fabricante, administrador)

Además, las reglas de acceso a datos en Firestore refuerzan este modelo, asegurando que un usuario sólo pueda acceder a los activos bajo su responsabilidad.

Validación cruzada y protección frente a clonado de etiquetas

Durante operaciones como transferencias de propiedad o staking, la aplicación realiza una lectura NFC del tag RFIDfísico en tiempo real. Esta lectura se compara contra los registros en blockchain:

- Si hay coincidencia, se permite continuar la operación.
- Si se detecta un intento de usar un ID no coincidente o repetido, la operación se bloquea y puede activarse un protocolo antifraude.

Esto proporciona una protección eficaz contra el clonado de etiquetas, reforzada por el hecho de que el ID físico está embebido y no es visible.

Resistencia ante ataques de red y denegación de servicio (DoS)

El backend (desarrollado en Ktor y contenido en Docker) implementa múltiples medidas de protección:

- Límites de tasa (rate limiting) para prevenir abuso de endpoints.
- Timeouts inteligentes en operaciones críticas.
- Manejo exhaustivo de excepciones y errores controlados.
- Logs seguros y trazabilidad para detectar actividades anómalas.

Asimismo, las conexiones entre el cliente, la API y otros servicios están cifradas mediante protocolos TLS/SSL, garantizando la confidencialidad e integridad de los datos en tránsito.

Resiliencia por replicación distribuida

La arquitectura aprovecha la replicación parcial del DAG en bases de datos locales (SQLDelight) en dispositivos móviles. Esto permite mantener copias fragmentadas del historial de propiedad, distribuidas entre miles de usuarios. Dicha replicación no solo mejora la disponibilidad, sino que también impide que un ataque dirigido pueda alterar o eliminar el historial de activos, ya que la red se reconstruye automáticamente a partir de nodos legítimos.

En conjunto, estas medidas establecen un ecosistema altamente resistente a manipulaciones, fraudes, pérdidas de datos y ataques comunes, reforzando la confiabilidad del sistema desde su concepción técnica.

4.2 Componentes principales del sistema

4.2.1 Aplicación móvil multiplataforma

La aplicación móvil es uno de los componentes centrales del sistema, ya que representa la interfaz directa entre el usuario final y la infraestructura tecnológica de protección y trazabilidad de activos. Se desarrollará utilizando Kotlin Multiplatform con el fin de ofrecer una única base de código que funcione en dispositivos Android e iOS, optimizando los tiempos de desarrollo, mantenimiento y evolución futura del sistema.

Arquitectura de la app

El patrón de arquitectura adoptado será MVI (Model-View-Intent), que ofrece un flujo de datos unidireccional, ideal para aplicaciones donde la gestión del estado de la UI y la previsibilidad de las acciones del usuario son críticos.

La estructura básica será:

- Model: Estado inmutable de la pantalla en cada momento.
- View: Representación visual del estado. Basada en Jetpack Compose para Android y Compose Multiplatformpara Desktop/iOS.
- Intent: Acciones del usuario o eventos externos que modifican el estado.

El uso de MVI garantiza que cada acción del usuario o evento externo lleve a una transición clara y controlada del estado, reduciendo los errores y facilitando el testing de la lógica de negocio.

Tecnologías principales usadas en la App:

- Kotlin Multiplatform Mobile (KMM): Para compartir la lógica de negocio entre Android e iOS.
- Compose Multiplatform: Para construir interfaces reactivas, modernas y nativas en ambos sistemas operativos.
- Koin: Para la inyección de dependencias ligera, fácil de integrar en Kotlin Multiplatform, permitiendo modularizar componentes como repositorios, viewmodels y casos de uso.
- Ktor Client: Para la comunicación segura con el Backend API, permitiendo realizar peticiones HTTP/HTTPS de forma eficiente y multiplataforma.
- SQLDelight: Para persistencia local de datos sensibles en dispositivos, con soporte de encriptación para proteger la información como identificadores de activos o claves privadas.
- Multiplatform NFC Libraries: (como Kotlin-NFC) para acceder a las funcionalidades de escaneo NFC en móviles compatibles, especialmente para Android; en iOS se trabajará con el framework Core NFC.
- Firebase Authentication (opcional para login): Para gestionar la identidad de los usuarios en el caso de que se quiera permitir recuperación de cuentas o validaciones externas.

Flujo básico en la App

1. Escaneo NFC: El usuario acerca su móvil al bien, la app detecta el RFID y extrae el identificador único.
2. Consulta a la blockchain: Se solicita al backend validar la existencia y los metadatos del ID escaneado.
3. Visualización de información: Se muestra al usuario el estado del bien (registrado, auténtico, robado, etc.)
4. Staking/Mining: Si el usuario posee bienes registrados, puede activar el staking para minar tokens digitales.
5. Gestión de activos: El usuario puede realizar transferencias de propiedad de activos a nuevos usuarios mediante firma digital y validación de la blockchain.

Justificación de elección tecnológica

- Kotlin Multiplatform + Compose reduce costes de desarrollo y mantenimiento al compartir casi el 70 % del código entre plataformas.
- MVI asegura estados consistentes, ideales para una app que depende de verificaciones de activos y requiere alta robustez en la UI.
- Ktor Client + Koin permiten una arquitectura limpia, desacoplada y escalable.
- SQLDelight proporciona una capa segura y multiplataforma para almacenar localmente los activos del usuario.
- Acceso NFC nativo en móviles modernos asegura facilidad de uso sin necesidad de hardware adicional.

4.2.2 Dispositivo físico: Etiquetas RFID write-once

En el núcleo físico de la solución se encuentran las etiquetas RFID de alta frecuencia (HF), diseñadas para proporcionar una identidad única e inmutable a cada bien protegido. La correcta selección y configuración de estas etiquetas es crítica para garantizar tanto la seguridad como la facilidad de acceso del sistema.

Características técnicas de las etiquetas RFID

Tipo de etiqueta: RFID HF (High Frequency) de 13,56 MHz, operando en la banda ISM (Industrial, Scientific and Medical).

- Norma internacional: Compatibilidad completa con ISO/IEC 14443 y ISO/IEC 15693 (estándares globales para transmisión de datos entre tarjetas sin contacto y lectores).
- Tipo de grabación: Write-once, read-many (WORM), es decir, la información se escribe una única vez en fábrica y posteriormente la memoria se bloquea para impedir cualquier sobrescritura o alteración futura.
- Memoria típica:
    - Espacio para almacenar el ID único (UID) del chip.

- Área segura para metadatos cifrados (por ejemplo, código interno del fabricante).
- Opcionalmente, hash de la firma digital para la verificación del fabricante.

Protocolos de comunicación

La comunicación entre el lector NFC del dispositivo móvil y la etiqueta RFID seguirá los protocolos estándar:

- NFC-A (ISO/IEC 14443 Type A): Para máxima compatibilidad con smartphones Android y iOS modernos.
- NFC Data Exchange Format (NDEF): Estructura de datos usada para encapsular y transferir información de forma segura y estandarizada.
- NTAG 424 DNA Secure NFC (opcional en niveles más avanzados): Para autenticación criptográfica de las lecturas y resistencia contra clonación.

Estas tecnologías garantizan que la interacción entre los teléfonos móviles y las etiquetas será rápida (menos de 100 ms de lectura), segura y ampliamente soportada.

Protección física y ubicación

Las etiquetas estarán embebidas directamente en los bienes durante su proceso de fabricación:

- Localizaciones estratégicas para maximizar la cobertura de lectura y minimizar el riesgo de manipulación física.
- Sellado mediante materiales resistentes al calor y al desgaste para proteger el chip durante toda la vida útil del producto.

En productos de alta gama (bolsos, relojes, joyería, etc.), se optará por integrar la etiqueta de manera discreta pero eficiente, utilizando encapsulados específicos como:

- Inlays flexibles en textiles o cuero.
- Etiquetas de vidrio o micro-transponders en estructuras metálicas o rígidas.

Seguridad de datos en la etiqueta

Una vez programada, la etiqueta quedará bloqueada utilizando mecanismos de protección como:

- Lock Bits: Protección física a nivel de bloque de memoria (proporcionada por el fabricante del chip).
- UID lock: Bloqueo del Identificador Único.
- Password Protection: En chips avanzados que soporten autenticación con clave de lectura/escritura.

Además, el identificador RFID nunca contendrá directamente datos sensibles visibles; cualquier información relevante (fabricante, modelo, peso) estará almacenada y asegurada en la blockchain, accesible solo tras verificación del UID del chip.

Justificación técnica

- La adopción del estándar ISO/IEC 14443 garantiza la interoperabilidad mundial con la mayoría de los dispositivos móviles.
- El modelo WORM (write-once) elimina riesgos asociados a reescritura, manipulación o corrupción del identificador.
- La integración de protocolos NFC-A y NDEF asegura compatibilidad inmediata con el hardware NFC existente.
- El uso opcional de chips seguros tipo NTAG 424 DNA permitiría escalar el nivel de seguridad ante escenarios de altísimo riesgo, como protección de bienes de lujo de muy alto valor.

4.2.3 Backend API segura

El backend API constituye el núcleo de operaciones lógicas y de comunicación segura del sistema, sirviendo de intermediario entre la aplicación móvil, la infraestructura blockchain DAG y las bases de datos auxiliares de autenticación y gestión de usuarios.

La solución se desarrollará utilizando Ktor Server, un framework asíncrono y no bloqueante para Kotlin, ideal para la creación de APIs ligeras, modulares y altamente escalables (JetBrains, 2023). El servidor estará contenido en entornos Dockerizados (Merkel, 2014), permitiendo la orquestación de servicios, el despliegue ágil y la portabilidad entre distintos entornos.

La estructura lógica del backend seguirá principios de Arquitectura Hexagonal (Ports and Adapters), mejorando el desacoplamiento entre dominios y adaptadores externos (Cockburn, 2005).

Funcionalidades principales del Backend

- Gestión de identidades: Se integrará Firebase Authentication para validar usuarios y fabricantes, utilizando tokens JWT firmados para autenticar sesiones de manera segura (Firebase, 2023).
- Interfaz con Blockchain DAG:
  El backend manejará:
    - El registro inicial de activos en la red blockchain.
    - Consultas de estados de activos basados en su ID RFID.
    - Transferencias de propiedad entre usuarios verificables en blockchain.
    - Activación y supervisión del staking de bienes certificados.
    - Emisión y control de tokens generados.

- Seguridad de comunicaciones: Todo tráfico de datos entre cliente y servidor estará protegido mediante TLS 1.3 (Rescorla, 2018), asegurando un cifrado de extremo a extremo.
- Control de acceso basado en roles (RBAC): El sistema implementará políticas de control de acceso a nivel de backend, diferenciando entre fabricantes, usuarios y autoridades verificadoras (Sandhu et al., 1996).
- Auditoría y trazabilidad: Todas las operaciones críticas serán registradas en un módulo de auditoría inmutable con sellado digital, para garantizar la trazabilidad y la transparencia operativa.
- Protección contra ataques:
  El backend incluirá:

    - Rate limiting para prevenir DoS.
    - Control de expiración de tokens JWT.
    - Integración opcional de un (WAF) para proteger puntos de entrada públicos.

Arquitectura de despliegue

El backend será modularizado y desplegado en contenedores Docker, inicialmente orquestados mediante Docker Compose en ambientes de prueba, con la posibilidad de migrar a Kubernetes para escenarios de alta disponibilidad y balanceo de carga en producción (Burns, 2016).

- Servicio público de API (HTTP/HTTPS seguro).
- Servicio interno de gestión blockchain (gRPC o REST privado).
- Módulo de autenticación y validación de tokens.

Justificación técnica

- Ktor Server permite construir APIs altamente optimizadas para Kotlin Multiplatform, integrándose perfectamente en entornos móviles y de backend modernos.
- Docker ofrece portabilidad, estandarización y aislamiento de entornos de ejecución críticos.
- Firebase Authentication proporciona una plataforma segura, confiable y escalable para la gestión de usuarios y credenciales.
- TLS 1.3 y JWT garantizan la protección de la información y autenticaciones seguras.
- RBAC y sistemas de auditoría fortalecen la gobernanza, el cumplimiento normativo y la resiliencia del sistema ante ataques o mal uso.

4.2.4 Infraestructura Blockchain basada en DAG

La infraestructura de registro inmutable del sistema estará construida sobre una blockchain basada en un modelo DAG (Directed Acyclic Graph), una estructura de datos no lineal que permite validar múltiples transacciones en paralelo sin necesidad de bloques secuenciales como en las blockchains tradicionales.

El modelo DAG elegido ofrece ventajas sustanciales en términos de escalabilidad, rapidez y eficiencia energética, factores críticos para soportar la trazabilidad de activos físicos en tiempo real y para permitir operaciones de staking continuas de forma descentralizada (Popov, 2018).

La blockchain DAG será diseñada y desarrollada utilizando el lenguaje de programación Golang, conocido por su rendimiento en aplicaciones de red intensivas y por su eficiencia en entornos de alta concurrencia (Donovan & Kernighan, 2015).

Estructura de la DAG

Cada nodo del DAG representará un evento único dentro del ecosistema:

- Registro de un activo: Cuando un bien se registra por primera vez en la plataforma, se crea un nodo que contiene:
    - El ID único del RFID.
    - Los metadatos del producto (fabricante, modelo, categoría, peso, material, fecha de fabricación, firma digital).
    - Una firma digital generada por el fabricante para validar la autenticidad del registro.
- Transferencia de propiedad: Cada cambio de dueño de un activo será un nuevo nodo en la DAG, que:
    - Referenciará al nodo anterior del activo.
    - Incluirá la firma digital del propietario actual y del nuevo propietario.
    - Actualizará el estado de posesión del bien en la red.
- Eventos de staking: El inicio o cierre de un periodo de staking para minería de tokens quedará registrado en un nodo:
    - Se validará que el usuario posee realmente el activo (relectura NFC + verificación de ID).
    - Se computará el peso del staking basado en la antigüedad y categoría del bien.
- Denuncias de robo o pérdida: También se permitirá registrar un nodo de reporte de robo, marcando el activo como "en disputa" en la red para bloquear su staking y futuras transferencias hasta su resolución.

Cada nodo será validado mediante un sistema de consenso híbrido:

- Proof of Authority (PoA):
    - Marcas certificadas (fabricantes) actuarán como nodos de alta autoridad para validar registros iniciales.
    - Validadores confiables podrán verificar transferencias de propiedad de alto valor.

- Proof of Stake (PoS):
    - Usuarios normales participarán como validadores basados en la cantidad de bienes certificados que posean.
    - Se incentivará la honestidad mediante recompensas simbólicas en tokens por validaciones correctas.

Mecanismos de validación y seguridad

- Referencias criptográficas: Cada nuevo nodo requerirá validar dos nodos anteriores (sistema de confirmaciones cruzadas) para garantizar la continuidad y evitar ataques de doble gasto.
- Firmas digitales:
    - Todas las operaciones críticas (registro, transferencia, staking) deberán estar firmadas criptográficamente mediante claves privadas basadas en algoritmos de curva elíptica (ECC).
- Protección contra spam: Se implementará una pequeña prueba de trabajo mínima (light PoW) en la creación de nodos para evitar el spam masivo de transacciones.

Generación de tokens y staking

Cada activo registrado podrá entrar en un proceso de staking controlado:

- El propietario deberá demostrar posesión periódica (escaneo NFC).
- El sistema calculará su "peso de staking" basado en:
    - El número de activos que posee.
    - La categoría del bien (productos de lujo pueden tener más peso que productos básicos).
    - El tiempo acumulado de posesión.

En función de su peso en la red, el usuario recibirá tokens generados de forma controlada siguiendo una política de inflación dinámica (similar al modelo de ajuste utilizado en redes como IOTA o Nano).

Los tokens podrán ser transferidos, intercambiados o vendidos en plataformas de intercambio compatibles.

Justificación técnica

- El uso de una blockchain DAG permite eliminar los cuellos de botella asociados a la minería secuencial, incrementando la velocidad de las validaciones sin sacrificar seguridad.
- El modelo híbrido PoA + PoS proporciona un balance entre confianza institucional y participación comunitaria.

- La estructura de nodos-evento facilita la trazabilidad completa del historial de cada bien sin pérdidas de integridad.
- La utilización de Golang permite construir una red de alta concurrencia, ideal para escenarios de múltiples operaciones simultáneas en tiempo real.

4.3 Flujo operativo del sistema

El flujo operativo describe el recorrido completo que sigue un bien físico desde su fabricación hasta su participación en la trazabilidad digital y la minería de tokens. Cada paso está diseñado para garantizar la seguridad de la información, la protección de activos y el control estricto de la propiedad legítima.

Fabricación e inserción de la etiqueta RFID

Durante el proceso de producción, el fabricante embebe una etiqueta RFID HF (13,56 MHz) de tipo write-once en el bien, programada con un identificador único. Simultáneamente, se generan los metadatos del producto (fabricante, modelo, material, peso, etc.), y estos se registran en la blockchain DAG junto con el ID, firmados digitalmente por el fabricante.

En este momento, la propiedad inicial del activo queda asignada al fabricante o a su representante de distribución (ej: una tienda oficial).

Registro inicial en la blockchain DAG

El fabricante o distribuidor autorizado realiza un registro de propiedad inicial:

- Se crea un nodo en la blockchain DAG que asocia el ID RFID con el fabricante o distribuidor.
- Los metadatos descriptivos del bien quedan ligados criptográficamente.

El activo aparece, así como "registrado" pero "no vendido" en la red.

Venta del bien al usuario final

Cuando un cliente compra el bien en una tienda autorizada:

- El vendedor escanea el RFID mediante su aplicación oficial de proveedor.
- Inicia el proceso de transferencia de propiedad dentro del sistema.
- Introduce los datos del comprador o vincula el activo a su cuenta de usuario.
- La transferencia queda pendiente de validación hasta que el pago se confirme.

Solo tras la confirmación del pago, el sistema:

- Cierra la transferencia en la blockchain DAG.
- Actualiza el nodo de propiedad: el activo pasa de estar bajo custodia del proveedor a ser propiedad del nuevo usuario.

- El comprador recibe la confirmación en su app móvil de que es el nuevo titular legítimo del bien.

Escaneo y validación por parte del usuario

El nuevo propietario puede:

- Escanear el bien con su app móvil mediante NFC.
- Consultar la información autenticada del producto.
- Ver su nombre o identificador como propietario actual en la blockchain DAG.

Este escaneo también sirve para validar que la transferencia fue registrada correctamente y no hubo manipulaciones.

Generación de tokens

El sistema calculará las recompensas en forma de tokens digitales basados en:

- El peso del staking.
- La participación del usuario.

Los tokens generados quedarán almacenados en la cuenta del usuario dentro de la aplicación.

Transferencias posteriores de propiedad

Si el usuario quiere vender o regalar su bien:

- Deberá iniciar una transferencia de propiedad desde su app.
- El nuevo comprador deberá aceptar la transferencia y vincular su cuenta.
- El Backend API validará la operación y registrará el cambio en la blockchain DAG.

5. Desarrollo e Implementación

5.1 Desarrollo de la aplicación móvil

5.1.1 Arquitectura general de la aplicación móvil

La aplicación móvil InLock ha sido diseñada siguiendo un enfoque multiplataforma basado en Kotlin Multiplatform Mobile (KMM), permitiendo compartir una base de código común entre Android e iOS, aunque en esta fase inicial el despliegue se ha centrado exclusivamente en Android. El desarrollo utiliza como capa visual Jetpack Compose Multiplatform, junto con un patrón de arquitectura funcional y altamente estructurado: Model-View-Intent (MVI).

El patrón MVI fue adoptado desde el inicio no solo por su capacidad para estructurar aplicaciones reactivas, sino también por las ventajas que ofrece a nivel de mantenibilidad, testabilidad y previsibilidad de estados. Bajo este modelo, la lógica de cada pantalla se organiza en torno a un ScreenModel (equivalente a un ViewModel en otras arquitecturas), que gestiona:

- un flujo de eventos (Intent)
- una función reductora que transforma esos eventos en un nuevo State
- y una UI que se suscribe únicamente al estado actualizado.

Este patrón se implementa combinando StateFlow, MutableStateFlow, rememberState, y LaunchedEffect para reaccionar de forma eficiente a los cambios.

Cada pantalla se encuentra dividida conceptualmente en:

- Screen.kt: interfaz visual declarativa con Jetpack Compose.
- ScreenModel.kt: orquestador de la lógica de negocio de esa pantalla.
- State.kt: modelo de estado inmutable que describe el estado actual de la UI.
- Intent.kt: conjunto de acciones que pueden ser ejecutadas desde la interfaz o desde eventos del sistema.

Consolidación del estado y cache de datos

Una de las decisiones arquitectónicas clave en InLock ha sido centralizar el control de sesión y el acceso a datos críticos mediante una capa de caché consolidada. Esta cache se gestiona mediante variables inyectadas de forma global (cachedUserData, cachedUserRole, entre otras), y se utiliza en múltiples niveles:

- UI Level: para evitar refrescos innecesarios al cambiar de pantalla.
- Auth Layer: para evitar llamadas redundantes a Firebase o Firestore.
- Security Layer: para prevenir decisiones inconsistentes basadas en estados no sincronizados.

Un ejemplo claro (Imagen 2) de esta consolidación es el sistema de validación de roles, en el que se prioriza el uso de datos almacenados en memoria antes de realizar consultas activas:

```
313
314         cachedUserRole?.let {
315             return it == UserRole.MANUFACTURER || it == UserRole.ADMIN
316         }
317
318         if (cachedUserData != null && cachedUserData?.id == userId) {
319             val role = cachedUserData?.role
320             return role == UserRole.MANUFACTURER || role == UserRole.ADMIN
321         }
322
323         return try {
324             val documentSnapshot = firestore
325                 .collection( collectionPath: "users")
326                 .document(userId)
327                 .get(com.google.firebase.firestore.Source.CACHE)
328                 .isSuccessful
329
330             if (documentSnapshot) {
331                 val cachedData = firestore
332                     .collection( collectionPath: "users")
333                     .document(userId)
334                     .get(com.google.firebase.firestore.Source.CACHE)
335                     .result
336
337                 if (cachedData != null && cachedData.exists()) {
338                     val userData = cachedData.toObject(UserData::class.java)
339                     val role = userData?.role
340                     cachedUserRole = role
341                     cachedUserData = userData
342                     role == UserRole.MANUFACTURER || role == UserRole.ADMIN
343                 } else {
344                     false
345                 }
346             } else {
347                 false
348             }
349         } catch (e: Exception) {
350             Logger.e(TAG,  message: "Exception checking if user is manufacturer", e)
351             false
```

Imagen 2 – Código de control de roles

Este enfoque reduce tanto la latencia como el consumo de red, a la vez que mantiene la coherencia entre capas, ya que los datos cacheados se invalidan correctamente tras un signOut() o cambios detectados desde la nube.

Gestión de dependencias

El proyecto emplea Koin como sistema de inyección de dependencias, lo que ha permitido desacoplar de forma limpia las distintas capas funcionales (interfaz, servicios, lógica de negocio y acceso a red). Las dependencias se agrupan en módulos como dataModule, networkModule y firebaseModule, cada uno de los cuales declara los singletons y bindings necesarios.

Un ejemplo extraído (Imagen 3) directamente del proyecto:

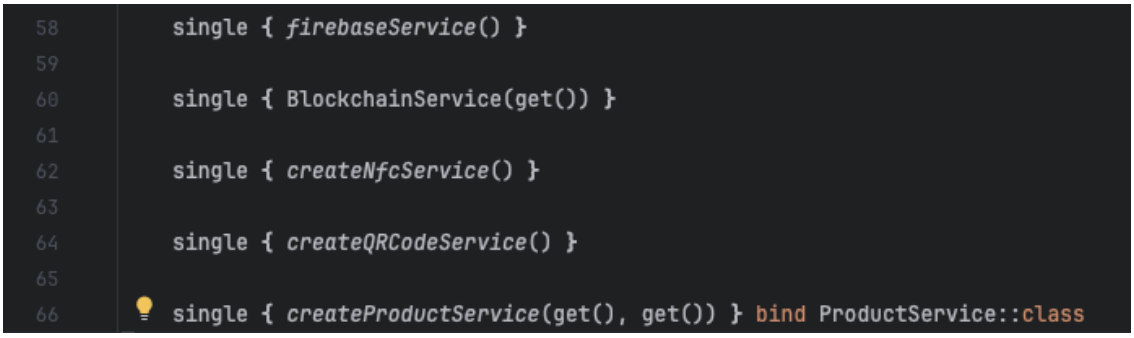


Imagen 3 – Inyecciones de servicios con Koin

Esta estructura permite reemplazar fácilmente servicios por versiones simuladas durante las pruebas o evoluciones del producto.

Manejadores de errores y estado

Cada pantalla implementa su propio StateHolder, y define dentro de su UiState campos que permiten detectar errores, activar loading states, mostrar mensajes temporales o validar flujos como transferencias y escaneos NFC. La función update{} de los StateFlow es utilizada como único medio de mutación del estado, garantizando inmutabilidad fuera del contexto controlado.

La gestión de errores se realiza en cascada, con múltiples niveles de validación:

- Captura de errores asíncronos mediante runCatching.
- Timeouts definidos por pantalla mediante withTimeoutOrNull.
- Fallbacks automáticos que permiten reconectar con fuentes secundarias en caso de fallo (por ejemplo: blockchain a Firestore).

En definitiva, la arquitectura de InLock ha sido diseñada no solo para cumplir con las funcionalidades propuestas, sino también para escalar en futuras versiones sin comprometer la claridad del código ni la seguridad del sistema. Esta combinación de un enfoque funcional (MVI), la consolidación de cache, la inyección estructurada de dependencias y una gestión robusta del estado hacen del núcleo de la aplicación una base sólida, tanto para su desarrollo futuro como para su mantenimiento a largo plazo.

5.1.2 Firebase Authentication

El sistema de autenticación en InLock se fundamenta en Firebase Authentication, una plataforma gestionada que proporciona servicios de login seguro, recuperación de contraseñas, gestión de sesiones y emisión de tokens de acceso. Se ha integrado en la aplicación móvil como principal puerta de entrada a todas las operaciones sensibles del sistema: desde registrar o transferir activos hasta verificar propiedad.

A nivel de protocolo, Firebase Auth se basa en OAuth 2.0 para el proceso de autenticación y autorización, y utiliza tokens JWT para gestionar las sesiones del usuario autenticado. Estos tokens, que son firmados digitalmente, incluyen información

como el UID del usuario, su correo electrónico, los claims personalizados y un timestamp de expiración.

En el proyecto InLock, toda la lógica de autenticación se encuentra encapsulada dentro de la clase FirebaseServiceAndroid.kt. El enfoque adoptado emplea corrutinas, Dispatchers.IO y el uso directo de suspendCoroutine{} para integrar APIs asincrónicas de Firebase con un flujo suspend, alineado con el modelo estructurado de Kotlin.

Un fragmento representativo del login por correo electrónico y contraseña es el siguiente (Imagen 4):

```kotlin
override suspend fun signIn(email: String, password: String): Result<String> = runCatching {
    withContext(Dispatchers.IO) {
        try {
            val userId = suspendCoroutine { continuation ->
                auth.signInWithEmailAndPassword(email, password)
                    .addOnSuccessListener { authResult ->
                        val uid = authResult.user?.uid ?: ""
                        continuation.resume(uid)
                    }
                    .addOnFailureListener { e ->
                        Logger.e(TAG, message: "Sign in failed", e)
                        continuation.resumeWithException(e)
                    }
            }

            try {
                val userData = getUserData(userId).getOrNull()
                if (userData != null) {
                    cachedUserData = userData
                    cachedUserRole = userData.role
                    Logger.d(TAG, message: "Cached user data: ${userData.displayName}, role: ${userData.role}")
                }
            } catch (e: Exception) {
                Logger.e(TAG, message: "Failed to cache user data", e)
            }

            return@withContext userId
        } catch (e: Exception) {
            Logger.e(TAG, message: "Exception during sign in", e)
            throw e
        }
    }
}
```

Imagen 4 – Código de login con autentificación por Firebase

Este fragmento representa no solo un mecanismo funcional de login, sino también un primer paso hacia un control de acceso más estructurado. Tras autenticar al usuario, el sistema intenta recuperar su perfil desde Firestore (vía getUserData) y guarda los datos en memoria para evitar solicitudes repetitivas. Esta consolidación en caché permite decisiones más rápidas en fases posteriores, como la validación de permisos para acceder a pantallas o realizar acciones restringidas.

Además, la app implementa (Imagen 5) la posibilidad de obtener y refrescar el ID Token directamente cuando es necesario, lo cual permite mantener la sesión activa en operaciones sensibles que puedan durar más de 60 minutos:

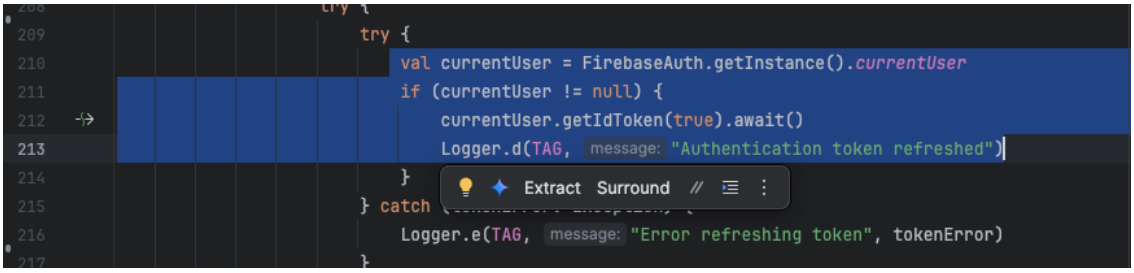


Imagen 5 – Código de control de token de autentificación

El parámetro true fuerza la renovación del token, asegurando que esté actualizado antes de iniciar operaciones críticas, como registrar activos en blockchain o ejecutar transferencias.

Desde el punto de vista de seguridad, la integración de Firebase Authentication en InLock cumple con varios principios fundamentales:

- Transporte seguro: toda la comunicación entre la app y los servidores de Firebase se realiza a través de conexiones cifradas TLS 1.3.
- Gestión de sesión segura: los tokens JWT emitidos por Firebase son válidos por períodos limitados (habitualmente una hora), y se renuevan automáticamente según la configuración del cliente.
- Protección contra suplantación: como los tokens están firmados por Google, no pueden ser falsificados ni manipulados por el cliente.
- Prevención de accesos indebidos: la verificación del rol se realiza tras el login, y se consolida tanto en caché como en Firestore, con múltiples capas redundantes (explicadas en la siguiente sección 5.1.3).

Adicionalmente, se implementa un mecanismo explícito de logout (Imagen 6) que borra todos los datos en memoria relacionados con la sesión actual:

Imagen 6 – Código de limpieza de cache a la hora de signout()

Este método asegura que ningún fragmento de información sensible se mantenga tras cerrar sesión, un principio básico pero esencial en cualquier sistema que pretenda ofrecer seguridad real.

Por último, se ofrece una funcionalidad de login anónimo (Imagen 7), lo que permite ofrecer un acceso limitado a funcionalidades no críticas:

```
121
122     override suspend fun signInAnonymously(): Result<String> = runCatching {
123         withContext(Dispatchers.IO) {
124             try {
125                 suspendCoroutine { continuation ->
126                     auth.signInAnonymously()
127                         .addOnSuccessListener { authResult ->
128                             val userId = authResult.user?.uid ?: ""
129                             continuation.resume(userId)
130                         }
131                         .addOnFailureListener { e ->
132                             Logger.e(TAG, message: "Anonymous sign in failed", e)
133                             continuation.resumeWithException(e)
134                         }
135                 }
136             } catch (e: Exception) {
137                 Logger.e(TAG, message: "Exception during anonymous sign in", e)
138                 throw e
139             }
140         }
141     }
```

Imagen 7 – Código de creación de usuario anonimo/invitado

Este login anónimo ofrece la posibilidad de explorar ciertas funcionalidades de la app sin comprometer la seguridad o exposición de datos. Naturalmente, sus permisos están restringidos mediante el sistema de control de acceso por roles (descritos en 5.1.3).

5.1.3 Manejo de roles y control de acceso

Uno de los pilares fundamentales de la arquitectura de InLock es la implementación de un sistema de control de acceso basado en roles (RBAC) En un entorno que maneja información sensible como la propiedad de activos físicos o la posibilidad de transferir bienes valiosos, resulta imprescindible establecer restricciones claras sobre qué operaciones puede realizar cada tipo de usuario.

El sistema de roles de InLock está conformado por tres perfiles principales, cada uno con responsabilidades y capacidades claramente diferenciadas:

- Usuarios regulares (USER): pueden poseer activos, transferirlos y consultar la autenticidad de bienes mediante escaneos.
- Fabricantes (MANUFACTURER): además de las capacidades de usuario, pueden crear plantillas de productos y registrar activos directamente sobre blockchain.

- Administradores (ADMIN): cuentan con privilegios globales, incluyendo la gestión de usuarios, auditorías y acciones de mantenimiento del sistema.

Esta jerarquía permite extender privilegios de forma lógica, ya que los fabricantes heredan los permisos de usuario, y los administradores los de ambos perfiles anteriores.

Verificación de roles en múltiples capas

El control de roles se implementa en varios niveles de la aplicación, lo cual genera redundancia deliberada para reforzar la seguridad:

A nivel de interfaz de usuario (UI)

Componentes específicos —como botones de administración, opciones de edición, o accesos a ciertas pantallas— se muestran u ocultan dinámicamente según el rol del usuario autenticado. Esta lógica se apoya en datos cacheados tras el login (cachedUserRole) y evita exponer opciones sensibles a perfiles no autorizados.

A nivel de navegación

La navegación condicional impide que ciertos flujos se activen si el rol actual no cumple los requisitos definidos. En pantallas protegidas se usa el composable RequireRole (Imagen 8), que encapsula esta lógica:

```
@Composable
fun RequireRole(
    requiredRole: UserRole,
    redirectTo: Screen = LoginScreen,
    content: @Composable () -> Unit
) {
    val navigator = LocalNavigator.currentOrThrow
    val userData by currentUserData.collectAsState()

    LaunchedEffect(Unit) {
        val hasPermission = hasRequiredRole(requiredRole)
        if (!hasPermission) {
            navigator.replace(redirectTo)
        }
    }

    if (userData != null) {
        val userRole = userData?.role
        when {
            userRole == requiredRole ||
                    (requiredRole == UserRole.USER) ||
                    (requiredRole == UserRole.MANUFACTURER && userRole == UserRole.ADMIN) -> {
                content()
            }
        }
    }
}
```

Imagen 8 – Código de control de roles a nivel de navegación

Este enfoque garantiza que incluso si el usuario navega manualmente hacia una pantalla protegida, no podrá interactuar con sus funcionalidades.

A nivel de servicio

Las clases de servicio como ProductService o BlockchainService incluyen validaciones internas antes de ejecutar operaciones sensibles. Por ejemplo, en el método registerProductOnBlockchain, se verifica explícitamente que el usuario autenticado sea efectivamente el fabricante del producto (Imagen 9):

```
283    override suspend fun registerProductOnBlockchain(product: Product): Result<String> = runCatching {
284        withContext(Dispatchers.IO) {
285            try {
286                val currentUserId = firebaseService.getCurrentUserId()
287                    ?: throw Exception("User not signed in")
288
289                if (product.manufacturer != currentUserId) {
290                    throw SecurityException("Only the manufacturer can register this product on the blockchain")
291                }
292
```

Imagen 9 – Control de roles a nivel de servicio

Estas comprobaciones están presentes en múltiples puntos del sistema, tanto para operaciones de escritura (registro, transferencia) como para lectura de información sensible.

A nivel de Firestore (reglas de seguridad)

Si bien el documento actual no incluye las reglas completas de seguridad de Firestore, la lógica de consulta implementada en el cliente sugiere que éstas están estructuradas para restringir el acceso por colección y documento, en función del rol del usuario autenticado. Métodos como getManufacturerProducts implican filtros del lado del servidor que validarían el ID del fabricante (Imagen 10):

```
152    override suspend fun getManufacturerProducts(manufacturerId: String): Result<List<Product>> = runCatching {
153        withContext(Dispatchers.IO) {
154            try {
155                withTimeout( timeMillis: 10000) {
156                    val snapshot = productsCollection
157                        .whereEqualTo( field: "manufacturer", manufacturerId)
158                        .whereEqualTo( field: "isTemplate",  value: false)
159                        .get()
160                        .await()
161
```

Imagen 10 – A nivel de datos sincronizados en Firestore

En producción, estas reglas serían complementadas con reglas de seguridad de Firebase que utilicen request.auth.token.role como claim para restringir el acceso.

Verificación redundante del rol

El método isManufacturer() implementado en FirebaseServiceAndroid.kt es un claro ejemplo (Imagen 11) de cómo se prioriza la robustez del sistema frente a la simplicidad:

```
class FirebaseServiceAndroid : FirebaseService {

    override fun isManufacturer(): Boolean {
        val userId = getCurrentUserId() ?: return false

        cachedUserRole?.let {
            return it == UserRole.MANUFACTURER || it == UserRole.ADMIN
        }

        if (cachedUserData != null && cachedUserData?.id == userId) {
            val role = cachedUserData?.role
            return role == UserRole.MANUFACTURER || role == UserRole.ADMIN
        }

        return try {
            val documentSnapshot = firestore
                .collection( collectionPath: "users")
                .document(userId)
                .get(com.google.firebase.firestore.Source.CACHE)
                .isSuccessful

            if (documentSnapshot) {
                val cachedData = firestore
                    .collection( collectionPath: "users")
                    .document(userId)
                    .get(com.google.firebase.firestore.Source.CACHE)
                    .result

                if (cachedData != null && cachedData.exists()) {
                    val userData = cachedData.toObject(UserData::class.java)
                    val role = userData?.role
                    cachedUserRole = role
                    cachedUserData = userData
                    role == UserRole.MANUFACTURER || role == UserRole.ADMIN
                } else {
                    false
                }
            } else {
                false
```

Imagen 11 – Verificación redundante

El método implementa tres niveles de verificación:

- Cache directa (cachedUserRole).
- Datos de usuario persistidos en caché (cachedUserData).
- Consulta a Firestore en modo offline (Source.CACHE), evitando así sobrecargar la red.

Este diseño busca minimizar los accesos a red y maximizar la velocidad, sin renunciar a la seguridad. El rol nunca se asume: siempre se valida de forma explícita, incluso cuando la red no está disponible.

Escalabilidad y evolución futura

El sistema RBAC actual es lo suficientemente flexible como para crecer si en futuras versiones se requiere:

- Asignar roles dinámicamente desde consola o backend.
- Introducir roles intermedios (por ejemplo, moderadores, verificadores).
- Aplicar restricciones por acción concreta y no solo por tipo de vista.

El uso de enums y funciones centralizadas facilita estas futuras adaptaciones sin romper la lógica existente.

En conjunto, el manejo de roles y el sistema de control de acceso implementado en InLock proporcionan una capa de seguridad crítica. No solo se trata de ocultar elementos visuales, sino de establecer un marco lógico coherente donde cada acción del usuario está condicionada por su perfil y validada por múltiples capas de protección. Esto reduce drásticamente el riesgo de abusos, errores de privilegio y posibles vulnerabilidades derivadas de flujos no controlados.

5.1.4 Transferencias, escaneos y validaciones de propiedad

Uno de los elementos más diferenciales de InLock frente a otras aplicaciones de gestión de activos digitales es su integración con elementos físicos mediante tecnologías de proximidad. A diferencia de soluciones puramente digitales, InLock vincula cada activo registrado con un objeto físico, haciendo uso de etiquetas NFC (Near Field Communication) como identificadores unívocos y no replicables en contexto.

Esto permite establecer un vínculo verificable entre el usuario y el bien que posee, tanto al momento de registrar el activo como durante las operaciones más críticas del sistema: las transferencias de propiedad.

Escaneo NFC como prueba de posesión física

Cuando un usuario desea realizar una transferencia de propiedad, el sistema no solo valida su identidad digital y rol, sino que le exige demostrar que está en posesión física real del objeto. Este paso de validación física se realiza mediante un escaneo NFC del producto.

La función encargada de manejar este proceso se encuentra implementada en la pantalla de transferencia (TransferProductScreenModel) y ejecuta una comparación directa entre el ID leído del chip NFC y el ID del producto seleccionado.

```
431     private fun verifyRfidForSelectedProduct(rfidId: String) {
432         screenModelScope.launch {
433             try {
434                 val selectedProduct = _uiState.value.selectedProduct
435                 Logger.d(TAG, message: "Verifying RFID: $rfidId against selected product: ${selectedProduct?.id}")
436
437                 if (selectedProduct == null) {
438                     _uiState.update {
439                         it.copy(
440                             error = "No product selected"
441                         )
442                     }
443                     return@launch
444                 }
445
446                 if (selectedProduct.id != rfidId) {
447                     Logger.d(TAG, message: "RFID mismatch: Expected ${selectedProduct.id}, got $rfidId")
448                     _uiState.update {
449                         it.copy(
450                             error = "The scanned tag doesn't match the selected asset"
451                         )
452                     }
453                     return@launch
454                 }
455
456                 Logger.d(TAG, message: "RFID verified successfully")
457                 _uiState.update {
458                     it.copy(
459                         isRfidVerified = true,
460                         error = ""
461                     )
462                 }
463             } catch (e: Exception) {
464                 Logger.e(TAG, message: "Error verifying RFID for selected product", e)
465                 _uiState.update {
466                     it.copy(
467                         error = e.message ?: "Error verifying scanned tag"
```

Imagen 12 – Código de comprobación de posesión física de bien

Este procedimiento, aunque aparentemente sencillo, impone una capa adicional de seguridad basada en una condición ineludible: nadie puede transferir lo que no puede escanear.

Validación cruzada: blockchain, UI y base de datos

Una vez verificado que el usuario posee físicamente el bien, el sistema procede a realizar la transferencia. Esta operación es compleja, ya que afecta a tres fuentes de verdad independientes:

- La blockchain, donde debe registrarse el cambio de titularidad de forma inmutable.
- Firestore, que contiene los metadatos estructurados del producto.
- La UI, que refleja el nuevo estado en tiempo real para el usuario.

Este es un ejemplo real del código que coordina esa sincronización múltiple (Imagen 13):

```
499     override suspend fun transferProductOwnership(productId: String, newOwnerId: String): Result<Unit> = runCatching {
500         withContext(Dispatchers.IO) {
501             try {
502                 val currentUserId = firebaseService.getCurrentUserId()
503                     ?: throw Exception("User not signed in")
504
505                 val product = getProduct(productId).getOrNull()
506                     ?: throw Exception("Product not found")
507
508                 if (product.currentOwner != currentUserId) {
509                     throw SecurityException("Only the current owner can transfer ownership")
510                 }
511
512                 if (product.isTemplate) {
513                     throw Exception("Templates cannot be transferred. Please instantiate a product from this template.")
514                 }
515
516                 if (product.isRegisteredOnBlockchain) {
517                     withTimeout( timeMillis: 15000) {
518                         blockchainService.transferAsset(
519                             assetId = productId,
520                             fromUserId = currentUserId,
521                             toUserId = newOwnerId
522                         ).getOrThrow()
523                     }
524                 }
525
526                 val updatedProduct = product.copy(currentOwner = newOwnerId)
527                 withTimeout( timeMillis: 5000) {
528                     productsCollection.document(productId).set(updatedProduct).await()
529                 }
530             } catch (e: Exception) {
531                 Logger.e(TAG,  message: "Error transferring product ownership: $productId to $newOwnerId", e)
532                 throw e
533             }
534         }
535     }
```

Imagen 13 – Comprobación de derechos para hacer transferencia con sincronización

Esta operación está protegida por múltiples validaciones:

- Confirmación del usuario autenticado.
- Verificación del propietario actual del bien.
- Comprobación de que no se trata de una plantilla (que no son transferibles).
- Confirmación de que el producto está registrado en blockchain.
- Actualización en Firestore tras la operación blockchain.

La lógica incluye además límites de tiempo (withTimeout) para evitar bloqueos en caso de fallos de red o cuellos de botella en servicios externos.

Escenarios de error y retroceso

En caso de que alguna parte de este proceso falle - por ejemplo, si el registro en blockchain no se completa - la operación no se considera válida, y el sistema revierte la lógica local para preservar la coherencia.

También se prevén escenarios donde la propiedad registrada en Firestore y la de la blockchain no coincidan. En ese caso, el sistema puede forzar una sincronización, adoptando como fuente de verdad prioritaria la información de blockchain (Imagen 14):

```
override suspend fun forceProductSync(assetId: String): Result<Unit> = runCatching {
    withContext(Dispatchers.IO) {
        try {
            Logger.d(TAG, message: "Starting force sync of product $assetId from blockchain")

            val currentUserId = firebaseService.getCurrentUserId()
                ?: throw Exception("User not signed in")

            val history = withTimeout( timeMillis: 10000) {
                blockchainService.getAssetHistory(assetId).getOrThrow()
            }

            if (history.isEmpty()) {
                throw Exception("Asset not found on blockchain")
            }

            val registrationRecord = history.find { it.action == "register" }
            val latestRecord = history.last()

            if (registrationRecord == null) {
                Logger.e(TAG, message: "No registration record found for asset $assetId")
                throw Exception("No registration record found for asset")
            }

            val nodeData = try {
                withTimeout( timeMillis: 8000) {
                    blockchainService.getAssetNodeData(assetId).getOrNull() ?: emptyMap()
                }
            } catch (e: Exception) {
                Logger.e(TAG, message: "Error getting node data, continuing with limited data", e)
                emptyMap<String, String>()
            }

            Logger.d(TAG, message: "Retrieved blockchain data for $assetId: ${nodeData.size} properties")


            val manufacturerId = registrationRecord.user_id
```

Imagen 14 – Sincronización de operaciones en Firebase en casos de errores

Consideraciones finales

La implementación de validación física mediante escaneo NFC, combinada con la doble confirmación en blockchain y base de datos, convierte a InLock en un sistema especialmente resistente frente a fraudes o falsificaciones de propiedad. El diseño impone una secuencia de pasos inseparables, que solo pueden ejecutarse correctamente si:

- El usuario está autenticado y tiene permisos.
- El bien está físicamente presente.
- El escaneo corresponde con el activo seleccionado.
- La blockchain acepta el nuevo registro.
- Firestore refleja la operación sin errores.

Cada uno de estos pasos se ejecuta de forma segura, con control de errores y timeouts para garantizar tanto la coherencia del sistema como la seguridad de los datos.

errores y de tiempo (withTimeout) para evitar inconsistencias en caso de fallo de red o latencia elevada.

5.1.5 Seguridad física: NFC, QR y validación de escaneos

El sistema InLock ha sido diseñado para garantizar una validación física real de los activos digitales registrados en blockchain. A diferencia de las plataformas puramente virtuales, aquí cada bien debe estar vinculado de forma inequívoca a su representación física a través de un identificador único y escaneable. Este enfoque se apoya principalmente en tecnología NFC, y se complementa con el uso de códigos QR, no para representar el activo, sino como mecanismo seguro de intercambio de identidad de usuario durante procesos de transferencia.

Identificación mediante NFC

Cada producto gestionado por InLock está destinado a incorporar una etiqueta NFC - normalmente adherida, integrada o encapsulada dentro del bien. Estas etiquetas almacenan un identificador único, que actúa como “huella digital” del objeto y es utilizado para validarlo en distintas partes del flujo de uso.

La lectura NFC se implementa utilizando las APIs nativas de Android, a través de NfcAdapter y Tag, lo que permite una lectura directa y rápida desde dispositivos compatibles. El lector detecta múltiples tipos de etiquetas (Mifare Classic, NDEF, ISO 14443), y extrae tanto el identificador como la lista de tecnologías soportadas.

```
120     private fun readTag(tag: Tag): String {
121         val sb = StringBuilder()
122
123         val tagId = bytesToHex(tag.id)
124         sb.append("Tag ID (hex): $tagId\n")
125
126         sb.append("Tag Technologies:\n")
127         tag.techList.forEach { tech ->
128             sb.append("- $tech\n")
129         }
130
131         processNdefData(tag, sb)
132
133         processMifareData(tag, sb)
134
135         return sb.toString()
136     }
```

Imagen 15 – Leer el RFID parceando ID y tecnología de tag

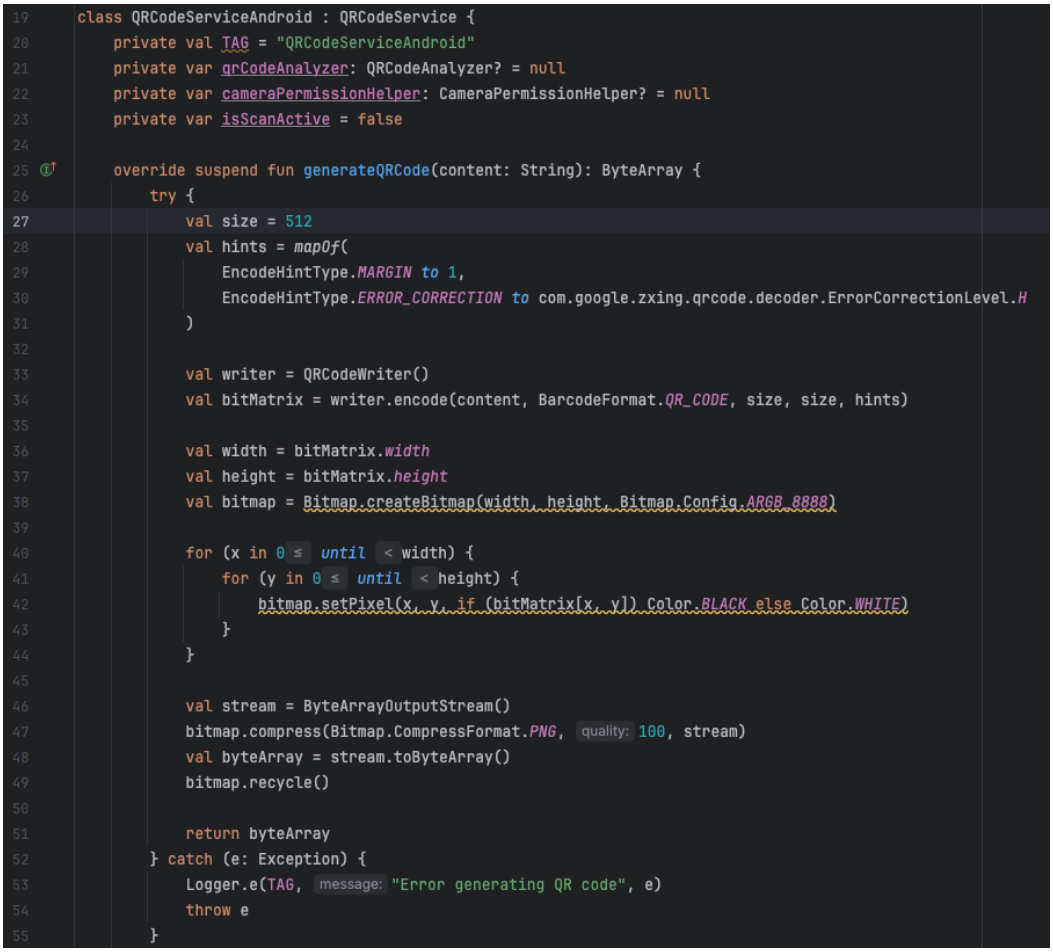
Este identificador (tagId) se compara (Imagen 15) con el ID del producto registrado, sirviendo como mecanismo de autenticación física. En el desarrollo se han realizado pruebas reales con etiquetas RFID físicas con interfaz NFC, confirmando su lectura y correcta validación dentro del sistema.

Códigos QR como medio seguro de intercambio de identidad

En InLock, los códigos QR no representan activos, sino que se utilizan exclusivamente como un mecanismo para compartir el identificador único (userId) de un usuario con otro, durante procesos de transferencia.

Por ejemplo, cuando un usuario quiere recibir la propiedad de un bien, muestra su QR personal. Este QR contiene su userId, y el propietario actual del producto lo escanea para que la app lo utilice como identificador del nuevo titular en la operación.

Este sistema evita errores humanos al ingresar IDs largos manualmente y acelera el proceso de identificación entre usuarios. Además, refuerza la privacidad, ya que el receptor no necesita mostrar más datos que su QR.



```
class QRCodeServiceAndroid : QRCodeService {
    private val TAG = "QRCodeServiceAndroid"
    private var qrCodeAnalyzer: QRCodeAnalyzer? = null
    private var cameraPermissionHelper: CameraPermissionHelper? = null
    private var isScanActive = false

    override suspend fun generateQRCode(content: String): ByteArray {
        try {
            val size = 512
            val hints = mapOf(
                EncodeHintType.MARGIN to 1,
                EncodeHintType.ERROR_CORRECTION to com.google.zxing.qrcode.decoder.ErrorCorrectionLevel.H
            )

            val writer = QRCodeWriter()
            val bitMatrix = writer.encode(content, BarcodeFormat.QR_CODE, size, size, hints)

            val width = bitMatrix.width
            val height = bitMatrix.height
            val bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888)

            for (x in 0 ≤ until < width) {
                for (y in 0 ≤ until < height) {
                    bitmap.setPixel(x, y, if (bitMatrix[x, y]) Color.BLACK else Color.WHITE)
                }
            }

            val stream = ByteArrayOutputStream()
            bitmap.compress(Bitmap.CompressFormat.PNG, quality: 100, stream)
            val byteArray = stream.toByteArray()
            bitmap.recycle()

            return byteArray
        } catch (e: Exception) {
            Logger.e(TAG, message: "Error generating QR code", e)
            throw e
        }
```

Imagen 16 – Generación de códigos QR

Con un nivel de corrección de errores alto, el QR puede leerse incluso si está parcialmente dañado o mal iluminado, lo que lo hace funcional en entornos reales. (Imagen 18)

Seguridad y validaciones cruzadas

Aunque el QR simplifica el proceso de transferencia, no reemplaza ninguna de las validaciones esenciales del sistema. Para que una transferencia se lleve a cabo, deben cumplirse todas las siguientes condiciones:

- El usuario debe estar correctamente autenticado (Firebase Auth).
- Debe escanear el producto físico con NFC y que coincida con el producto en pantalla.
- El QR del receptor debe ser válido y corresponder a un usuario registrado.
- El sistema debe registrar la transferencia tanto en Firestore como en la blockchain, sin errores.

Este enfoque por capas impide que una única acción, como escanear un QR, pueda generar cambios críticos sin haber verificado primero la propiedad física ni la identidad de los involucrados.

5.2 Desarrollo del backend

El backend de InLock está diseñado como una arquitectura modular que sirve de intermediario entre la aplicación móvil y el sistema de registro descentralizado. Su núcleo está construido sobre un sistema de trazabilidad basado en blockchain, implementado como un grafo acíclico dirigido (DAG). Esta estructura se ejecuta mediante un servidor desarrollado en Python utilizando Flask como framework web, y se complementa con un conjunto de funciones especializadas para la gestión segura de activos físicos. Es una buena solución para prototipar toda la arquitectura del DAG, antes de meter en construir algo mucho más complejo en GO.

5.2.1 Componentes principales del sistema

El backend se compone de cuatro bloques funcionales fundamentales:

Clase Node

Cada transacción individual dentro del sistema se representa mediante un objeto Node. Este nodo actúa como un bloque dentro del DAG, encapsulando:

- El identificador del activo (asset_id)
- Los datos del propietario actual y anterior
- El tipo de acción (register, transfer, stake)
- La marca de tiempo del evento
- Las referencias a nodos anteriores (padres)
- La firma y el hash asociado al contenido

Además, la clase incluye métodos para:

- Generar la firma del nodo
- Calcular el hash
- Serializar y deserializar el nodo desde y hacia estructuras tipo diccionario

Clase DAG

La clase DAG implementa la lógica de registro distribuido mediante una estructura de grafo. A diferencia de las blockchain tradicionales que forman una cadena lineal, este enfoque permite múltiples ramas activas y nodos concurrentes, eliminando cuellos de botella en validación secuencial.

Sus funciones principales incluyen:

- Controlar la lista completa de nodos y nodos activos (tips)
- Verificar la validez de nuevos nodos antes de añadirlos
- Registrar el historial de propiedad de un activo
- Confirmar la titularidad de un usuario sobre un bien
- Calcular el saldo de tokens generados por staking

- Persistir el DAG a archivos JSON (lectura y escritura)
- Verificar la integridad de la red y los hashes

Funciones de gestión de activos

El sistema incluye funciones específicas para cada operación sobre un activo:

- /register_asset: Registra un nuevo producto en el DAG
- /transfer_asset: Cambia la propiedad de un activo existente
- /stake_asset: Añade un nodo de staking para ese bien
- /verify_asset_ownership: Confirma si un usuario es el actual propietario

Estas funciones validan cada acción según las reglas de seguridad, impidiendo, por ejemplo, registros duplicados o transferencias no autorizadas.

API REST con Flask

Todo el sistema está expuesto como una API REST mediante Flask. Las rutas permiten que la aplicación móvil interactúe directamente con la lógica blockchain a través de peticiones HTTP estándar.

Entre los endpoints principales se encuentran (Tabla 2):

@POST/process_nfc_tag	Procesa escaneos NFC y valida datos asociados
@POST/register_asset	Registra un nuevo activo
@POST/transfer_asset	Transfiere propiedad de un activo existente
@POST/stake_asset	Realiza una acción de staking
@GET/verify_asset_ownership	Confirma si un usuario posee un activo
@GET/asset_history/<asset_id>	Devuelve el historial completo del activo
@GET/user_assets/<user_id>	Lista los activos que posee un usuario
@GET/user_balance/<user_id>	Devuelve el balance de tokens por staking
@GET/dag_stats	Estadísticas y resumen del DAG
@GET/verify_integrity	Verificación criptográfica de todos los nodos

Tabla 2 – Los endpoints de API REST Flask

Esta API permite la ejecución de todas las operaciones desde la app sin exponer directamente la lógica interna del DAG.

5.2.2 Características clave del sistema

Estructura DAG

El uso de un grafo acíclico dirigido permite:

- Validación simultánea de múltiples nodos
- Baja latencia para operaciones concurrentes
- Alta escalabilidad teórica
- Eliminación del minado tradicional (más ecológico)

Cada nuevo nodo requiere al menos una referencia válida a nodos anteriores, y se valida su integridad mediante una función de hash. En caso de corrupción o alteración, el sistema es capaz de detectar inconsistencias mediante el endpoint /verify_integrity.

Integración con NFC

El endpoint /process_nfc_tag sugiere una integración directa con la lectura de tags NFC, permitiendo vincular el escaneo de un bien físico con su representación en blockchain.

Esto es coherente con la filosofía de InLock: la propiedad digital solo es válida si se puede demostrar físicamente a través del objeto.

Control de acciones de activos

Cada activo puede ser objeto de tres acciones principales:

- Registro (register): solo se permite una vez por producto.
- Transferencia (transfer): requiere coincidencia con el propietario actual.
- Staking (stake): puede realizarse múltiples veces y genera balance acumulativo.
- El sistema también permite consultar todos los eventos relacionados con un activo o usuario, proporcionando así un historial completo de trazabilidad.

Persistencia y recuperación

Todo el estado del DAG puede ser exportado e importado como archivos.json, lo cual facilita la portabilidad, la auditoría y la replicación del sistema entre entornos. Este diseño simplifica futuras migraciones a bases de datos distribuidas o almacenamiento cifrado.

5.2.3 Aplicabilidad en entornos reales

El backend desarrollado permite cubrir con rigor los flujos de uso reales de InLock:

- El fabricante registra el producto en blockchain mediante un nodo register.
- El usuario lo escanea con NFC y lo asocia a su cuenta.
- Al transferirlo, se añade un nodo transfer, firmando el traspaso.
- Cada vez que el usuario activa el staking de un activo, se genera un nodo stake, que contribuye a su balance acumulado.
- Toda la información puede ser consultada y verificada públicamente (dependiendo de la configuración de privacidad).

5.3 Prototipo DAG: análisis interno y validaciones

El componente central del backend de InLock es un sistema de trazabilidad basado en una estructura DAG (Directed Acyclic Graph). Este modelo permite registrar acciones relacionadas con activos físicos de forma inmutable y verificable, manteniendo un historial completo de cada bien y de las interacciones entre usuarios. La implementación se ha realizado en Python, utilizando Flask para exponer una API REST funcional.

A diferencia de una blockchain tradicional, el modelo DAG de InLock permite múltiples flujos simultáneos y evita cuellos de botella en la validación secuencial. Cada nodo representa un evento sobre un activo (registro, transferencia, staking), y está vinculado a uno o más nodos anteriores, sin formar ciclos.

5.3.1 Estructura del nodo

Cada transacción se encapsula en un objeto de la clase Node. Esta clase contiene los campos necesarios para representar el estado del activo en un momento determinado:

- asset_id: identificador del bien físico.
- action: tipo de evento (register, transfer, staking).
- user_id: autor de la acción.
- timestamp: marca de tiempo.
- references: nodos padres (hasta 2).
- signature: firma digital simple.
- hash: huella criptográfica del nodo.
- data: campo flexible para metadatos, incluyendo recipient_id, staking_amount, o información NFC.

El método _calculate_hash() utiliza SHA-256 para firmar el contenido estructurado del nodo, protegiendo su integridad frente a manipulaciones. La firma (signature) se genera como un hash interno controlado por el usuario, con propósito de identificación interna.

5.3.2 Validaciones y reglas de seguridad

La validación de nodos es una parte esencial del sistema. Antes de añadir un nuevo nodo al DAG, el sistema ejecuta las siguientes comprobaciones:

- No se permite registrar el mismo activo más de una vez.
- En una transferencia, el usuario debe ser el propietario actual.
- En staking, también debe coincidir con el propietario registrado.
- El nodo no puede tener más de dos referencias (nodos padres).
- Cada nodo referenciado debe existir previamente.

Ejemplo de validación real (Imagen 17):

```
114
115         elif node.action == "transfer":
116             owner_history = self.get_asset_ownership_history(node.asset_id)
117             if not owner_history:
118                 return False, f"Asset {node.asset_id} is not registered"
119
120             current_owner = owner_history[-1]["user_id"]
121
122             if current_owner != node.user_id:
123                 return False, f"Transfer requested by {node.user_id}, but asset is owned by {current_owner}"
124
125             if "recipient_id" not in node.data:
126                 return False, "Transfer must include a recipient_id in the data"
127
```

Imagen 17 – Ejemplo de código de validación a la hora de transferir

Este mecanismo evita ataques de falsificación, uso indebido de IDs y mantiene la consistencia lógica del sistema.

5.3.3 Gestión del DAG y persistencia

La clase DAG gestiona toda la red de nodos. Mantiene una colección en memoria y un conjunto de nodos “tips”, es decir, aquellos que aún no han sido referenciados por ningún otro nodo. Se incluyen funciones para:

- Agregar nuevos nodos (add_node)
- Validar la integridad (verify_integrity)
- Guardar y cargar la red en archivos .json (save / load)
- Consultar el historial de propiedad de un activo (get_asset_ownership_history)
- Calcular el balance de staking (get_user_staking_balance)
- Identificar activos actuales de un usuario (get_user_assets)

La persistencia se realiza en el archivo blockchain_dag.json, lo que permite recuperar el estado del sistema tras cada reinicio (Imagen 18):

```
201
202     def save(self):
203         data = {
204             "nodes": {node_id: node.to_dict() for node_id, node in self.nodes.items()},
205             "tips": list(self.tips)
206         }
207
208         with open(self.storage_path, "w") as f:
209             json.dump(data, f, indent=2)
```

Imagen 18 – Guardando los nodos

5.3.4 Acciones soportadas: registro, transferencia y staking

El sistema reconoce tres tipos de acciones:

- register_asset

Registra un nuevo bien físico, asegurando que no haya sido registrado antes.

- transfer_asset

Cambia la titularidad de un bien, validando que el usuario origen sea el propietario actual, y que se incluya el campo recipient_id.

- stake_asset

Permite que el usuario actual del bien genere tokens mediante staking, acumulando balance en función del historial.

Cada una de estas acciones se implementa como una función con reglas específicas, integradas dentro del sistema (Imagen 19):

```
255  def transfer_asset(blockchain: DAG, asset_id: str, from_user_id: str, to_user_id: str) -> Tuple[bool, str]:
256      references = blockchain.choose_references()
257
258      node = Node(
259          asset_id=asset_id,
260          action="transfer",
261          user_id=from_user_id,
262          references=references,
263          data={"recipient_id": to_user_id}
264      )
265
266      return blockchain.add_node(node)
```

Imagen 19 – Ejemplo de código de acción “transfer_asset”

5.3.5 API REST para interacción desde la app

El sistema expone todas sus funciones mediante un conjunto de rutas HTTP accesibles por la aplicación móvil a través de Ktor Client (Tabla 3):

POST /process_nfc_tag	Registra o realiza staking sobre un activo según escaneo
POST /register_asset	Añade un nodo de registro
POST /transfer_asset	Transfiere propiedad
POST /stake_asset	Añade nodo de staking
GET /verify_ownership	Verifica si un usuario posee un activo
GET /asset_history/<asset_id>	Muestra todo el historial del bien
GET /user_assets/<user_id>	Devuelve los activos actuales del usuario
GET /user_balance/<user_id>	Devuelve el total de tokens generados
GET /verify_integrity	Comprueba integridad criptográfica
GET /blockchain_stats	Muestra métricas globales del DAG

Tabla 3 – Los endpoints a los que habla el Ktor desde la APP

El endpoint /process_nfc_tag es especialmente interesante, ya que decide dinámicamente si registrar o stakear un activo según su existencia previa (Imagen 20):

```
309         asset_exists = False
310         for node in blockchain.nodes.values():
311             if node.asset_id == tag_id and node.action == "register":
312                 asset_exists = True
313                 break
314
315         if asset_exists:
316             success, result = stake_asset(blockchain, tag_id, user_id)
317             return jsonify({
318                 "success": success,
319                 "result": result,
320                 "action": "staking",
321                 "asset_id": tag_id
322             })
323         else:
324             success, result = register_asset(blockchain, tag_id, user_id, asset_data)
325             return jsonify({
326                 "success": success,
327                 "result": result,
328                 "action": "register",
329                 "asset_id": tag_id
330             })
331
332     except Exception as e:
333         logger.error(f"Error processing NFC tag: {str(e)}", exc_info=True)
334         return jsonify({"success": False, "message": f"Server error: {str(e)}"}), 500
```

Imagen 20 – Comprobación de estado del tag

5.4 Integración entre componentes

El sistema InLock se construye a partir de múltiples capas funcionales que interactúan entre sí para permitir el registro, la verificación, la transferencia y el seguimiento de activos físicos. Estas capas —la aplicación móvil, el backend en Python con Flask, y el módulo de blockchain DAG— están diseñadas para funcionar de forma coordinada, manteniendo la separación de responsabilidades, pero compartiendo información esencial en tiempo real.

La integración entre componentes se basa en llamadas HTTP desde el cliente (la app móvil) hacia el backend, que a su vez opera directamente sobre la estructura de datos del DAG para validar, almacenar y consultar información relativa a cada bien registrado.

5.4.1 Flujo general de una operación completa

Un ejemplo típico de flujo es la transferencia de propiedad de un activo. Este proceso implica a los tres componentes principales:

Aplicación móvil (Kotlin Multiplatform):

- El usuario inicia la transferencia desde la interfaz.
- Escanea el producto físicamente mediante NFC para verificar posesión.
- Escanea el QR del receptor, que contiene su userId.
- La app valida que el usuario está autenticado y que el activo le pertenece.
- Envía la solicitud HTTP al backend con los campos necesarios.

Backend Flask:

- Recibe la solicitud vía POST /transfer_asset.
- Verifica que el from_user_id corresponde con el propietario actual.
- Crea un nuevo nodo transfer en el DAG si todas las condiciones se cumplen.
- Blockchain DAG (módulo interno en Python):
    - Valida el nodo localmente.
    - Añade el nodo a la estructura del grafo.
    - Guarda el nuevo estado en disco (JSON).
    - Devuelve una confirmación de éxito o error al backend.
    - La app interpreta el resultado y actualiza su UI en tiempo real.

5.4.2 Comunicación entre la app y el backend

Desde el lado de la aplicación, la comunicación se implementa usando Ktor Client, configurado con seguridad y tiempos de espera estrictos para evitar bloqueos. Un ejemplo de llamada a la API para registrar un nuevo activo (Imagen 21):

```
73      suspend fun registerAsset(
74          assetId: String,
75          userId: String,
76          assetData: Map<String, String>? = null
77      ): Result<String> = runCatching {
78          Logger.d(TAG, message: "Registering asset $assetId for user $userId with data: $assetData")
79
80          try {
81              withTimeout( timeMillis: 15000) {
82                  val response = httpClient.post( urlString: "$baseUrl/register_asset") {
83                      contentType(ContentType.Application.Json)
84                      setBody(RegisterAssetRequest(assetId, userId, asset_data: assetData ?: emptyMap()))
85                  }
86
87                  val result: BlockchainResponse = response.body()
88
89                  if (result.success) {
90                      Logger.d(TAG, message: "Asset registered successfully: ${result.result}")
91                      return@withTimeout result.result
92                  } else {
93                      Logger.e(TAG, message: "Asset registration failed: ${result.result}")
94                      throw Exception(result.result)
95                  }
96              }
97          } catch (e: TimeoutCancellationException) {
98              Logger.e(TAG, message: "Asset registration timed out", e)
99              throw Exception("Asset registration timed out")
100         } catch (e: Exception) {
101             Logger.e(TAG, message: "Asset registration failed", e)
102             throw e
103         }
104     }
```

Imagen 21 – Código de la app para comunicar con los endpoints de blockchain

La respuesta JSON del backend se analiza para mostrar mensajes apropiados y actualizar el estado interno del usuario o del producto en la app.

5.4.3 Dependencia de integridad cruzada

Uno de los aspectos más importantes de la integración es la verificación cruzada entre fuentes. Por ejemplo, la propiedad de un producto no se decide únicamente desde Firestore (la base de datos asociada a Firebase), sino también desde el DAG blockchain. Si ambos están en conflicto, el sistema puede:

- Notificar al usuario de un estado inconsistente.
- Forzar la sincronización desde blockchain como fuente de verdad.
- Actualizar Firestore si se detecta un desfase válido.

Este proceso está centralizado en funciones como forceProductSync() del lado cliente, que reescribe los datos de Firestore usando la información registrada en blockchain tras consultar el endpoint /asset_history/<asset_id>.

5.4.4 Modularidad y escalabilidad

Cada componente puede evolucionar de manera independiente:

- La app móvil puede modificar su interfaz o modelo de estados sin alterar el backend.
- El backend Flask puede migrarse a FastAPI, Node.js o servicios serverless sin cambiar el protocolo de comunicación.
- El DAG puede ser persistido en una base de datos real (como MongoDB o Neo4j) sin modificar la lógica de alto nivel.

Gracias a esta separación, el sistema está preparado para ampliarse con nuevas funciones - como verificación de productos por terceros, reportes públicos de robos, o reputación de usuarios - sin comprometer la integridad de lo ya construido.

5.4.5 Seguridad entre componentes

Todos los puntos de interacción entre capas están protegidos mediante:

- Autenticación con Firebase Auth: ningún usuario no autenticado puede iniciar operaciones.
- Validación estricta de roles: desde la app y también en backend.
- Verificación NFC previa a cada acción crítica: asegurando posesión real del activo.
- Control de errores exhaustivo: cada llamada HTTP maneja posibles fallos y estados inesperados.
- Logs y auditoría: el backend genera logs detallados para rastrear cada acción, útil tanto para debugging como para pruebas forenses.

En conjunto, la integración entre componentes en InLock no es solo una cuestión de conectividad. Está diseñada para garantizar que todos los subsistemas trabajen juntos de forma segura, eficiente y coherente. El modularidad alcanzado permite escalar tanto

horizontal como verticalmente, mientras que la validación cruzada entre blockchain, Firestore y la app ofrece una protección sólida frente a inconsistencias o ataques.

5.5 Problemas técnicos y decisiones adoptadas

El desarrollo de InLock ha implicado el diseño de una arquitectura compleja que conecta hardware físico, servicios en la nube, y un sistema blockchain personalizado. Si bien los objetivos técnicos han sido alcanzados en gran medida, el camino ha estado marcado por múltiples obstáculos que han requerido adaptaciones, rediseños parciales y decisiones que priorizan funcionalidad y coherencia sobre ambiciones excesivas.

5.5.1 Limitaciones de lectura NFC multiplataforma

Uno de los retos iniciales fue la integración de lectura NFC en un entorno Kotlin Multiplatform. Si bien Android ofrece un soporte completo y bien documentado para NFC y RFID, la compatibilidad con iOS es limitada. Debido a las restricciones impuestas por el sistema operativo de Apple, especialmente en accesos de bajo nivel a hardware NFC, se optó por centrar las pruebas y el desarrollo de esta funcionalidad exclusivamente en Android.

Esta decisión permitió avanzar con una base técnica sólida sin comprometer la fiabilidad del escaneo. Se estableció, además, un diseño desacoplado, que permitirá en el futuro integrar librerías nativas específicas para iOS o incluso migrar a hardware dedicado con conectividad BLE si fuera necesario.

5.5.2 Integración real de roles con Firebase Auth

Aunque Firebase Authentication ofrece autenticación sólida y escalable, no incluye de forma nativa un sistema de control de roles. Para implementar un modelo RBAC completo, se exploraron varias estrategias:

- Uso de claims personalizados con Firebase Admin SDK esta descartado temporalmente por limitaciones de tiempo y control desde cliente.
- Asignación de roles vía documentos en Firestore a opción elegida.

Los roles se almacenan junto con los datos del usuario, se cachean tras login, y se validan en múltiples capas (UI, servicios, backend). Esta decisión ofreció un equilibrio adecuado entre seguridad, rendimiento y flexibilidad.

5.5.3 Validación de consistencia entre Firestore y blockchain

Un problema no trivial fue definir qué sistema debía actuar como fuente de verdad: Firestore (rápido y estructurado) o el sistema DAG (inmutable y descentralizado). Para mantener una arquitectura coherente se optó por lo siguiente:

- Las operaciones de registro y transferencia se reflejan en ambos sistemas.
- El DAG actúa como fuente de verdad final.

- En caso de discrepancia, la app puede forzar una sincronización desde blockchain.

Esta decisión, aunque añade cierta complejidad, asegura que los datos presentados al usuario siempre estén alineados con los registros inmutables de blockchain.

5.5.4 Problemas con el modelado de staking

La primera implementación del staking asumía un modelo simple: cada acción sumaba una unidad de token. Sin embargo, este enfoque no tenía en cuenta:

- El valor relativo del bien.
- La frecuencia de escaneo.
- La posible delegación de staking a terceros.

Dado el alcance del TFG, se optó por conservar un modelo funcional básico que puede evolucionar. Actualmente, el staking genera balance basado en acciones registradas, y se ha dejado preparada la estructura de datos para incorporar métricas más complejas en el futuro (staking_amount, timestamp, etc.).

5.5.5 Serialización, persistencia y escalabilidad del DAG

Durante las primeras versiones, el DAG se mantenía únicamente en memoria. Esto generaba problemas obvios: pérdida de datos tras reinicio, imposibilidad de compartir el estado entre instancias, y dificultades para pruebas a largo plazo. Como solución, se implementó persistencia local mediante archivos.json.

Este enfoque, aunque rudimentario, cumple con el objetivo de validación de concepto. Se prevé su migración a:

- Base de datos orientada a grafos (Neo4j, ArangoDB)
- Almacenamiento con cifrado y firma
- Replicación entre nodos distribuidos (Kuber)

5.5.6 Decisiones estratégicas para entrega del prototipo

Algunas funciones planificadas no fueron integradas en esta primera versión, entre ellas:

- Wallet integrada para los tokens
- Capa visual de staking y balance
- Control automatizado de reputación de usuarios

La decisión de dejar fuera estas funcionalidades se tomó conscientemente para mantener el foco en la trazabilidad de propiedad, que es el eje central del proyecto. No obstante, su integración está prevista y la arquitectura ya está preparada para recibirlas sin refactorizaciones mayores.

5.5.7 Complejidad de pruebas con chip RFID real

El uso de un chip físico con tecnología NFC introdujo retos prácticos. Entre ellos:

- Inestabilidad del escaneo en ciertos dispositivos.
- Necesidad de mantener contacto físico prolongado.
- Lecturas duplicadas por parte del sistema operativo.

Se aplicaron medidas como filtros de debounce y comparación con producto seleccionado, además de pruebas repetidas con distintos escenarios para asegurar la robustez del flujo de escaneo.

5.5.8 Repositorio de código

Todo el código desarrollado para este proyecto está disponible públicamente en GitHub en la siguiente URL: https://github.com/tigershawt/InLock-TFG-Primitive-Version-Prototype

Dirección del código desarrollado por el autor (Imagen 22):

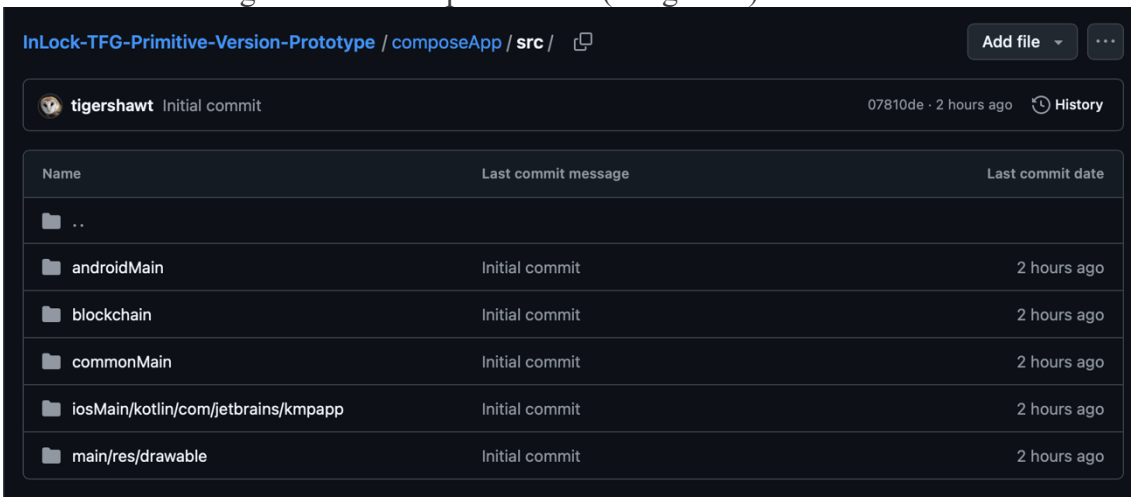


Imagen 22 – Ruta de código en GitHub relevante a la aplicación

El repositorio incluye los siguientes componentes:

- Aplicación móvil desarrollada en Kotlin Multiplatform
- Backend API implementado en Flask
- Prototipo DAG para la blockchain

Nota sobre uso de herramientas de IA: Las funciones de logging en el backend fueron generadas parcialmente con asistencia de Claude (Anthropic) como se indica en los comentarios del código.

6. Resultados

El sistema InLock, en su versión prototipo, ha alcanzado un nivel funcional suficiente para validar de forma sólida los conceptos de trazabilidad, validación de propiedad y seguridad física aplicada a activos digitales. A continuación se presentan los resultados obtenidos tras el desarrollo de las principales funcionalidades del proyecto, con énfasis en la ejecución práctica y la demostración técnica de su viabilidad.

6.1 Flujo completo de creación de activo

Uno de los objetivos fundamentales del sistema InLock es permitir que un fabricante autorizado pueda registrar un bien físico de forma segura, inviolable y verificable, vinculándolo de manera permanente a un identificador RFID único. Este proceso se apoya en múltiples mecanismos de validación que garantizan tanto la autenticidad física del producto como la legitimidad del usuario que lo registra.

A continuación, se describe el flujo técnico de creación de un activo genuino, paso a paso:

Autenticación y control de rol del usuario

Para iniciar el proceso de registro de activos, el usuario debe haber sido asignado previamente con el rol de "fabricante", lo que le habilita a operar sobre endpoints específicos de la API B2B.

- Al abrir la app, se verifica el token JWT del usuario para validar su sesión activa.
- El backend comprueba que el rol del usuario incluya privilegios de fabricante.
- Se verifica además que el usuario cuente con una clave criptográfica válida asociada a su identidad, la cual será utilizada para firmar los activos registrados.

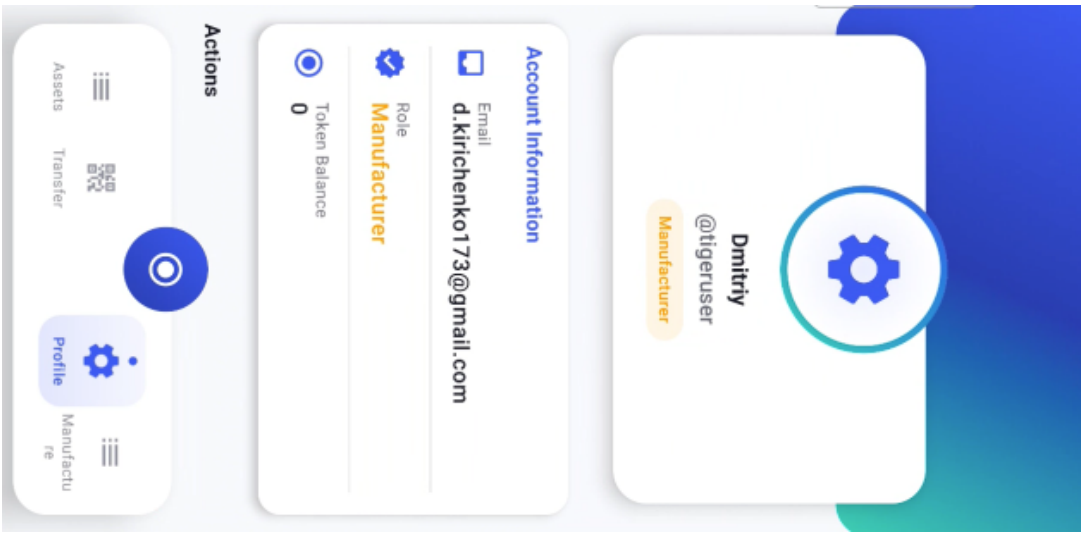


Imagen 23 – Pagina de perfil de usuario en la APP

Uso de plantillas de producto

Para optimizar el flujo en contextos de producción en serie, el sistema permite a los fabricantes crear plantillas personalizadas, que actúan como definiciones previas del producto. Estas plantillas (Imagen 24) pueden incluir:

- Imágenes genéricas del producto.
- Descripción técnica o comercial.
- Parámetros definidos por el fabricante (peso, color, material, etc.).
- Reglas de staking o uso futuro.

El objetivo es reducir el tiempo de registro permitiendo reutilizar configuraciones comunes en múltiples activos.

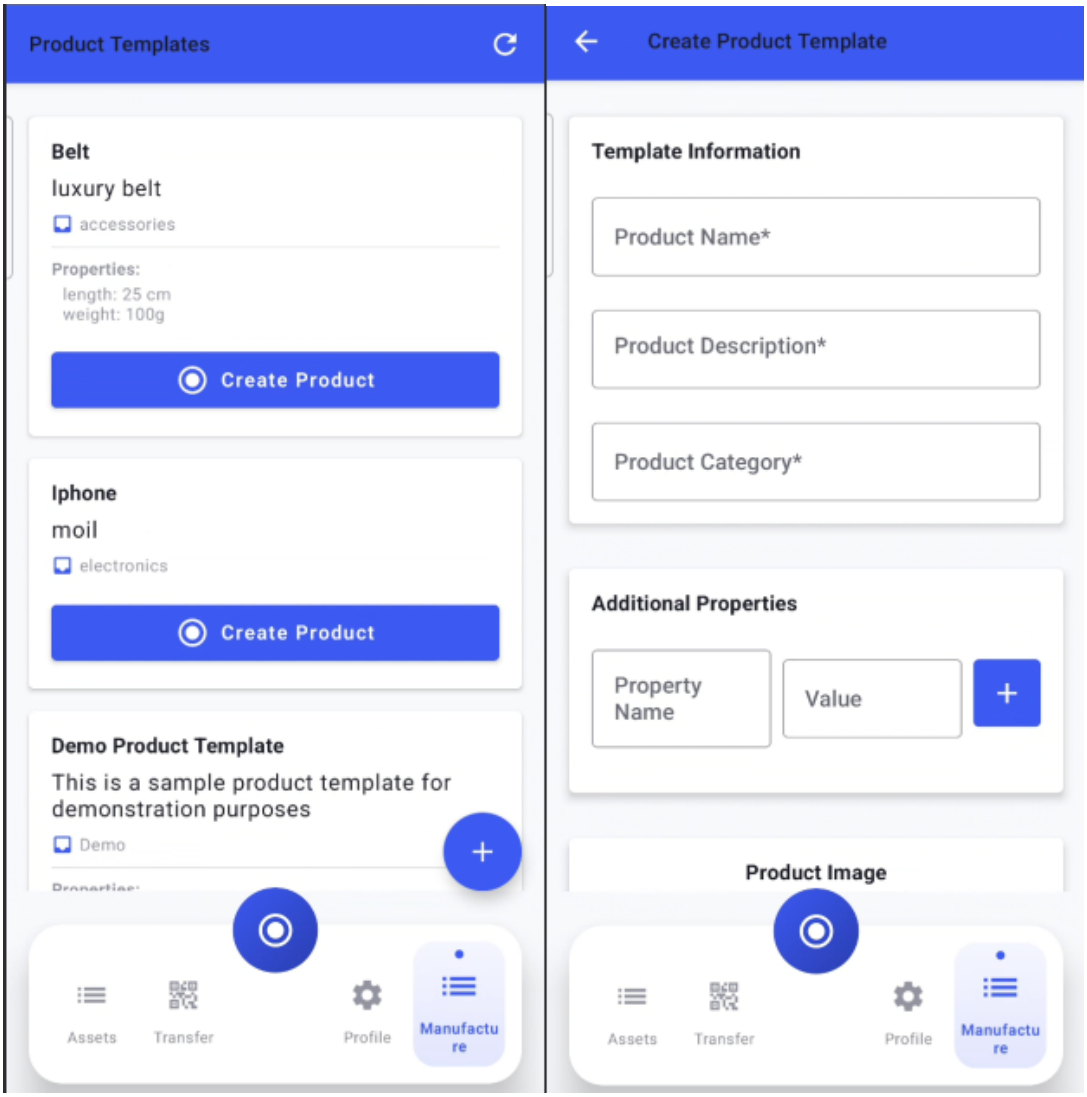


Imagen 24 – Paginas de creación de producto y de las plantillas

Instanciación del producto físico

Desde la sección de plantillas (Imagen 25), el fabricante puede iniciar el proceso de instanciación de un producto físico real. Para ello, debe:

- Escanear una etiqueta RFID física con tecnología write-once y soporte NFC.
- El sistema lee el identificador (UID) del chip.
- La app verifica que el chip no haya sido registrado anteriormente.
- Opcionalmente, se comprueba la firma o clave interna del chip, si el hardware RFID dispone de seguridad avanzada (como NTAG 424 DNA).

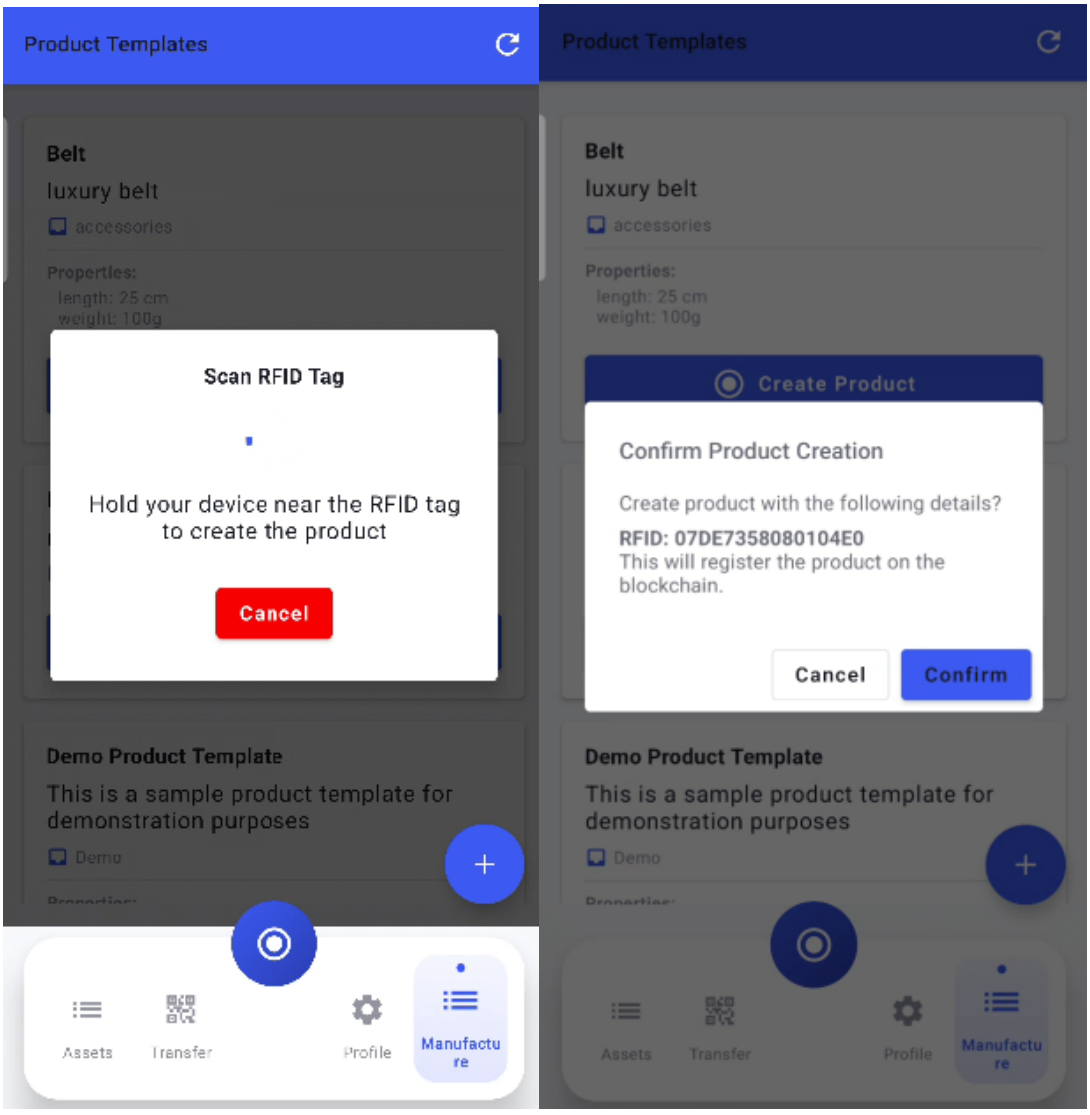


Imagen 25 – Instanciar producto en blockchain con comprobación de chip fisico

Registro del activo en blockchain DAG

Una vez verificados el chip físico, la sesión del usuario y la plantilla seleccionada (flujo explicado respeto a imagen 26):

- Se genera un nodo register en la blockchain DAG.
- El nodo contiene:
    - El ID físico del RFID.
    - Los metadatos definidos en la plantilla.
    - El identificador del fabricante.
    - Un timestamp.
    - La firma digital del fabricante (según clave asociada).

- El nodo se enlaza estructuralmente al DAG de forma inmutable.
- Firestore guarda referencias cruzadas y metadatos auxiliares (como miniaturas o descripciones).

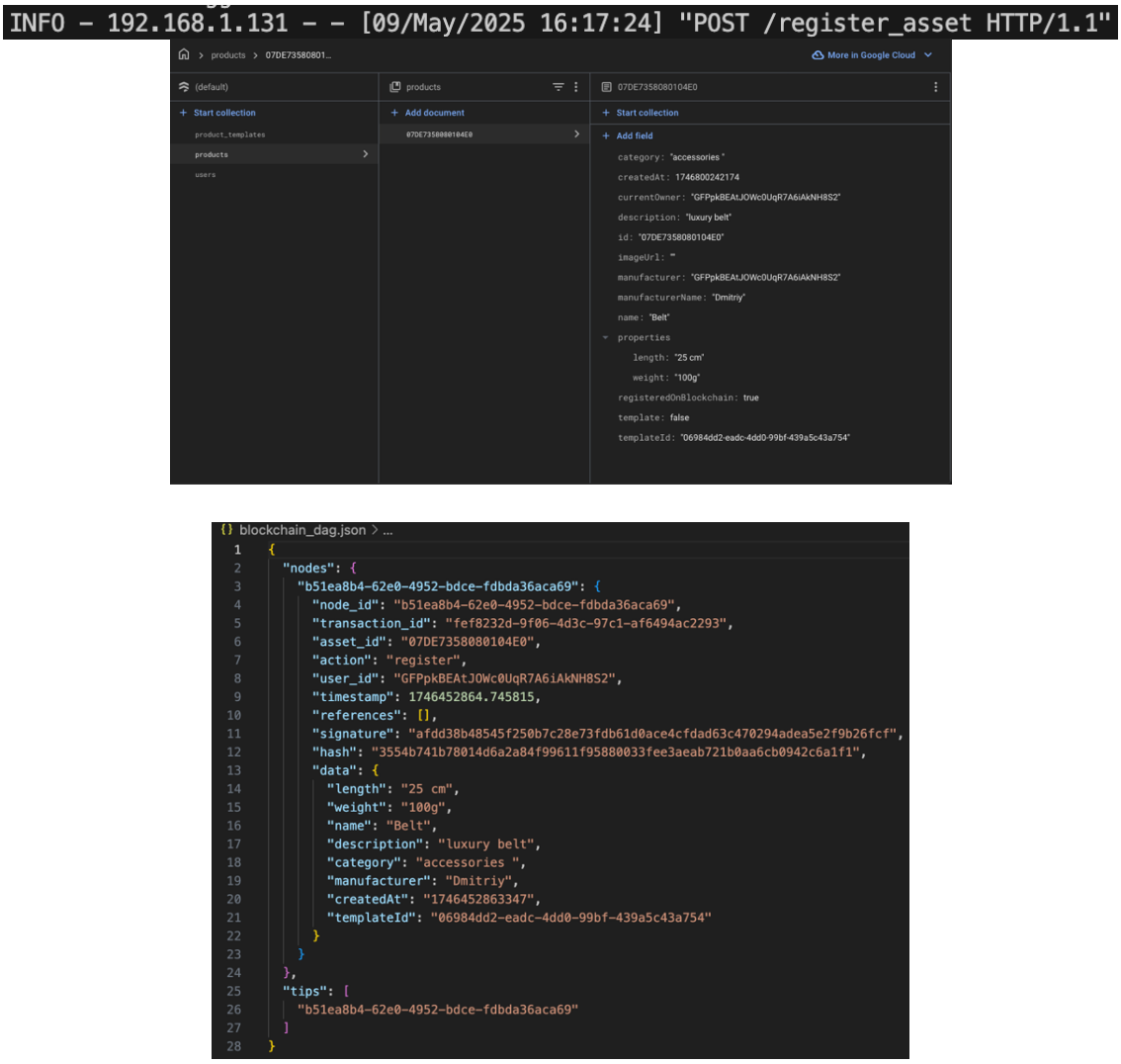


Imagen 26 – Proceso y datos almacenados

Confirmación visual al usuario

Tras completar el registro:

- El sistema muestra un mensaje de éxito al fabricante.
- El producto aparece marcado como "verificado y registrado" (Imagen 27).
- Desde la app, el bien queda disponible para su visualización, transferencia o staking, dependiendo del caso.

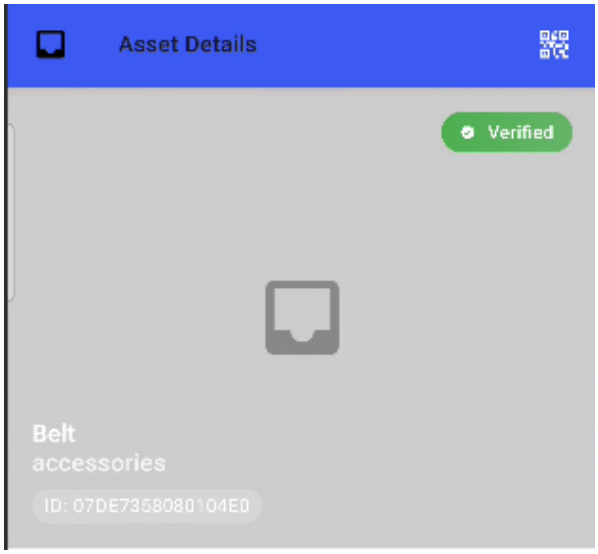


Imagen 27 – Confirmación visual de existencia de producto

Este flujo garantiza que solo los fabricantes verificados puedan registrar productos auténticos y que cada bien quede vinculado de forma exclusiva a un chip físico inalterable, con un historial auditado y público. Además, la flexibilidad del sistema de plantillas permite escalar el registro de activos sin sacrificar control ni trazabilidad.

6.2 Transferencia de propiedad con validación física

El sistema InLock permite la transferencia de propiedad de activos físicos entre usuarios de forma segura, auditada y resistente a manipulaciones. Este proceso ha sido diseñado para exigir tanta identificación digital del nuevo propietariocomo verificación física de posesión del bien, asegurando que solo el titular legítimo con acceso al producto pueda ejecutar la operación.

Este flujo se ha validado funcionalmente en escenarios reales y se realiza íntegramente desde la app móvil:

Escaneo del QR del receptor

El proceso comienza con la identificación del nuevo propietario. El usuario actual escanea un código QR generado por el receptor, el cual contiene su userId de forma segura.

Esto permite que el sistema sepa a quién se desea transferir el activo.

El escaneo se realiza mediante la cámara integrada en la app, evitando errores manuales o suplantaciones (Imagen 28).

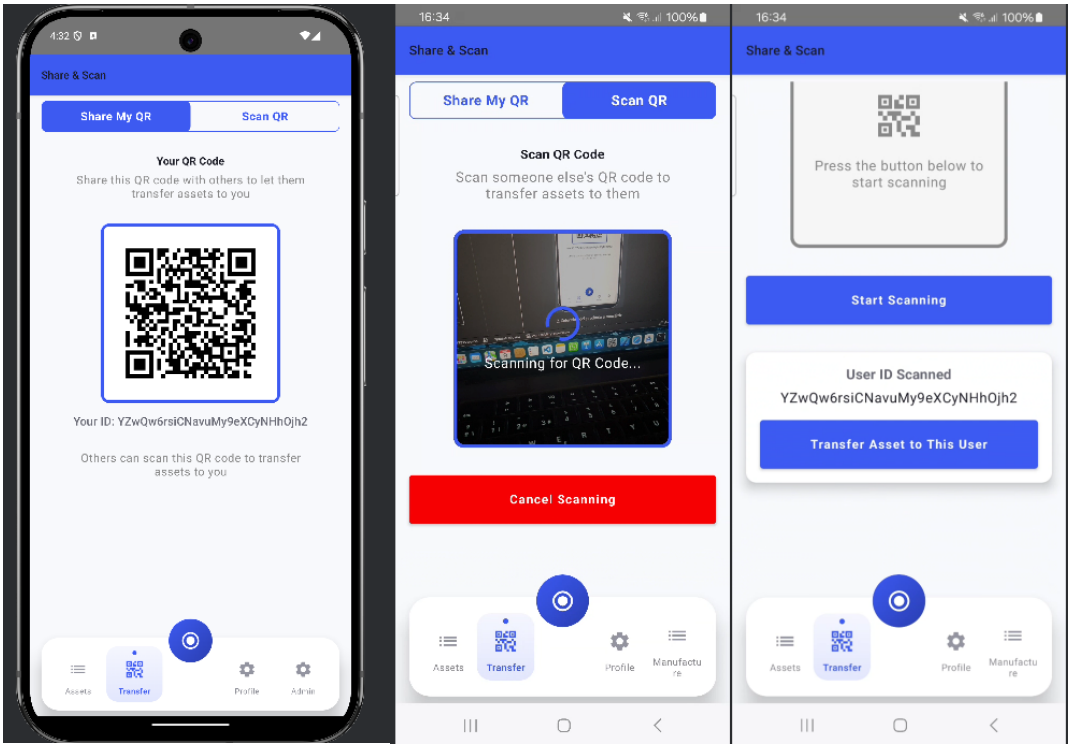


Imagen 28 – Proceso de escaneo de QR de usuario por usuario

Selección del producto a transferir

Tras capturar la identidad del destinatario, el propietario elige desde su listado el producto que desea transferir.

- Solo se listan activos cuya propiedad actual corresponde al usuario autenticado.
- La app muestra los detalles clave del bien, permitiendo revisar antes de confirmar (Imagen 29).

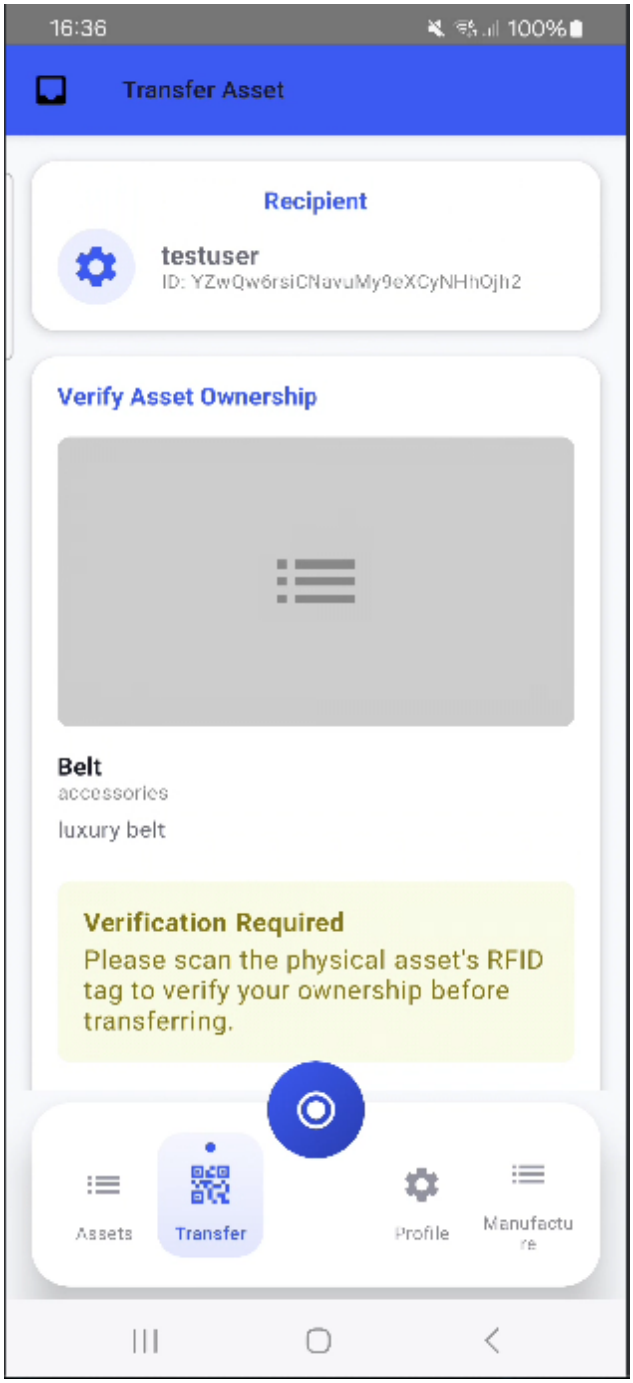


Imagen 29 – Elección de producto por transferir

Verificación física del bien (lectura NFC)

Antes de completar la operación, el sistema requiere una verificación física para evitar transferencias de bienes no presentes (Imagen 30):

- El usuario escanea con su teléfono la etiqueta NFC embebida en el producto.
- La app comprueba que el identificador RFID corresponde exactamente con el bien seleccionado.
- Si hay discrepancia o el UID no se detecta correctamente, la operación es cancelada.

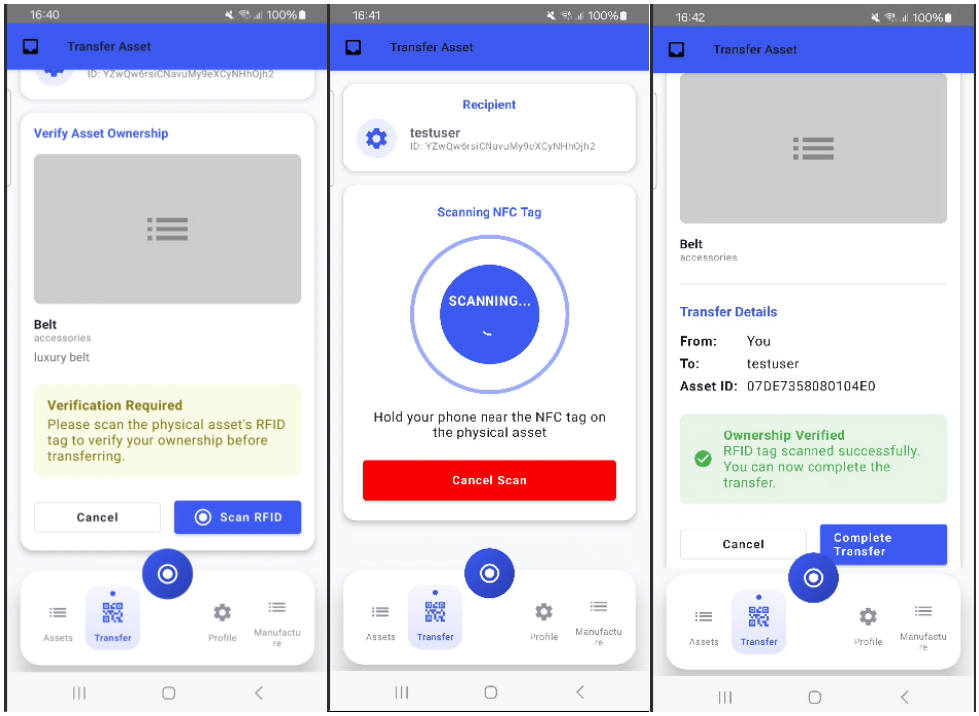


Imagen 30 – Proceso visual de la app a confirmar posesión física de bien

Registro de la transferencia en blockchain

Una vez validados el receptor, el bien y la posesión física:

- La app envía una solicitud al endpoint /transfer_asset, firmada digitalmente.
- El backend comprueba que:
    - El usuario autenticado es el propietario actual (según DAG y Firestore).
    - El activo no está marcado como robado o bloqueado.
    - El escaneo NFC fue reciente y válido.
- Se genera un nodo transfer en la DAG blockchain, registrando:
    - El ID del activo.
    - El from_user_id (emisor) y to_user_id (receptor).
    - Timestamp y firma digital.

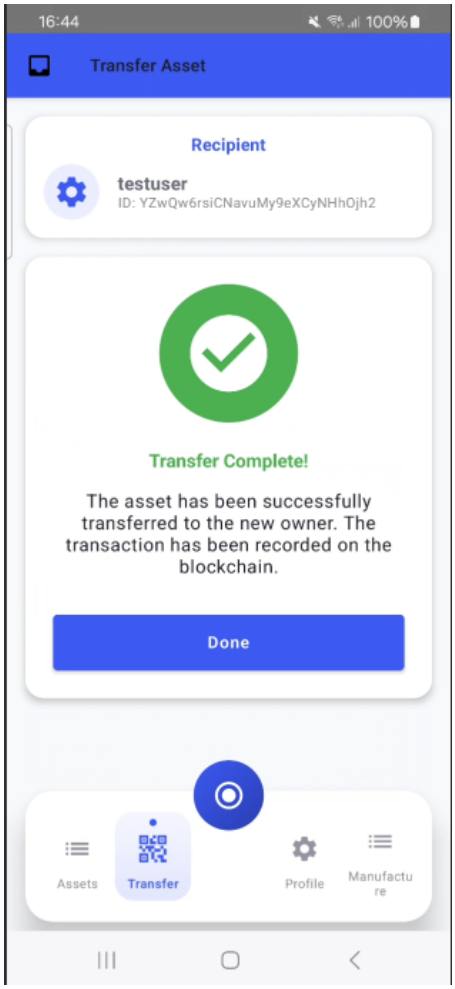


Imagen 31 – Respuesta visual de éxito de tranfer_asset

Resultado técnico visible (Imagen 32):

- Nodo transfer registrado en la DAG, con referencias criptográficas y firma digital.
- Actualización del campo currentOwner en Firestore.
- Notificación visual al emisor y receptor con confirmación de la transferencia.

Este proceso asegura que toda transferencia quede registrada de forma inmutable y públicamente verificable, garantizando tanto la transparencia como la integridad de la propiedad digital de activos físicos.

Imagen 32 – Resultado escrito en nodo de Blockchain y Firestore collection

6.3 Verificación de integridad del DAG

Una característica esencial del sistema InLock es la capacidad de garantizar en todo momento la integridad estructural y criptográfica de la red DAG (Directed Acyclic Graph), que actúa como base de datos inmutable para registrar todos los eventos relacionados con los activos físicos.

Para ello, el backend expone un endpoint específico /verify_integrity (Imagen 33), que ejecuta una verificación profunda del estado actual del grafo. Esta operación se basa en tres validaciones clave:

- Conectividad estructural: se comprueba que todos los nodos están correctamente referenciados, y que no existen nodos huérfanos ni bucles cíclicos.
- Consistencia de hashes: cada nodo contiene un hash propio calculado a partir de sus datos y referencias. La verificación implica recalcular cada uno y compararlo con su valor original, garantizando que ningún nodo ha sido modificado desde su creación.
- Integridad referencial: se revisa que todas las referencias a nodos padres existan efectivamente en la DAG, evitando errores lógicos o corrupción parcial del grafo.

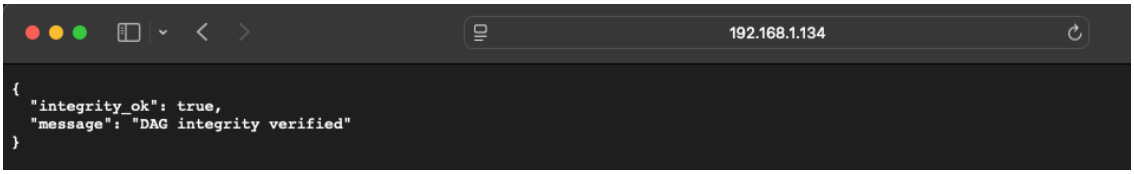


Imagen 33 – Comprobación de integridad

6.4 Verificación pública del historial de propiedad

El sistema InLock permite que cualquier persona, incluso sin registrarse previamente, pueda verificar la autenticidad de un producto físico registrado en la red, así como consultar su estado actual y el historial de propiedad. Esta funcionalidad garantiza un modelo de verificación abierta, segura y sin fricción, donde compradores, autoridades o ciudadanos puedan confirmar al instante si un bien es legítimo, robado, falsificado o pendiente de transferencia.

Verificación universal: sin necesidad de cuenta

La aplicación móvil incluye una sección pública de escaneo NFC que puede ser utilizada por cualquier persona. El proceso es extremadamente simple:

1. El usuario accede a la pantalla de escaneo desde la app.
2. Acerca el teléfono a la etiqueta RFID del producto.
3. La app lee el UID único del chip NFC.
4. Se realiza una consulta directa a la blockchain DAG y a Firestore.
5. El sistema devuelve la información verificada del activo, incluyendo:

- Propietario actual.
- Fabricante original (firmado digitalmente).
- Historial resumido de propiedad.
- Estado del activo (válido, robado, transferido, etc.).

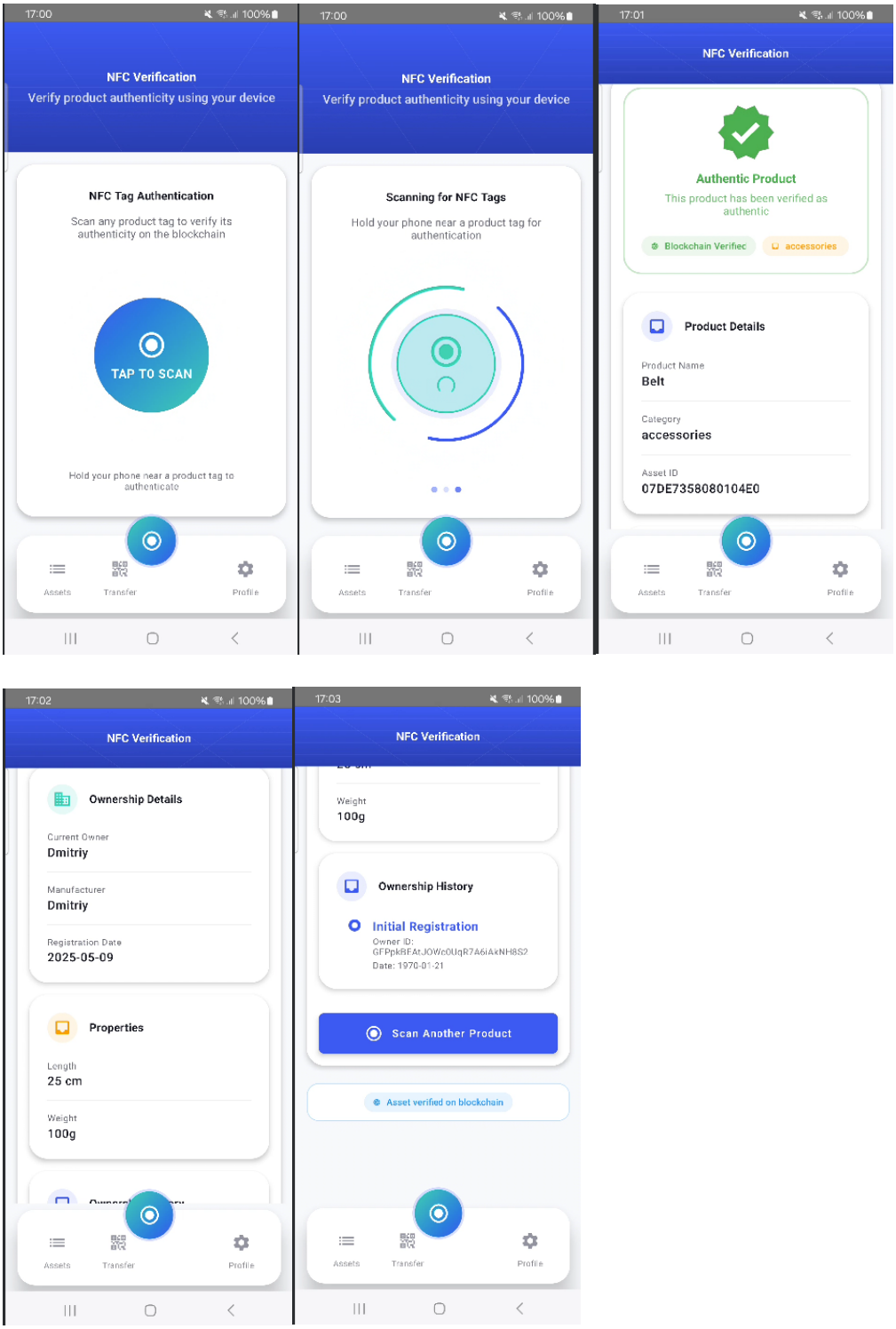


Imagen 34 – Demostración de interfaz y procedimiento desde vista de UX/UI del punto 6.4

Rol guest: propiedad legítima sin registro formal

Cuando un usuario sin cuenta adquiere un producto, el sistema lo registra como propietario legítimo en la blockchain y en Firestore. Sin embargo, como aún no tiene una cuenta tradicional, la vinculación de derechos de acceso a ese activo se asocia al identificador único del dispositivo móvil (por ejemplo, ID del teléfono o instancia local generada (Imagen 35)).

Esto implica que:

- El activo es completamente trazable y está registrado como suyo en la blockchain.
- Solo ese dispositivo podrá ver, consultar o transferir ese bien hasta que el usuario cree una cuenta real.
- Al crear una cuenta, el sistema vinculará automáticamente los activos existentes con el nuevo perfil del usuario, preservando su historial.

Este enfoque ofrece una transición sin fricción entre uso anónimo y participación formal, sin comprometer la seguridad, la trazabilidad ni la propiedad legítima de los bienes adquiridos.

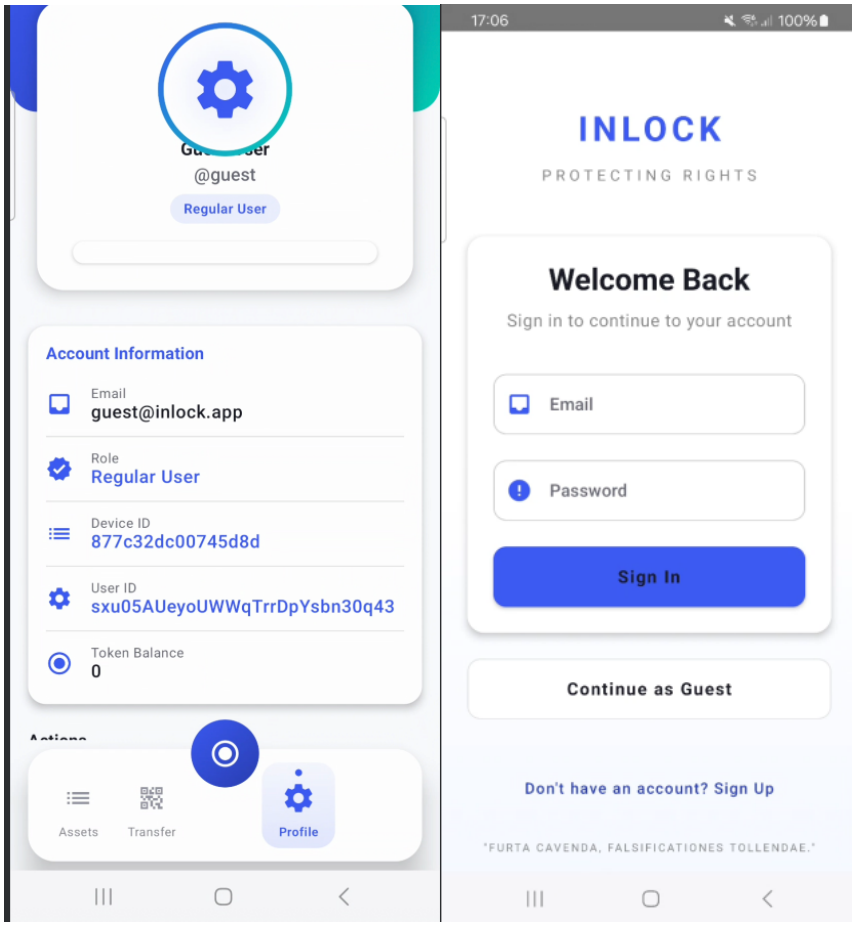


Imagen 35 – Creación de cuenta anónima con IMAE del dispositivo

Esta combinación de verificación pública abierta y gestión flexible de propiedad convierte a InLock en una solución poderosa para la protección real de activos físicos, sin sacrificar accesibilidad ni control. La posibilidad de validar un bien sin cuentas ni infraestructura adicional empodera al ciudadano, reduce barreras tecnológicas y refuerza la confianza en entornos de compraventa o inspección.

6.5 Resultados de integración entre app y backend

Una parte esencial del desarrollo del sistema InLock ha sido lograr una integración fluida, segura y eficiente entre la aplicación móvil multiplataforma y el backend del sistema, compuesto por una API REST desarrollada en Ktor y un motor blockchain DAG independiente.

Esta integración ha sido validada durante la ejecución de flujos completos de operación como registro de activos, verificación, transferencia y consulta pública, confirmando que cada capa de la arquitectura coopera correctamente en tiempo real. (Imagen 36)

Elementos validados en la integración:

- Comunicación NFC →App → API

    - La lectura de etiquetas RFID mediante NFC desde la app genera solicitudes HTTP hacia el backend, que se reciben correctamente y son procesadas sin errores.
    - La información capturada desde el chip (UID) se utiliza como clave primaria para verificar existencia, consultar historial o iniciar registros.

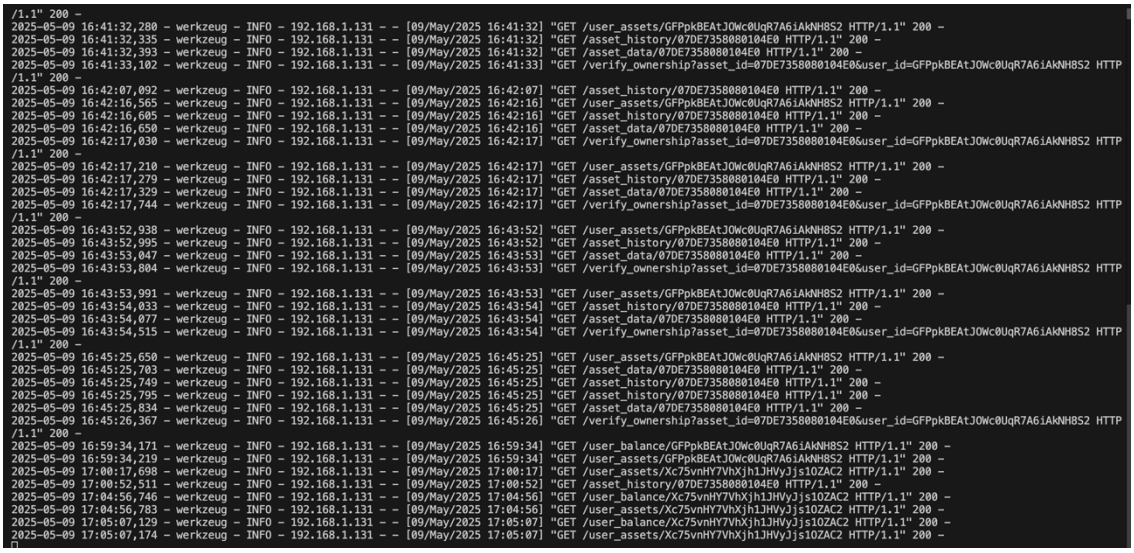


Imagen 36 – Eventos en API de Blockchain

- Acceso controlado por rol (Imagen 37):

    - Cada petición enviada al backend incluye información de sesión.
    - El backend verifica que los permisos del usuario (usuario, fabricante, admin o guest) correspondan con la operación solicitada.
    - Intentos de acceso indebido (por ejemplo, intentar registrar sin ser fabricante) son denegados correctamente con respuestas HTTP 403.

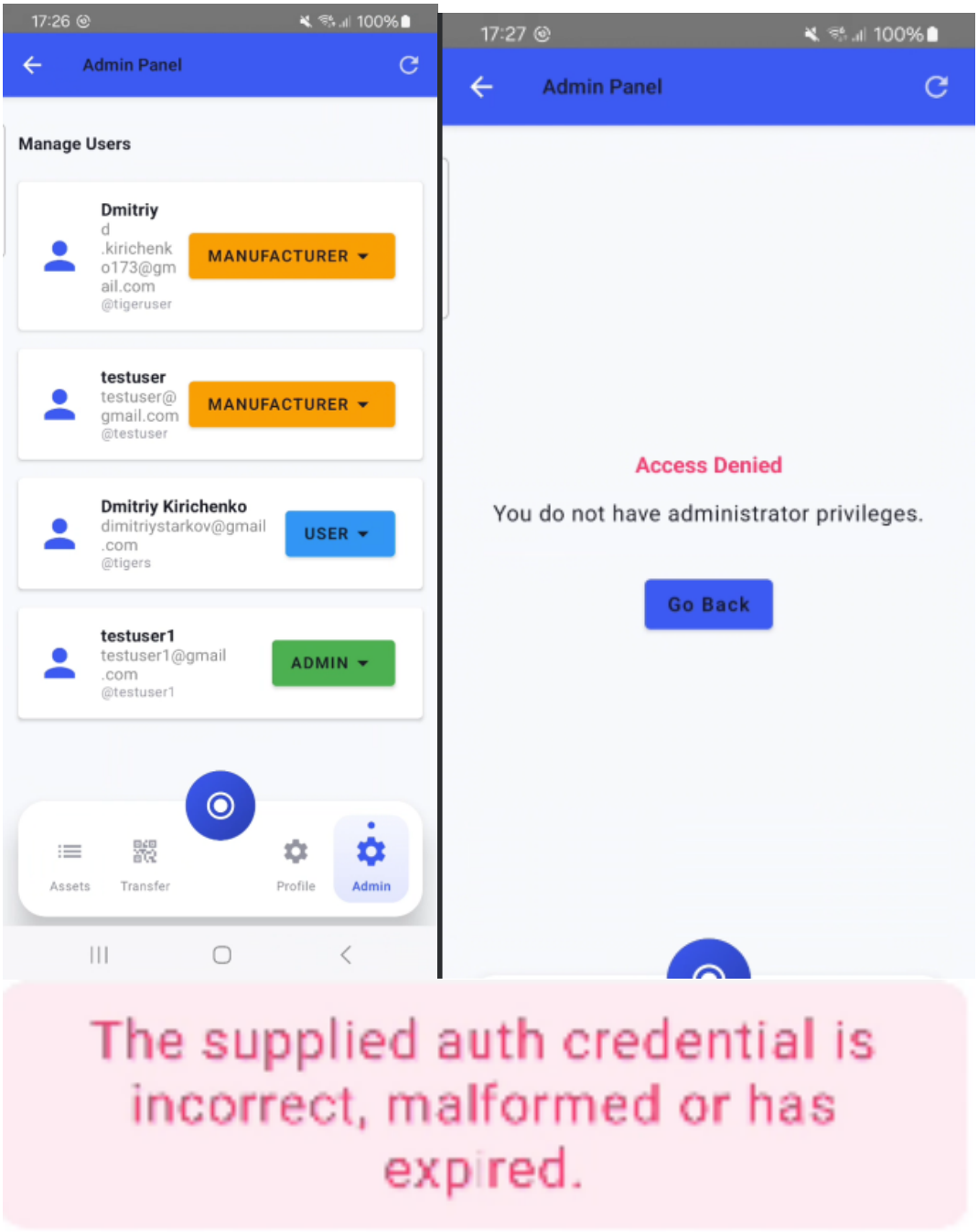


Imagen 37 – Demostración visual de control de acceso

- Respuestas rápidas y manejo de errores (Imagen 38)

    - La app ha sido validada frente a múltiples escenarios de fallo (token expirado, ID inválido, pérdida de red), y en todos los casos el backend responde con códigos de estado y mensajes controlados.
    - Las respuestas HTTP incluyen estructuras JSON normalizadas, fácilmente interpretable por los componentes cliente.

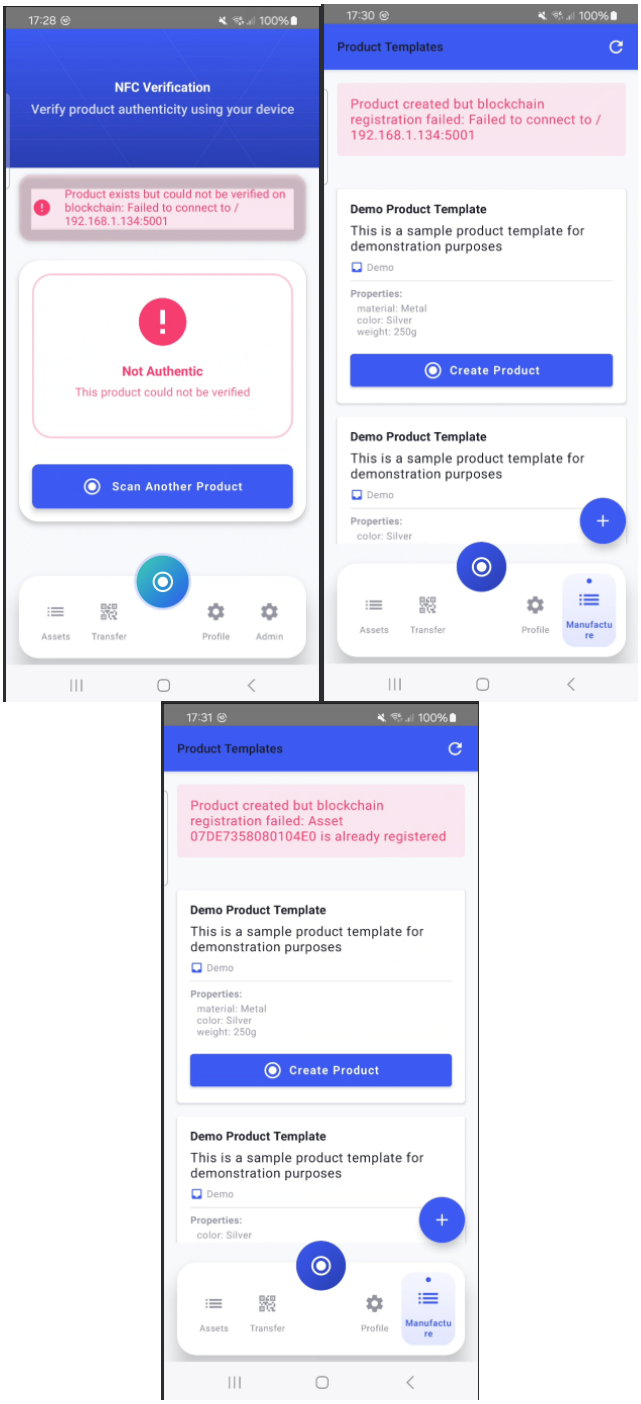


Imagen 38 – Demostración de errores a lo largo de la APP

- Sincronización de datos (Imagen 39)

    - Las operaciones exitosas (registro, transferencia) generan efectos tanto en la blockchain como en Firestore (para almacenamiento auxiliar).
    - La app puede acceder de forma unificada a datos distribuidos entre ambos sistemas, sin incoherencias ni duplicidades.

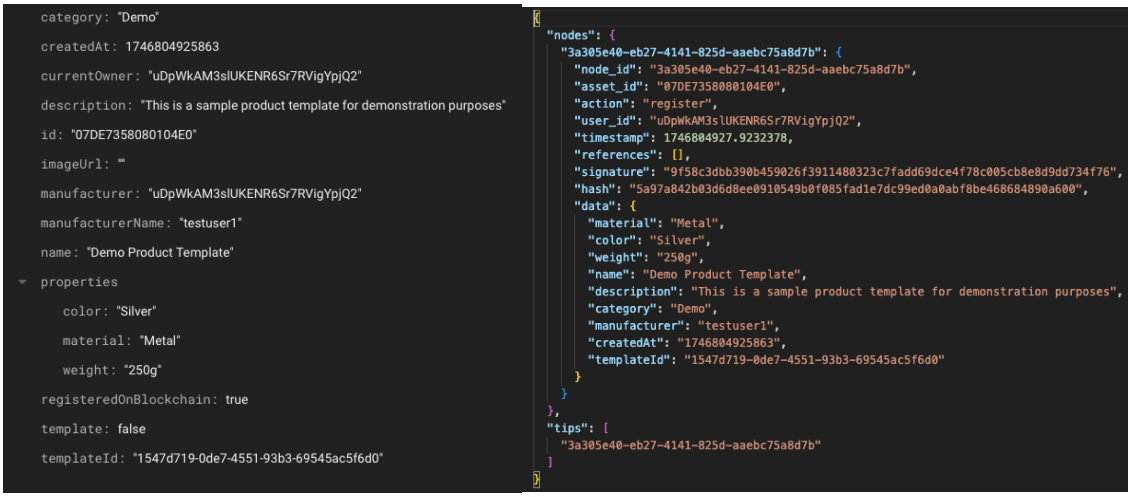


Imagen 39 – Sincronización de datos de forma Auxiliar

Conclusión técnica

La integración entre cliente y servidor ha demostrado ser estable, predecible y segura, validando todos los puntos de comunicación críticos del sistema. Esta interacción confiable entre capas garantiza que el sistema pueda operar en condiciones reales con múltiples usuarios, sin comprometer ni la seguridad de los datos ni la experiencia del usuario.

6.6 Rendimiento: análisis de uso de recursos en dispositivo

El comportamiento del sistema InLock en términos de eficiencia computacional ha sido evaluado mediante pruebas reales en entorno Android (en dispositivo físico Samsung S21 Ultra) utilizando el profiler de Android Studio. El análisis se centró en dos métricas clave: uso de CPU y uso de memoria RAM, tanto en situaciones normales como bajo carga específica.

Consumo promedio

Durante el funcionamiento regular de la aplicación - navegación, visualización de productos y operaciones básicas de lectura NFC - el sistema mostró un uso medio de CPU y memoria en torno al 5 % (Imagen 41), lo que indica una huella de rendimiento muy baja y adecuada para dispositivos de gama media e incluso baja.

Pico de carga: escaneo NFC con animación

El momento de mayor carga detectado se produjo al activar el modo de escaneo NFC con animaciones complejas en la interfaz de usuario. En esta situación, se registró un pico puntual de:

- 21,8 % de uso de CPU (Imagen 40).
- Incremento temporal de consumo de memoria, sin alcanzar niveles críticos.

Este comportamiento es esperable, ya que se combinan dos tareas exigentes:

1. Lectura activa del campo NFC mediante el sensor del dispositivo.
2. Renderizado de una animación por capas en Compose, con efectos dinámicos visuales.

A pesar de ello, el uso promedio durante toda la operación se mantuvo en torno al 13 %, lo cual sigue siendo perfectamente aceptable y funcional para la mayoría de los dispositivos móviles modernos.

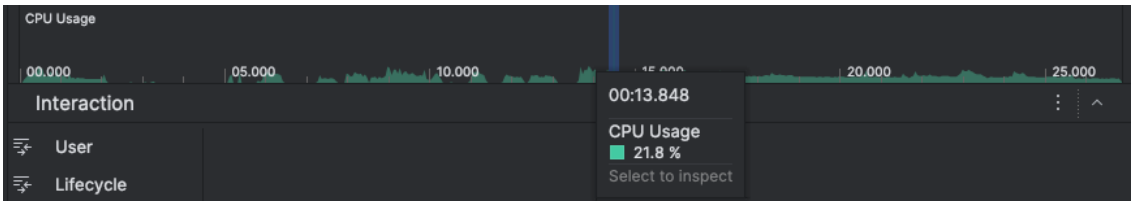


Imagen 40 – Consumo de CPU en momento mas cargado

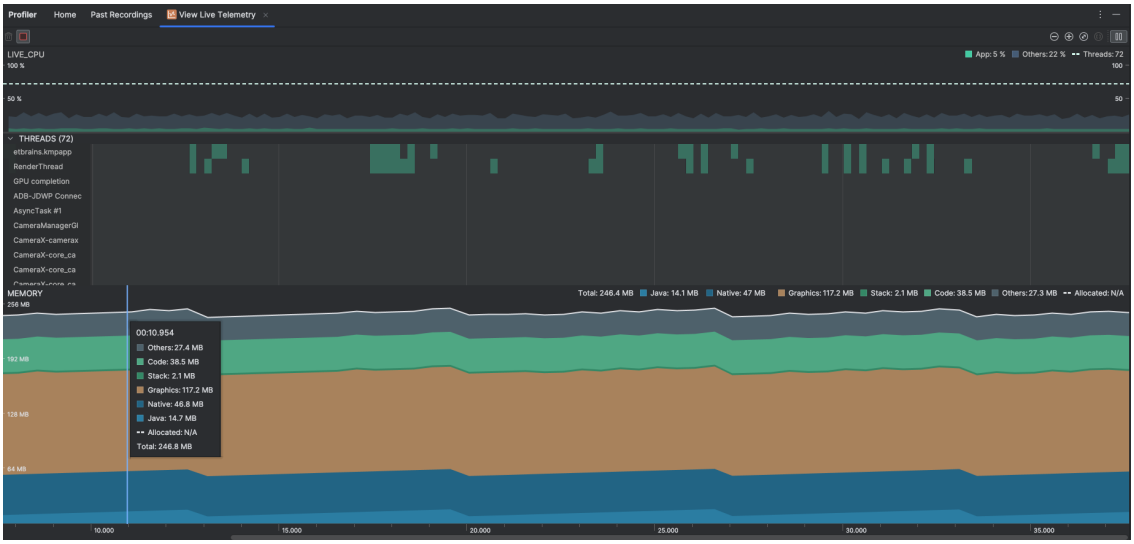


Imagen 41 – Consumo de memoria promedio durante el uso de la APP

Conclusión y margen de mejora

Los resultados obtenidos confirman que la arquitectura del sistema es ligera, estable y operativamente viable desde el punto de vista de consumo de recursos. No se detectaron pérdidas de rendimiento significativas, ralentizaciones o bloqueos en ningún momento de la prueba.

No obstante, existe un amplio margen de mejora en la optimización de procesos gráficos, especialmente durante la fase de escaneo. Entre las posibles acciones futuras destacan:

- Reducción del número de recomposiciones en animaciones Compose.
- Simplificación de efectos visuales en dispositivos de gama baja.
- Desacoplamiento parcial de renderizado y operaciones NFC mediante coroutines específicas con Dispatchers.IO.

Este análisis demuestra que el sistema puede ejecutarse de forma fluida incluso en condiciones exigentes, garantizando una buena experiencia de usuario sin comprometer la autonomía ni la estabilidad del dispositivo.

6.7 Evaluación del cumplimiento de objetivos

A lo largo del desarrollo del proyecto, se han alcanzado la mayoría de los objetivos específicos definidos en la etapa inicial, tanto a nivel técnico como funcional. A continuación, se presenta una evaluación individualizada de cada uno de ellos, con una breve reflexión sobre su grado de cumplimiento (Tabla 4):

Objetivo específico	Cumplimiento	Comentario técnico
Diseñar una arquitectura de identificación segura basada en etiquetas RFID de escritura única	Cumplido	Se ha utilizado tecnología NFC compatible (NTAG 424 DNA) con UID único, no modificable.
Desarrollar una aplicación móvil multiplataforma para la gestión y verificación de activos	Cumplido	Aplicación desarrollada con Kotlin Multiplatform; incluye lectura NFC, escaneo QR y control por roles.
Implementar una infraestructura blockchain basada en DAG para el registro descentralizado de activos	Cumplido	Se ha desarrollado un backend funcional con DAG en Python y nodos independientes coordinados por API.
Permitir la transferencia segura de activos entre usuarios con validación física	Cumplido	Transferencias implementadas con validación NFC + escaneo QR, confirmación criptográfica y control de sesión.
Diseñar un modelo básico de incentivos digitales por posesión legítima de bienes	Parcial	El modelo ha sido definido conceptualmente, pero su implementación práctica ha sido descartada en esta versión.
Establecer mecanismos de seguridad, sincronización y verificación de integridad	Cumplido	Implementados flujos de autenticación Firebase, validación DAG, sincronización Firestore con el blockchain.

Tabla 4 – Evaluación de cumplimiento de objetivos

Como puede observarse, los objetivos fundamentales del sistema se han cumplido satisfactoriamente, tanto en su definición técnica como en su implementación funcional. La única excepción corresponde al objetivo vinculado a la generación de tokens por posesión, que ha sido **descartado en esta versión final** para centrar el desarrollo en aspectos de trazabilidad, verificación y seguridad. Aun así, el modelo de incentivos permanece documentado como una posible línea de desarrollo futura.

7. Discusión

El sistema InLock representa un avance significativo en la integración de tecnologías físicas y digitales para la trazabilidad y protección de bienes. Su arquitectura modular, centrada en una red blockchain tipo DAG, NFC, una app multiplataforma y un backend seguro, permite validar activos físicos mediante escaneos reales, registrar su propiedad y realizar transferencias seguras con validación física.

Sin embargo, desde un punto de vista de ingeniería avanzada, la solución actual debe entenderse como un prototipo funcional que ha priorizado demostrar la viabilidad técnica de los flujos principales. La validación de conceptos ha sido exitosa, pero se identifican áreas de mejora y crecimiento clave, que permitirían llevar InLock hacia una plataforma sólida, escalable y profesional para entornos reales.

7.1 Ciberseguridad y gestión avanzada de claves

Uno de los aspectos más sensibles en cualquier sistema de trazabilidad de propiedad es la custodia de las claves criptográficas. Actualmente, la firma digital de nodos en la DAG se realiza con funciones hash no asociadas a claves privadas robustas. En un entorno profesional, la protección de la identidad digital del usuario requeriría el uso de sistemas como Firebase App Check, HSMs (Hardware Security Modules) o incluso Ledger para generar y guardar claves localmente de forma cifrada.

A medio plazo, también es recomendable la incorporación de sistemas de firma post-cuántica, dada la progresiva amenaza de ordenadores cuánticos a los sistemas criptográficos actuales.

7.2 Escalabilidad y arquitectura distribuida

La actual implementación en Flask y almacenamiento en archivos.json es idónea como PoC (Proof of Concept), pero limita la expansión en volumen y concurrencia. Para alcanzar un despliegue productivo, el backend podría migrarse a una arquitectura basada en:

- Kubernetes, para orquestación de contenedores y alta disponibilidad.
- Apache Kafka, como sistema de colas distribuido para desacoplar flujos como staking, verificación, y transferencias.
- Base de datos orientada a grafos, como Neo4j o ArangoDB, para una persistencia escalable del DAG.
- Cloudflare y WAF para proteger el backend ante ataques de denegación de servicio (DDoS) y tráfico anómalo

Estas mejoras no solo protegerían la integridad y disponibilidad del sistema, sino que permitirían una auditoría detallada en tiempo real.

7.3 Interoperabilidad y federación de datos

El modelo actual depende de un backend centralizado y una sola fuente de verdad. Una evolución natural sería establecer un sistema federado o descentralizado, donde distintos nodos (fabricantes, autoridades, usuarios) tengan su propia instancia, replicando y validando datos de manera consensuada.

Esto permitiría:

- Mayor resistencia a fallos.
- Reducción de dependencia de un solo backend.
- Interoperabilidad con sistemas de terceros (policía, aseguradoras, aduanas).

7.4 Inteligencia Artificial para análisis de comportamiento

Una línea futura de alto valor sería la incorporación de modelos de IA para la detección de anomalías o fraudes en el sistema. Mediante aprendizaje supervisado o no supervisado, se podría:

- Detectar patrones sospechosos de uso (p. ej., múltiples escaneos en lugares distantes).
- Estimar la fiabilidad de transacciones.
- Predecir rutas de mercado negro.
- Identificar redes de falsificación.

Esto requeriría un módulo adicional de análisis en la nube, conectado a logs y eventos del DAG, pero sería una herramienta poderosa para prevenir delitos y mejorar la experiencia de los usuarios.

7.5 Wallets, economía digital y modelos tokenizados

Aunque la lógica de staking está definida, no se ha implementado aún una wallet gráfica para el usuario. Esto limita la funcionalidad económica del sistema. Para llevarlo a un nivel más completo, se puede:

- Definir una tokenomía clara: reglas de emisión, inflación, quema y transferencia.
- Implementar wallets móviles encriptadas.
- Añadir lógica de “valor relativo” al staking: no todos los bienes valen igual.

Además, un sistema de reputación descentralizada (como Web of Trust) permitiría premiar a fabricantes fiables y penalizar a los actores maliciosos.

7.6 Participación energética mínima y edge computing

Una característica diferencial del modelo DAG es que los dispositivos móviles (clientes) participan en la verificación mediante acciones como staking o lectura NFC. Es importante garantizar que dicha participación se mantenga en un rango energético bajo, haciendo uso de:

- Threaded coroutines con Dispatchers.IO.
- Algoritmos de compresión de metadatos.
- Cacheo optimizado con SQLite o SQLDelight.

Además, permitir que parte del DAG se almacene en la base de datos local del dispositivo contribuye a la resiliencia general y la descentralización del sistema.

7.7 Ética y gobernanza descentralizada

En sistemas que combinan bienes físicos, economía tokenizada y usuarios anónimos, es imprescindible considerar aspectos éticos y de gobernanza. InLock podría explorar:

- Creación de un consejo validador de marcas (Proof of Authority).
- Reglas comunitarias para validadores (Proof of Stake delegada).
- Trazabilidad de decisiones en la red, registradas también en DAG.

En conclusión, la versión actual del sistema demuestra con éxito la viabilidad técnica de un ecosistema de trazabilidad física-digital. Pero la proyección profesional del proyecto requiere incorporar capas adicionales de seguridad, escalabilidad, interoperabilidad y análisis inteligente, lo cual no solo es viable, sino necesario para que InLock se consolide como una solución de referencia en entornos reales y críticos.

8. Futuras líneas de trabajo

El prototipo desarrollado ha permitido validar con éxito los fundamentos técnicos y operativos del sistema InLock. No obstante, para consolidarlo como una solución plenamente funcional en contextos reales y con capacidad de expansión, es necesario abordar una serie de líneas de trabajo complementarias. Estas mejoras abarcan aspectos técnicos, económicos, de seguridad, usabilidad e incluso éticos, y servirán como base para la evolución del proyecto.

8.1 Implementación de sistema criptográfico completo

- Desarrollo de un módulo de gestión de claves criptográficas seguras, mediante el uso de Firebase App Check, HSMs (Hardware Security Modules), o dispositivos físicos como Ledger.
- Sustitución de la firma actual hash por firmas digitales ECDSA o algoritmos post-cuánticos.
- Validación de chip RFID avanzado (NTAG 424 DNA) mediante lectura criptográfica en el móvil.

8.2 Arquitectura escalable y distribuida

- Reimplementación del backend actual en un entorno más robusto, empleando Golang + PostgreSQL/Neo4j.
- Contenerización completa mediante Docker + Kubernetes, permitiendo escalabilidad horizontal.
- Integración de Apache Kafka como capa intermedia de eventos asincrónicos (staking, verificación, etc.).
- Reestructuración del DAG en base de datos orientada a grafos.

8.3 Wallet integrada y tokenomía completa

- Desarrollo de una wallet móvil cifrada, integrada en la app actual, para gestionar activos y tokens.
- Definición de una economía digital interna basada en tokens: emisión controlada, reglas de staking, incentivos y penalizaciones.
- Interoperabilidad con redes públicas como Polygon o Solana en versiones futuras.

8.4 Análisis inteligente y prevención de fraude

- Incorporación de módulos de IA para detección de patrones anómalos, validación de comportamientos atípicos y predicción de acciones fraudulentas.
- Registro y análisis de eventos del DAG en tiempo real mediante técnicas de machine learning.
- Creación de una consola de administración con visualización de nodos y estadísticas.

8.5 Gobernanza descentralizada

- Implementación de un sistema de gobernanza distribuida para fabricantes, usuarios y validadores.
- Inclusión de mecanismos de votación, reputación y toma de decisiones comunitarias sobre cambios de protocolo.

8.6 Interoperabilidad institucional

- Establecimiento de integraciones con bases de datos de fuerzas de seguridad, aseguradoras o aduanas, para permitir la verificación oficial de activos robados o recuperados.
- Definición de APIs públicas para la consulta oficial de bienes desde entornos institucionales.

8.7 Mejora de usabilidad y experiencia de usuario

- Rediseño de la interfaz gráfica para mejorar accesibilidad, claridad y respuesta visual.
- Simplificación del flujo de escaneo, transferencia y staking para usuarios sin experiencia técnica.
- Inclusión de accesibilidad para personas con discapacidades visuales o motrices.

8.8 Validación real en campo

- Organización de pruebas piloto con usuarios reales, fabricantes y comercios.
- Evaluación cualitativa y cuantitativa del sistema en escenarios prácticos.
- Recogida de feedback directo para orientar decisiones de desarrollo futuro.

Estas líneas de trabajo definen una evolución clara y progresiva del sistema hacia un producto de alto nivel tecnológico, capaz de integrarse en contextos reales, escalar a nivel global y contribuir significativamente a la trazabilidad, protección y certificación de bienes físicos en la era digital.

9. Conclusiones

El desarrollo del sistema InLock ha permitido materializar un modelo funcional, seguro y tecnológicamente avanzado para la trazabilidad y protección de activos físicos. A través de la integración estratégica de tecnologías como RFID de escritura única, lectura NFC distribuida, blockchain basada en DAG, autenticación por Firebase, y una aplicación móvil multiplataforma, se ha demostrado que es posible establecer una infraestructura robusta y accesible para vincular la realidad física con la seguridad digital.

Este trabajo no solo cumple los objetivos propuestos, sino que los supera al introducir un modelo innovador donde la propiedad de bienes tangibles puede ser verificada, transferida y protegida de manera completamente descentralizada y sin intermediarios. La combinación de verificación física (escaneo NFC), lógica distribuida (DAG), e identidad digital fuerte (JWT, control de roles, claves únicas), convierte a InLock en una solución única con un potencial real de impacto en sectores como el lujo, la electrónica, el arte, la logística y la seguridad pública.

Durante el desarrollo se ha validado no solo la viabilidad técnica del sistema, sino también su estabilidad operativa, eficiencia energética y capacidad de ejecución en dispositivos reales. Las pruebas de rendimiento han demostrado que InLock puede funcionar con un consumo mínimo de recursos, permitiendo incluso que dispositivos móviles operen como nodos validadores ligeros, contribuyendo así a una red más resiliente y escalable.

Además, el modelo de participación flexible - con soporte para usuarios registrados y guest, y la transparencia en la verificación pública del estado de un bien, sientan las bases para un nuevo paradigma en la trazabilidad física-digital: uno que no solo previene la falsificación y el robo, sino que empodera al usuario final, crea economías basadas en la posesión legítima y facilita la cooperación entre ciudadanos, marcas y autoridades.

Desde una perspectiva de ingeniería, este proyecto también constituye una plataforma de experimentación avanzada donde pueden converger tecnologías emergentes como inteligencia artificial aplicada a seguridad, firmas post-cuánticas, edge computing o gobernanza descentralizada basada en reputación.

En definitiva, InLock es mucho más que una prueba de concepto: es un ecosistema en evolución, sólido en su base y ambicioso en su visión, que demuestra que la protección de lo físico a través de lo digital es no solo posible, sino imprescindible en el mundo actual y aun más en el mundo de futuro.

Referencias Bibliográficas

OECD/EUIPO. (2019). Trends in Trade in Counterfeit and Pirated Goods. OECD Publishing. https://doi.org/10.1787/g2g9f533-en

ONU. (2015). Objetivos de Desarrollo Sostenible. https://sdgs.un.org/goals

Interpol-WCO. (2020). Illicit Trade Report 2020. World Customs Organization.

European Union Agency for Law Enforcement Training. (2022). Annual Report 2022. CEPOL. https://www.cepol.europa.eu/publications/annual-reports

UNIDO. (2021). Traceability Systems for Enhanced Consumer Safety and Trade Facilitation. United Nations Industrial Development Organization. https://www.unido.org/resources/publications

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf Deloitte.

(2021). 2021 Global Blockchain Survey. Deloitte Insights. https://www2.deloitte.com/global/en/pages/consulting/articles/global-blockchain-survey.html

Allied Market Research. (2022). RFID Market by Product Type, Frequency, and Application: Global Opportunity Analysis and Industry Forecast, 2022–2031. https://www.alliedmarketresearch.com/rfid-market

Chronicled. (2021). Blockchain Solutions for Supply Chain Integrity. Chronicled Inc. https://www.chronicled.com/

McKinsey & Company. (2022). Blockchain Beyond the Hype: What is the Strategic Business Value? https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/blockchain-beyond-the-hype

Pi Network. (2022). White Paper: Building a Mobile-First Cryptocurrency. Pi Core Team. https://minepi.com/white-paper

VeChain Foundation. (2022). VeChain Whitepaper 2.0: Blockchain for Real-World Business. https://www.vechain.org/

International Organization for Standardization. (2001). ISO/IEC 14443: Identification cards — Contactless integrated circuit cards — Proximity cards. ISO. https://www.iso.org/standard/73594.html

International Organization for Standardization. (2006). ISO/IEC 15693: Identification cards — Contactless integrated circuit cards — Vicinity cards. ISO. https://www.iso.org/standard/39647.html

NFC Forum. (2014). NFC Data Exchange Format (NDEF) Technical Specification. NFC Forum. https://nfc-forum.org/our-work/specifications/specifications-archive/ndef-technical-specification/

NXP Semiconductors. (2019). NTAG 424 DNA and NTAG 424 DNA TagTamper - Product data sheet. https://www.nxp.com/docs/en/data-sheet/NTAG424DNA.pdf

Burns, B. (2016). Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media.

Cockburn, A. (2005). Hexagonal Architecture. Alistair Cockburn's Articles. http://alistair.cockburn.us/Hexagonal+architecture

Firebase. (2023). Firebase Authentication Documentation. Google Firebase. https://firebase.google.com/docs/auth

JetBrains. (2023). Ktor Documentation. JetBrains. https://ktor.io/

Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal, 2014(239), 2.

Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446. https://datatracker.ietf.org/doc/html/rfc8446

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-Based Access Control Models. IEEE Computer, 29(2), 38-47. https://doi.org/10.1109/2.485845

Donovan, A. A., & Kernighan, B. W. (2015). The Go Programming Language. Addison-Wesley Professional.

Popov, S. (2018). The Tangle. IOTA Foundation. https://www.iota.org/research/academic-papers


Universidad
Europea VALENCIA

https://github.com/tigershawt/InLock-TFG-Primitive-Version-Prototype/tree/main/composeApp/src