Universitat Oberta de Catalunya

uoc.edu

Trazabilidad de residuos plásticos

Amador Jaume Barceló
Máster en Ciberseguridad y Privacidad
Sistemas de Blockchain

Nombre Tutor/a de TF: José Luis de la Rosa Esteva
Nombre Profesor/a responsable de la asignatura: Víctor García Font

Fecha Entrega: enero de 2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Trazabilidad de residuos plásticos*
Nombre del autor:	*Amador Jaume Barceló*
Nombre del consultor/a:	*José Luis de la Rosa Esteva*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*01/2023*
Titulación o programa:	*Programa de estudios*
Área del Trabajo Final:	*Sistemas de blockchain*
Idioma del trabajo:	*Castellano*
Palabras clave	*Blockchain, trazabilidad, residuos*

Resumen del Trabajo

La trazabilidad es un aspecto fundamental en la gestión de residuos. El uso de la tecnología blockchain puede facilitar el mantenimiento de un registro descentralizado, inmutable y transparente de los procesos por los que pasa a cada lote de residuos.

Este proyecto tiene como objetivo diseñar e implementar una prueba de concepto del caso de uso de trazabilidad de residuos plásticos. La implementación se realiza sobre la blockchain privada y permisionada de Hyperledger Fabric para proporcionar más privacidad y evitar la alta volatibilidad de latencias y costes de transacción en blockchains públicas como Ethereum.

La metodología seguida para completar este proyecto es la metodología del prototipado con la finalidad de disponer de un prototipo funcional lo antes posible y tener más flexibilidad para aplicar modificaciones correctivas o perfectivas.

Finalmente, se realiza un análisis de la seguridad, la privacidad y la viabilidad de la solución propuesta.

Abstract

Traceability is a fundamental aspect in waste management. The use of blockchain technology can facilitate the maintenance of a decentralized, immutable and transparent record of the processes that each waste batch goes through.

This project aims to design and implement a proof of concept of the plastic waste traceability use case. The implementation is done on the private and permissioned Hyperledger Fabric blockchain in order to provide more privacy and avoid the high volatility of latencies and transaction costs in public blockchains such as Ethereum.

The methodology followed to complete this project is the prototyping methodology in order to have a functional prototype as soon as possible and have more flexibility to apply corrective or perfective modifications.

Finally, an analysis of the security, the privacy and the viability of the proposed solution is performed.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1. Contexto y justificación del Trabajo

El plástico es un material ampliamente utilizado en diferentes sectores, especialmente para el empaquetado y embalaje tanto comercial como industrial de productos. Plastics Europe [1] menciona que se consumieron 49’1 megatoneladas (Mt) y se generaron 29’5 Mt de residuos plásticos durante el año 2020 en los países de la Unión Europea. De estos residuos solo se recicló el 34’6%, mientras que el 23’4% terminó depositado en vertederos.

Esta situación ha provocado que surgieran diferentes iniciativas con el objetivo de mejorar la gestión de los residuos plásticos. Un aspecto fundamental para efectuar esta gestión de forma adecuada y eficiente es mantener un registro de los procesos que experimentan los residuos plásticos a lo largo de su ciclo de gestión, desde su generación hasta su reciclaje o eliminación. Además de los procesos, es importante identificar a todos los actores y sus respectivas responsabilidades.

La gestión de residuos plásticos incluye su generación, recogida, clasificación y reciclaje, aparte del proceso de fabricación de nuevos productos a partir del plástico reciclado. Asimismo, existen múltiples actores que intervienen en alguno de estos procesos, como los generadores de residuos (por ejemplo, hogares, fábricas o comercios), recolectores de residuos, clasificadores, transportistas, plantas de reciclaje, fabricantes de nuevos productos plásticos y autoridades de regulación, entre otros [2].

Para abordar el problema de la trazabilidad en la gestión de residuos plásticos se puede hacer uso de la blockchain. Esta tecnología emergente enfocada en la descentralización ofrece tolerancia a fallas, robustez, inmutabilidad y transparencia. Estas propiedades son muy útiles en la detección de fraudes.

No obstante, las blockchains públicas como Ethereum presentan una alta volatibilidad en la latencia, el rendimiento, el coste de las transacciones y el precio de las criptomonedas. Estos problemas se reducen significativamente en las blockchains privadas como Hyperledger Fabric, que además proporcionan un mayor nivel de privacidad porque solo permiten el acceso a la red a los participantes autorizados.

Un caso de uso de la tecnología blockchain es la tokenización, es decir, la representación de activos en formato digital mediante tokens. En particular,

los tokens no fungibles (NFT) permiten representar activos únicos y pueden implementarse mediante contratos inteligentes siguiendo el estándar ERC-721.

1.2. Objetivos del Trabajo

El objetivo general de este trabajo de fin de máster (TFM) es desarrollar una prueba de concepto (PoC) para el caso de uso de trazabilidad de residuos plásticos que aproveche las propiedades de la tecnología blockchain descritas en la sección anterior.

Los objetivos específicos de este trabajo son:

- Diseñar un sistema seguro para la trazabilidad de residuos plásticos a lo largo de su gestión.
- Implementar una PoC basada en el diseño elaborado.
- Analizar la seguridad y la privacidad que garantiza la PoC.
- Analizar la viabilidad de la PoC en términos económicos y de rendimiento.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Este trabajo contribuye positivamente a la sostenibilidad medioambiental y a la reducción de la huella ecológica. Desde el punto de vista de los Objetivos de Desarrollo Sostenible (ODS) de la ONU para el año 2030, impacta de forma positiva en los siguientes objetivos:

- ODS 11 – Ciudades y comunidades sostenibles
- ODS 13 – Cambio climático
- ODS 14 – Vida bajo el agua
- ODS 15 – Vida en tierra

Como ya se ha mencionado en la introducción, la mayoría de los residuos plásticos no se reciclan y una parte significativa de éstos se aboca en vertederos. A esto, hay que añadir los vertidos de residuos plásticos que llegan a los mares y océanos y que perjudican gravemente a los ecosistemas marinos.

El sistema de trazabilidad que se desea implementar busca facilitar el seguimiento de cada residuo plástico y su evolución. Por lo tanto, puede ayudar a incrementar la eficiencia y la eficacia de los diferentes procesos a lo largo del ciclo de gestión de estos residuos. De esta forma, este trabajo puede contribuir a impulsar la economía circular del plástico, porque una mejor gestión de los residuos puede aumentar la cantidad de residuos plásticos reciclados.

No obstante, el uso de la tecnología blockchain genera un mayor consumo energético respecto a soluciones centralizadas. A pesar de esto, las blockchains privadas y permisionadas como Hyperledger Fabric presentan un consumo energético aproximado de 1 J por transacción, muy inferior al de las blockchains públicas como Ethereum, que se sitúa en 10^3 J por transacción si utilizan el protocolo de consenso Proof of Stake (PoS) o 10^9 J por transacción si utilizan Proof of Work (PoW) [3].

En cuanto al impacto ético y social, disponer de un registro inmutable de todos los procesos que experimenta un residuo plástico permite identificar y detectar las actuaciones indebidas o fraudulentes que un determinado actor haya realizado. Sin embargo, este registro también puede revelar datos personales y datos sensibles de los diferentes actores. Para mitigar este riesgo, el objetivo es minimizar los datos que se publican en este registro.

1.4. Enfoque y método seguido

La metodología elegida para elaborar este trabajo es la de prototipado. El objetivo es obtener un prototipo funcional en la primera entrega de seguimiento (PEC2) y, partiendo de este prototipo, añadir el resto de la funcionalidad para la segunda entrega de seguimiento (PEC3). De esta forma, se garantiza que la tecnología blockchain escogida, así como las demás librerías, posibilitan la implementación del sistema diseñado. Otra ventaja de utilizar esta metodología es disponer de margen temporal suficiente para aplicar correcciones y modificaciones.

1.5. Planificación del Trabajo

La tabla 1 contiene la planificación de las tareas necesarias para completar este proyecto y las entregas programadas. El diagrama de Gannt se muestra en la figura 1.

Nombre	Fecha de inicio	Fecha de fin	Duración
TFM – Trazabilidad de residuos plásticos	28/09/22	26/01/23	87
1. Plan de trabajo	28/09/22	10/10/22	9
1.1. Elaboración del Plan de Trabajo	28/09/22	10/10/22	9
Entrega PEC1	11/10/22	11/10/22	0

2. Diseño inicial y desarrollo del prototipo	11/10/22	7/11/22	20
2.1. Definición de entidades y requisitos	11/10/22	13/10/22	3
2.2. Modelado del sistema	14/10/22	24/10/22	7
2.2.1. Diseño de la arquitectura	14/10/22	18/10/22	3
2.2.2. Diagrama de casos de uso	17/10/22	18/10/22	2
2.2.3. Diagramas de clases	19/10/22	20/10/22	2
2.2.4. Diagramas de secuencia	19/10/22	24/10/22	4
2.3. Desarrollo del prototipo	19/10/22	03/11/22	12
2.3.1. Configuración de la red blockchain	19/10/22	21/10/22	3
2.3.2. Implementar y testear el prototipo	21/10/22	03/11/22	10
2.4. Documentación	19/10/22	07/11/22	14
2.4.1. Redacción de la documentación	19/10/22	03/11/22	12
2.4.2. Preparación de la entrega de seguimiento	04/11/22	07/11/22	2
Entrega PEC2	08/11/22	08/11/22	0
3. Diseño final y desarrollo de la PoC	08/11/22	5/12/22	20
3.1. Feedback y correcciones	08/11/22	15/11/22	6
3.1.1. Documentar feedback recibido	08/11/22	11/11/22	4
3.1.2. Corregir errores del diseño	08/11/22	11/11/22	4
3.1.3. Corregir vulnerabilidades del prototipo	08/11/22	15/11/22	6
3.2. Implementación de la PoC	14/11/22	25/11/22	10
3.2.1. Implementar funcionalidades restantes	14/11/22	25/11/22	10
3.2.2. Implementar y testear front-end	14/11/22	25/11/22	10
3.3. Análisis de seguridad y privacidad	28/11/22	05/12/22	6
3.4. Análisis de viabilidad	28/11/22	05/12/22	6
Entrega PEC3	06/12/22	06/12/22	0
4. Memoria y presentación	06/12/22	26/01/23	38
4.1. Redacción de la memoria	06/12/22	09/01/23	25

4.1.1. Extraer documentación relevante	06/12/22	09/12/22	4
4.1.2. Redactar la memoria	09/12/22	09/01/23	22
Entrega PEC4 - Memoria final	10/01/23	10/01/23	0
4.2. Elaboración de la presentación	10/01/23	16/01/23	5
Entrega Presentación en vídeo	17/01/23	17/01/23	0
4.3. Preparación de la defensa	17/01/23	26/01/23	8

Tabla 1. Planificación del proyecto

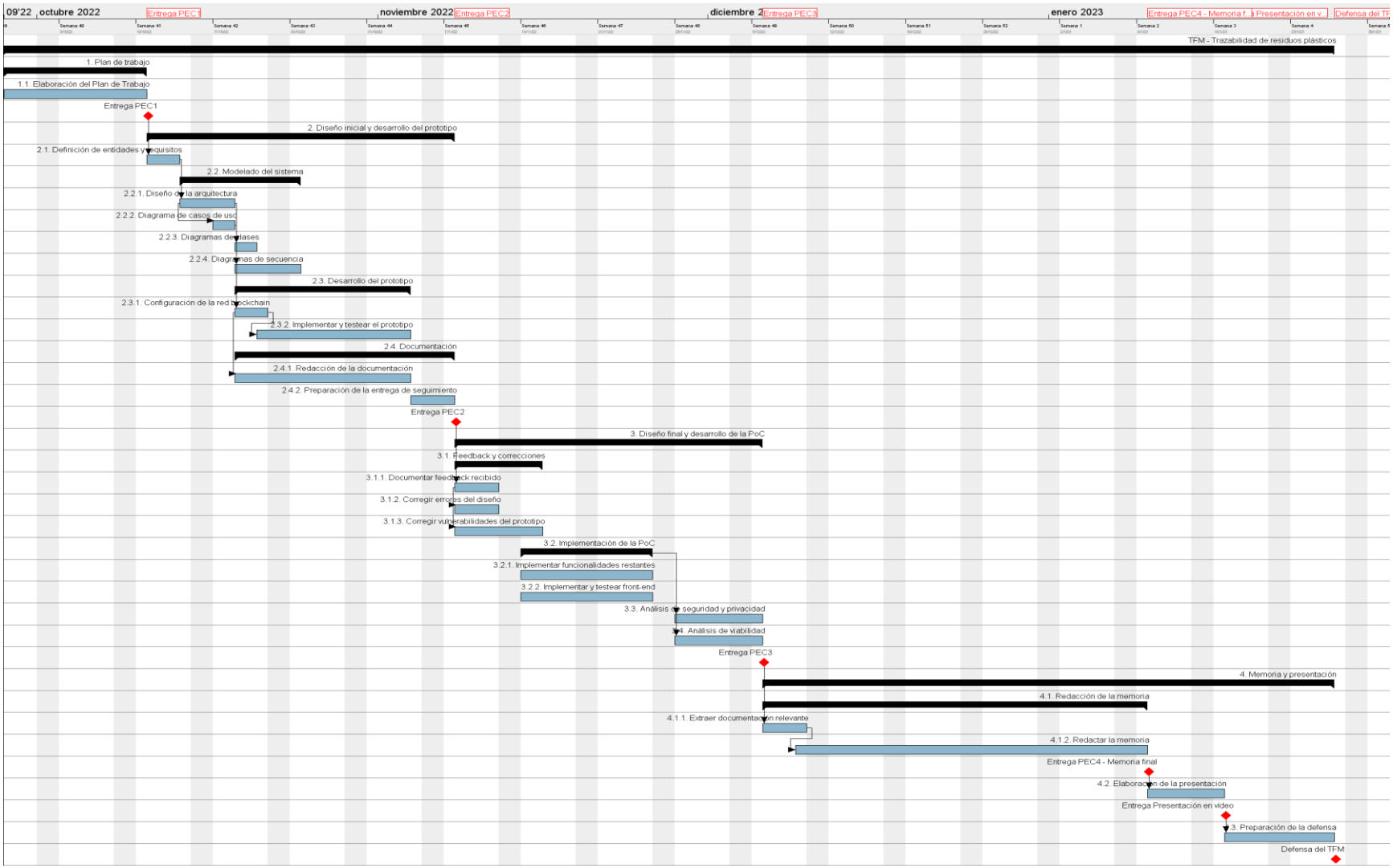


Figura 1. Diagrama de Gannt

1.6. Breve sumario de productos obtenidos

Los productos obtenidos son la memoria final del TFM y la PoC del caso de uso de trazabilidad de residuos plásticos, que incluye la configuración de la red de Hyperledger Fabric, los contratos inteligentes desarrollados y el código fuente y el ejecutable de la aplicación de escritorio.

1.7. Breve descripción de los otros capítulos de la memoria

El resto del documento contiene siete capítulos más. En el capítulo 2 se revisan las propuestas relacionadas con la gestión de residuos que utilizan blockchain.

En el capítulo 3 se explican los conceptos teóricos necesarios para la comprensión del desarrollo de la PoC. Entre estos conceptos, se incluyen los fundamentos de la blockchain utilizada, que es Hyperledger Fabric.

En los siguientes capítulos se presenta el desarrollo de la PoC. En el capítulo 4 se detalla el diseño del sistema. En este capítulo se incluye la definición de las entidades y los requisitos que se han considerado, así como la arquitectura y los diagramas de casos de uso, de clases y de secuencia. En el capítulo 5 se describe la configuración de la red y la implementación de los contratos inteligentes en base al diseño efectuado en el capítulo anterior.

En el capítulo 6 se muestra un ejemplo del funcionamiento de la aplicación de escritorio. En el capítulo 7 se expone el análisis de la solución desarrollada, tanto a nivel de seguridad y privacidad como de viabilidad. Finalmente, en el capítulo 8 se incluyen las conclusiones del proyecto.

2. Estado del arte

Existen diferentes proyectos que utilizan la tecnología blockchain para la gestión de residuos. Lenz et al. [4] analizaron veinte de estos proyectos y los categorizaron en función del grupo de actores involucrados (empresas, clientes y administración pública) y el tipo de residuos que se consideran (residuos domésticos, industriales, plástico, etc.). Asimismo, establecen que en el ámbito de residuos hay cinco grandes casos de uso: documentación del producto, certificación y registro, trazabilidad, tokenización y automatización con inteligencia artificial (IA) y dispositivos de Internet de las cosas (IoT).

En el ámbito de la trazabilidad, los proyectos más relevantes son Excess Materials Exchange (EME) y Circularise. El primero es una plataforma de nivel empresarial construida con Hyperledger Fabric que permite el intercambio de cualquier activo de producción de una empresa para que pueda ser utilizado en otras empresas. Otorga a cada activo una identidad (pasaporte de recursos) enlazada al activo físico mediante un código de barras, QR o chips para asegurar su trazabilidad. Este pasaporte de recursos contiene toda la información del activo (composición, características, etc.), que está almacenada en una base de datos centralizada. Se puede compartir parte de esta información con otras empresas.

Circularise, por su parte, utiliza una blockchain pública y sin permisionado como Ethereum para proporcionar trazabilidad sobre los residuos. Utiliza pruebas de conocimiento nulo (ZKP) para proteger datos sensibles y de propiedad de la empresa.

En el ámbito de la tokenización, plataformas como Plastic Bank, Recereum o Naturecoin ofrecen tokens en Ethereum como recompensa a aquellas personas u organizaciones que presenten residuos en puntos de recogida. Estos tokens pueden ser canjeados por productos, descuentos o recompensas en los establecimientos que acepten estos tokens.

Aparte de los proyectos mencionados, también hay que destacar PortNet [5]. Este proyecto está centrado en la trazabilidad de los residuos generados en el transporte marítimo. Utiliza el enfoque de pasaporte de residuos que denomina *Wastepass* y que está implementado mediante contratos inteligentes.

3. Marco teórico

Este capítulo tiene como objetivo ofrecer una introducción a los conceptos de Incoterm y tokenización, que serán utilizados para la gestión de las entregas de residuos y para la representación de los residuos plásticos en formato digital, respectivamente. Adicionalmente, se explican los fundamentos de Hyperledger Fabric, la blockchain que se utiliza en el desarrollo de la solución.

3.1. Incoterms

Los Incoterms definen reglas aceptadas globalmente tanto para el comercio internacional como para el comercio nacional. Los publica la Cámara Internacional de Comercio (en inglés *International Commercial Council*, ICC). Los Incoterms se representan en un formato de tres letras: EXW, FCA, CPT, DAT, etc. Aunque hay más de 10 Incoterms diferentes, se pueden agrupar en cuatro categorías en función de las obligaciones y los riesgos para el comprador y el vendedor [6]:

- *Categoría E.* La mercancía se entrega en la instalación del vendedor.
- *Categoría F.* El comprador elige al transportista principal.
- *Categoría C.* El vendedor elige al transportista principal.
- *Categoría D.* El vendedor entrega la mercancía en la instalación del comprador o en un lugar pactado con éste.

3.2. Tokenización

La tokenización es la representación de activos en formato digital mediante tokens. En este proyecto, se utilizan tokens no fungibles, es decir, tokens que representan activos únicos. El estándar más popular de este tipo de tokens es el ERC-721.

Sin embargo, en este proyecto es necesario que un mismo token contenga diferentes tokens no fungibles. Es lo que se denomina token compuesto (en inglés, *composable token*). Aunque para este tipo de tokens existen otros estándares como el ERC-998, se utiliza EIP-4944 [7] que solo permite que se emita un token ERC-721. De esta forma, la dirección del contrato puede identificar al único token emitido en el contrato y utilizarse como dirección propietaria de otros tokens ERC-721.

3.3. Hyperledger Fabric

La tecnología blockchain que se utiliza para el desarrollo de la PoC es Hyperledger Fabric. A continuación, se introducen los fundamentos de esta tecnología, así como las características más relevantes.

Como ya se ha mencionado, Hyperledger Fabric es una blockchain privada y permisionada que solo permite a las entidades autorizadas realizar operaciones sobre la red. No está destinada a un caso de uso específico, sino que abarca diferentes casos de uso en el sector empresarial. Todo esto queda confirmado en su documentación oficial, que define Hyperledger Fabric como una “plataforma de registro distribuido permisionado de grado empresarial que ofrece modularidad y versatilidad para un amplio conjunto de casos de uso de la industria” [8].

Las redes de Hyperledger Fabric constan de dos elementos principales: organizaciones y canales. Los miembros que pueden acceder a la red se denominan organizaciones. Los canales son *ledgers* (registros de transacciones) separados del resto que permiten a un subconjunto de organizaciones enviar transacciones sin que puedan ser consultadas por otras organizaciones que pertenezcan al canal.

Cada organización está constituida por los siguientes elementos:

- *Peer.* Es un tipo de nodo que aprueba transacciones y las almacena. Tiene instalados y ejecuta los contratos inteligentes.
- *Orderer.* Es otro tipo de nodo cuya función es recibir las transacciones, ordenarlas y devolverlas a las peers para su almacenamiento. El conjunto de orderers de un canal constituye el servicio de ordenamiento.
- *Autoridad de certificación* (CA). Emite identidades digitales, constituidas por una clave privada y un certificado digital x509. La clave privada se utiliza para firmar las transacciones antes de enviarlas a la red. Puede ser una CA raíz o una CA intermedia en función de si se emite o no su propio certificado digital.
- *Usuarios.* Pueden acceder a la red con unos determinados permisos y roles definidos por la misma organización.
- *Administrador.* Es un usuario especial de la organización que sirve para aceptar determinadas operaciones que requieren privilegios especiales, como aprobar la incorporación de una nueva organización o cambiar la configuración de un canal.
- *Membership Service Provider (MSP).* Es el mecanismo utilizado para verificar que una identidad digital ha sido emitida por una CA de

confianza para la organización. Todos los miembros de una organización que consulten o envíen transacciones deben tener una identidad digital emitida por una CA reconocida por el MSP de la organización.

En referencia al mecanismo de consenso de Hyperledger Fabric, éste se basa en algoritmos deterministas, que garantizan que cualquier bloque válido se añadirá a la cadena de bloques. Esta es una de las principales diferentes que tiene con blockchains públicas como Ethereum, donde el protocolo de consenso es probabilístico. En la actualidad, el único servicio de consenso que no está obsoleto es Raft. Este protocolo utiliza el modelo *leader and follower*, es decir, hay un nodo líder elegido de forma aleatoria entre los nodos de ordenamiento disponibles que replica los mensajes a los demás nodos. Es *crash fault tolerant* (CFT) porque puede tolerar la falla de menos de la mitad de los nodos de ordenamiento de un canal.

El último aspecto sobre Hyperledger Fabric que se va a explicar son los contratos inteligentes, que son programas que definen unas reglas para la ejecución de procesos sobre la blockchain en la que se han desplegado. En Hyperledger Fabric, los contratos inteligentes se despliegan empaquetados en forma de chaincode. Seguidamente, se enumeran las diferencias más relevantes que tienen con los contratos inteligentes de Ethereum:

- Se pueden programar en los lenguajes de programación de Go, Java o Javascript.
- Se puede determinar el nombre identificativo del chaincode, lo que se corresponde con la dirección de un contrato en Ethereum.
- Se pueden empaquetar dos o más contratos inteligentes dentro de un mismo chaincode.

4. Diseño

Este capítulo presenta el diseño de la propuesta, la definición de las entidades, los requisitos de seguridad y privacidad considerados, la arquitectura y los diagramas de casos de uso, de clases y de secuencia. El diseño de esta propuesta tiene en cuenta su posterior implementación en Hyperledger Fabric, por lo que se utilizan términos propios de esta tecnología.

4.1. Definición de entidades

Como se ha descrito en la introducción, hay diferentes actores involucrados en el ciclo de gestión de residuos plásticos. En este proyecto, se considera un caso genérico y se define una entidad para cada tipo de actor:

- *Generador de residuos*. Puede ser tanto una ciudad, representada por su Ayuntamiento, o un gran propietario, como fábricas o centros comerciales, entre otros. En todo caso, se considera que tiene la capacidad para estimar el peso y la pureza de los residuos plásticos.
- *Servicio de recogida.* Se encarga de recoger los residuos plásticos del generador de residuos y trasladarlos a la planta de selección. La recogida puede efectuarse en la instalación del generador de residuos o que el generador de residuos los traslade a un almacén del servicio de recogida.
- *Planta de selección.* Separa los residuos impropios, es decir, todo residuo que no sea plástico, y clasifica los residuos plásticos en los siete tipos de plástico definidos mediante el Código Identificador de Resina (RIC) [9]: PETE, HPDE, PVC, LPDE, PP, PP y otros. Una vez clasificados, los compacta en bloques y ordena su traslado a la planta de reciclaje.
- *Planta de reciclaje.* Recibe los residuos ya clasificados, los procesa y los convierte en gránulos de resina de plástico para que vuelvan a ser utilizables.
- *Fabricante.* Adquiere los gránulos de plástico reciclados para moldearlos y fabricar nuevos productos.
- *Transportista.* Transporta los residuos de una entidad a otra, como es el caso del traslado de los bloques de plástico clasificados desde la planta de selección a la planta de reciclaje o los gránulos de plástico desde la planta de reciclaje hasta el fabricante.

En este escenario no se han tenido en cuenta posibles agentes o negociantes que actúen en nombre de alguna organización. Tampoco se ha considerado la autoridad reguladora.

4.2. Requisitos de seguridad y privacidad

En esta sección se definen los requisitos de seguridad y privacidad que se consideran en la elaboración de la propuesta. Hay que tener en cuenta que los requisitos se definen a nivel de lote de residuos plásticos, no a nivel de un residuo plástico individual por motivos de eficiencia.

Los requisitos de seguridad que se tienen en cuenta para el desarrollo de la PoC son:

- *Trazabilidad*. Es el principal requisito que se persigue y consiste en que debe poder seguirse la evolución de los residuos plásticos, tanto de sus propiedades como de los diferentes traslados entre diferentes entidades.
- *Autenticación del propietario.* Los residuos plásticos deben estar asociados a su propietario en todo momento y únicamente el propietario o la entidad a la que éste autorice puede realizar operaciones sobre ellos (por ejemplo, el transportista).
- *No repudio*. Debe garantizarse que ninguna entidad pueda negar haber (o no haber) ejecutado un determinado proceso.
- *Disponibilidad*. Tanto la información como el sistema han de estar disponibles en todo momento para consultar la trazabilidad de cada lote de residuos plásticos y poder efectuar los correspondientes procesos sobre este lote.
- *Identificación*. Todas las entidades deben ser conocidas y estar identificadas a nivel de organización.
- *Transferibilidad segura.* El proceso de transferencia de residuos de una entidad a otra debe ser seguro. Una vez que se ha transferido un lote de residuos plástico, el propietario anterior no puede realizar ninguna operación sobre éste. Tampoco, se puede transferir ningún lote de residuos plásticos cuando ya exista un acuerdo de traslado en vigor.

En cuanto a la privacidad del sistema, se debe almacenar la mínima información posible en la cadena de bloques. En todo caso, los procesos internos de cada entidad son privados y, por lo tanto, no se publicarán. Tampoco debe revelarse información a terceros que no estén involucrados directamente en el sistema.

4.3. Arquitectura

Para cada una de las entidades definidas en la sección 4.1, se crea una organización en Hyperledger Fabric. Todas las organizaciones se conectan a un canal privado. La política de este canal define las organizaciones que pueden acceder al canal y desplegar e interactuar con los contratos inteligentes.

Cada organización consta de una CA y un *peer*. La CA dispone de un certificado digital autofirmado, aunque en un caso real este certificado podría ser emitido por una CA externa de confianza (por ejemplo, la autoridad reguladora). Por su parte, el *peer* permite a los miembros de la organización operar sobre el canal e interactuar con los contratos desplegados. Este nodo tiene una copia del *ledger* del canal y tiene instalados los contratos inteligentes. En la figura 2 se muestra un esquema general de la arquitectura de la red con las seis organizaciones consideradas y los componentes de cada una de ellas.

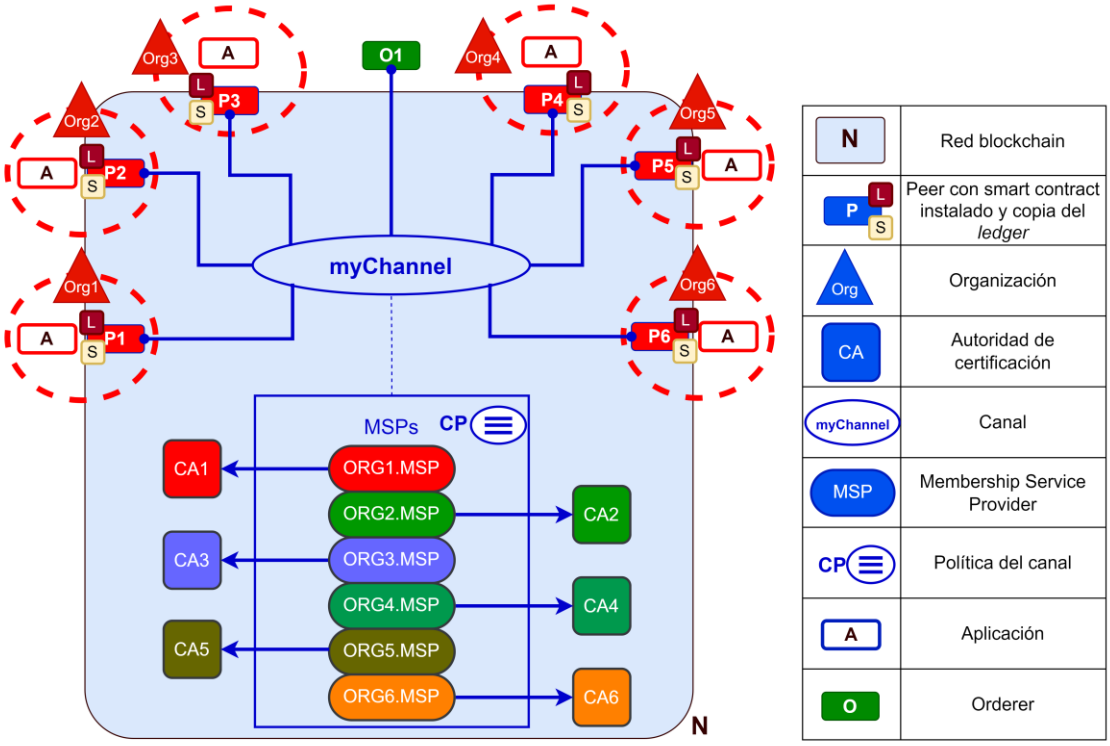


Figura 2. Arquitectura de la red

En la tabla 2 se muestra la relación entre los identificadores que aparecen en la figura 2 y las entidades definidas en la sección 4.1.

ID	Nombre
Org1	Generador de residuos
Org2	Servicio de recogida
Org3	Planta de selección
Org4	Planta de reciclaje
Org5	Fabricante
Org6	Transportista

Tabla 2. Identificadores de las organizaciones

En la PoC, la aplicación A es la misma para todas las organizaciones. No obstante, en un caso real puede ser diferente para cada organización. Esta aplicación tiene acceso a la wallet del usuario (o usuarios) responsables de la organización. En esta wallet, se guardan las claves privadas de los usuarios. Además, la aplicación debe tener acceso a una base de datos que almacene las direcciones de los contratos inteligentes desplegados en la red y que forman parte de la solución. En este caso, se guardan las direcciones de los contratos en un archivo de formato yaml compartido para todas las organizaciones.

4.4. Diagrama de casos de uso

El diagrama de casos de uso contiene las diferentes organizaciones y los procesos que puede realizar cada una de ellas, como se puede apreciar en la figura 3.

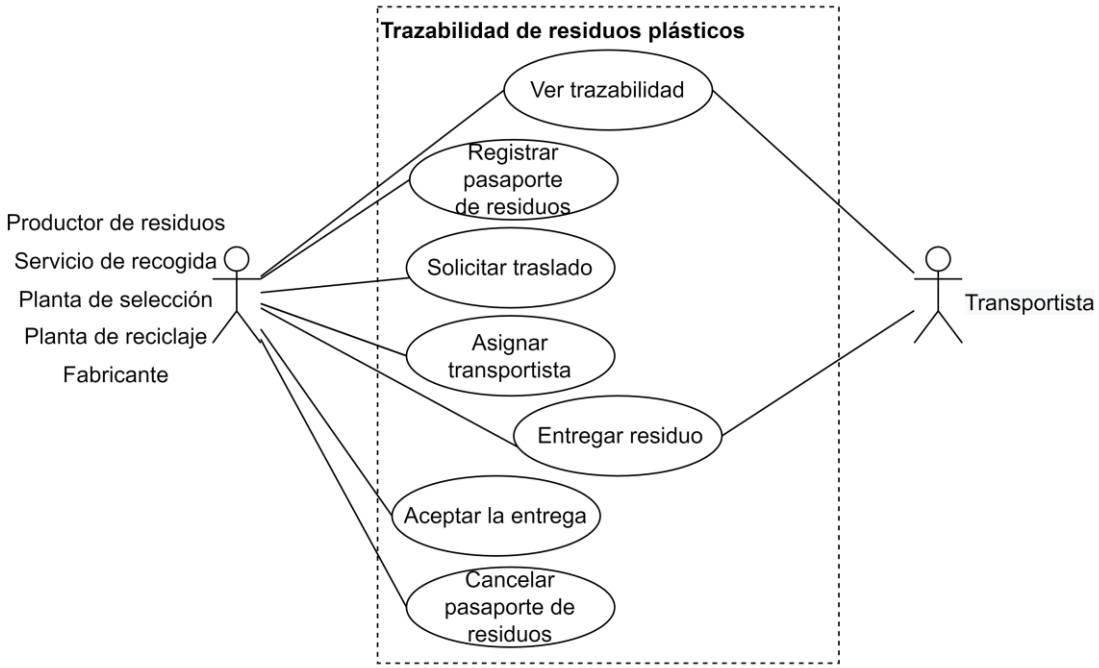


Figura 3. Diagrama de casos de uso

Seguidamente, se describen los procesos del diagrama anterior.

- *Ver trazabilidad.* Todas las organizaciones pueden consultar la trazabilidad de un lote de residuos plásticos en todas sus actualizaciones, tanto de creación o de modificación (clasificación o reciclaje) como de transferencia.
- *Registrar pasaporte de residuos.* Este proceso puede ser utilizado para registrar un nuevo lote de residuos plásticos y para anotar cambios en la composición de un lote (contenido, peso, etc.).
- *Solicitar traslado.* Se ejecuta cuando dos organizaciones han llegado a un acuerdo para el traslado de un lote de residuos plásticos de una organización a otra.
- *Asignar transportista.* Este proceso puede ser realizado por la organización de origen o de destino del traslado cuando el transporte no sea directamente realizado por una de las dos organizaciones.
- *Entregar residuo.* El residuo se entrega a la organización correspondiente en función de las condiciones negociadas en el contrato de traslado.
- *Aceptar la entrega.* El destinatario confirma que acepta el lote de residuos plásticos.
- *Cancelar pasaporte de residuos.* Sirve para eliminar un lote de residuos plásticos del sistema y desvincularlo de su propietario porque lo haya utilizado para fabricar un nuevo producto (caso del fabricante) o lo haya enviado a una organización externa al sistema para su eliminación (caso de la planta de selección con los residuos plásticos no reciclables).

En caso de que alguna organización rechace los residuos plásticos, los pueden devolver iniciando un nuevo proceso de transferencia. Adicionalmente, no se tienen en cuenta los procesos para cancelar los traslados.

4.5. Diagramas de clases

En esta sección se describen los contratos inteligentes y su relación entre ellos mediante la representación del diagrama de clases de la solución. En primer lugar, los contratos inteligentes ERC721 y ERC4944 implementan los estándares ERC-721 y EIP-4944, respectivamente.

El contrato inteligente WastePass gestiona la información y los procesos relacionados con el lote de plástico que tiene asociado. Hereda las funcionalidades del ERC4944 y mantiene la siguiente estructura de datos:

- *PlasticBatch*. Contiene los datos relacionados con el lote de plástico:
    - *Batch*. Es un identificador del lote de plástico. Contiene el identificador de la transacción con el que se ha registrado el WastePass.
    - *Stuff*. Indica el tipo de plástico que contiene el lote de acuerdo con los tipos definidos en RIC 0: PETE, HPDE, PVC, LPDE, PP, PP y otros. Cuando no se conozca el tipo de plástico o contenga diferentes tipos de plástico, el valor será *None*.
    - *Purity*. Indica el nivel de pureza del plástico que contiene.
    - Weight. Indica el peso neto en kilogramos del lote de plástico.
- *PreviousWastePassID*. Es el identificador del WastePass anterior.

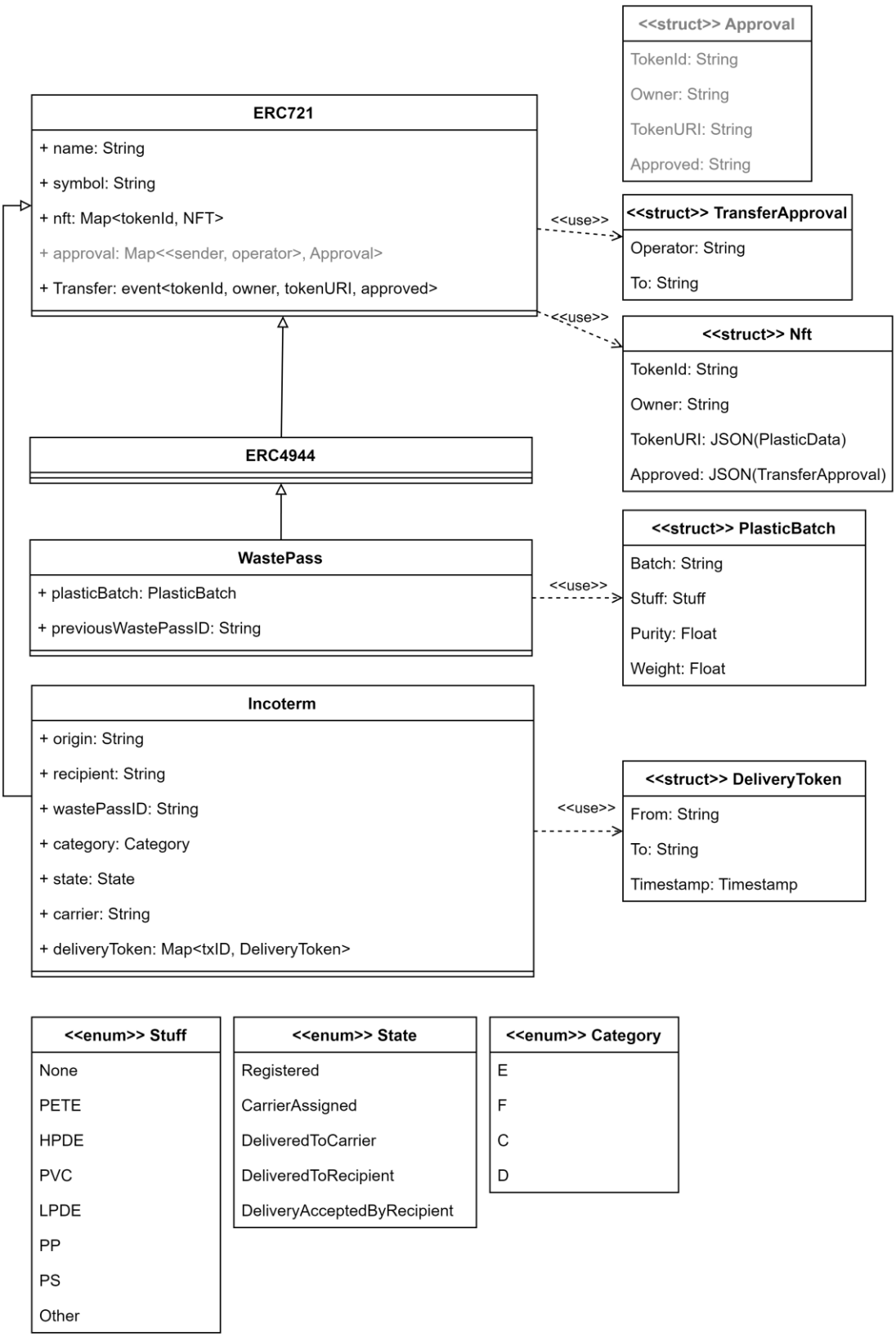


Figura 4. Diagrama de clases

El contrato inteligente Incoterm contiene la información y los procesos relacionados con el traslado de un lote de plástico de una instalación de origen a una de destino. Hereda del contrato inteligente ERC721 y mantiene la siguiente estructura de datos:

- *Origin*. Indica la organización que posee el lote de residuos plásticos.
- *Recipient*. Indica la organización que adquiere el lote de residuos plásticos.
- *WastePassID*. Contiene el identificador del WastePass asociado con el Incoterm.
- *Category*. Indica la categoría del Incoterm (E, F, C y D).
- *State*. Indica el estado actual del Incoterm. Puede ser uno de los siguientes estados:
    - *Registered*. El WastePass ha sido registrado.
    - *CarrierAssigned*. El transportista ha sido asignado.
    - *DeliveredToCarrier*. El lote de residuos plásticos ha sido recogido de la instalación de origen por el transportista.
    - *DeliveredToRecipient*. El lote de residuos plásticos ha sido entregado en la instalación de destino.
    - *DeliveryAcceptedByRecipient*. El lote de residuos plásticos ha sido aceptado por el destinatario.
- *Carrier*. Contiene la información del transportista. Este campo estará vacío cuando la categoría del Incoterm sea E o D.
- *DeliveryTokens*. Contiene los tokens NFT que se emiten a lo largo del traslado. Todos estos tokens son propiedad del WastePass asociado a este Incoterm. Para cada token, se almacena la siguiente información:
    - *From*. Indica quien entrega el lote de residuos plásticos.
    - *To*. Indica quien recibe el lote de residuos plásticos.
    - *Timestamp*. Indica el momento en el que se registró la entrega.

En los campos anteriores que se refieren a una organización, se almacenan la identidad de la organización en el formato de secuencia RDN:

```
CN=[CA],OU=[OU],O=[O],L=[L],C=[C]
```

De esta forma, cada organización puede optar por el modelo de autenticación que más le convenga, como emitir certificados a sus empleados (como podría ser el caso del transportista) o emitir un solo certificado asociado a un servidor en su instalación y que desde éste se ejecuten las transacciones.

4.6. Diagramas de secuencia

Una vez explicado el diagrama de clases, se detallan las interacciones entre las diferentes entidades (organizaciones y contratos inteligentes) contempladas en el sistema. Antes de nada, se supone que cada proceso es ejecutado por un responsable de la organización.

El primer proceso es el del registro de un lote de residuos plásticos por parte de su propietario. Este proceso se realiza mediante la función *Initialize* del contrato WastePass y el propietario debe especificar las características del lote de residuos (contenido, pureza y peso). Con estas características, el WastePass emite el único NFT del contrato cuyo identificador es “0”. En caso de que el lote de residuos plásticos proceda de otro lote previo, el propietario indicará el identificador del WastePass anterior. En tal caso, el contrato inteligente ejecutará la función *Burn* del *WastePass* anterior para cancelarlo.

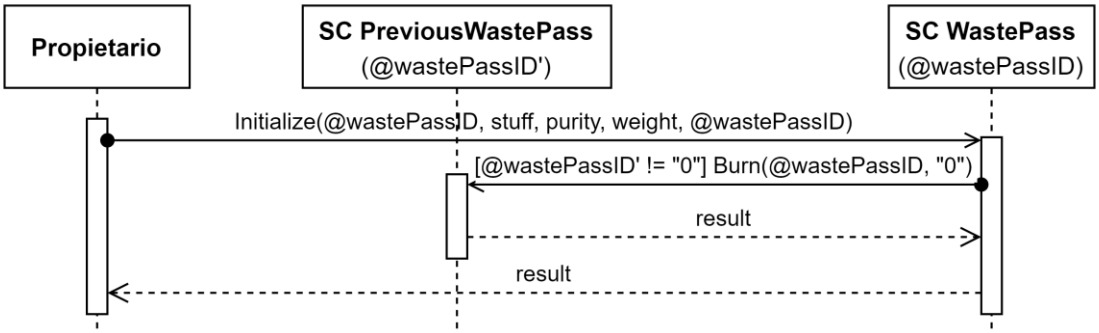


Figura 5. Diagrama de secuencia de la función *Initialize* del WastePass

El otro proceso que está presente en el WastePass es la cancelación de un lote de residuos plásticos. Este proceso únicamente puede ser ejecutado por el propietario del lote, que indica el identificador del NFT del WastePass.

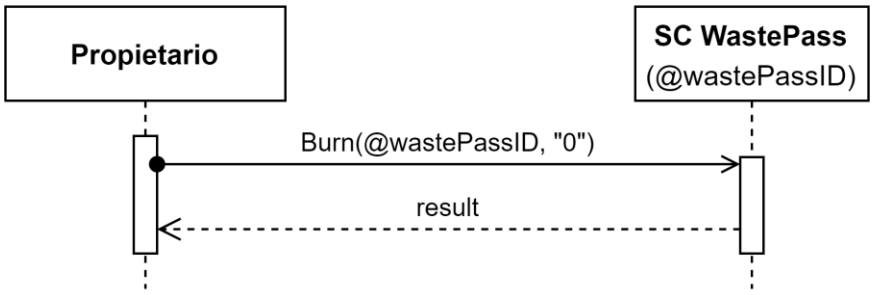


Figura 6. Diagrama de secuencia de la función *Burn* del *WastePass*

A continuación, se describen los procesos relacionados con el traslado de un lote de residuos plásticos de una organización a otra. Estos procesos están gestionados mediante Incoterms. Se supone que las condiciones del traslado han sido negociadas con anterioridad a la ejecución de los siguientes

procesos. En primer lugar, el propietario de un lote de residuos plásticos (origen) solicita el traslado de los residuos a otra organización (destinatario). Para hacer esto, el propietario ejecuta la función *Initialize* del Incoterm e indica el destinatario, el identificador del WastePass y la categoría del Incoterm (E, F, C o D). El Incoterm actualiza su estado a *Registered* o a *CarrierAssigned*, en función si es necesario o no asignar a un transportista, y se transfiere el WastePass indicado a sí mismo para gestionar el traslado del lote de residuos plásticos de una organización a otra y evitar que el lote se transfiera a otra organización mientras el Incoterm está vigente. Una vez ejecutada esta función, se notifica al destinatario que el Incoterm ya está inicializado.

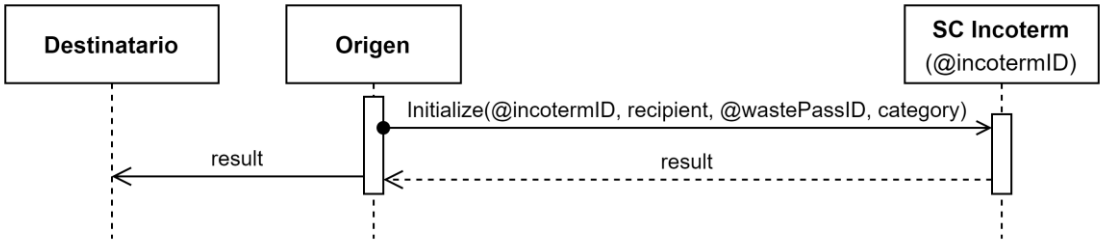


Figura 7. Diagrama de secuencia de la función *Initialize* del *Incoterm*

Cuando la categoría del Incoterm sea C o F, un transportista debe ser asignado por el origen o el destinatario del traslado, respectivamente. Para realizar este proceso, se ejecuta la función *AssignCarrier* y se indica el transportista asignado. El estado del Incoterm pasa a *CarrierAssigned.* Una vez ejecutada esta función, se notifica al transportista que tiene un traslado pendiente.

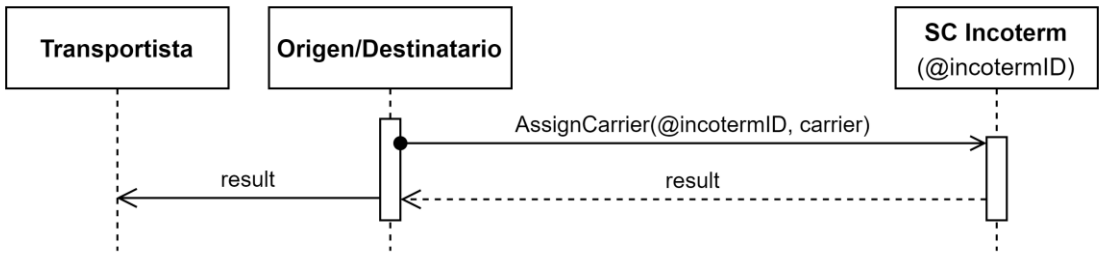


Figura 8. Diagrama de secuencia de la función *AssignCarrier* del *Incoterm*

Con el contrato Incoterm inicializado y el transportista asignado se puede iniciar el traslado. La figura 9 muestra el diagrama de secuencia de un traslado con transportista. El origen entrega el lote de residuos plásticos al transportista y ejecuta la función *Deliver* del Incoterm para que quede constancia de este hecho. Tras esto, el estado del Incoterm pasa a *DeliveredToCarrier.*

El transportista traslada el lote de residuos hasta la instalación del destinatario. Cuando completa este traslado, ejecuta la función *Deliver* para

actualizar el estado del contrato a *DeliveredToRecipient*. Cada vez que se ejecuta la función *Deliver* se emite un token de tipo *DeliveryToken,* que registra la entrega del lote de residuos plásticos desde el origen al transportista o desde el transportista al destinatario, así como la fecha en que tiene lugar esta entrega.

Por último, el destinatario confirma que acepta el lote de residuos plásticos mediante la ejecución de la función *AcceptDelivery*. El Incoterm transfiere la propiedad del WastePass al destinatario.

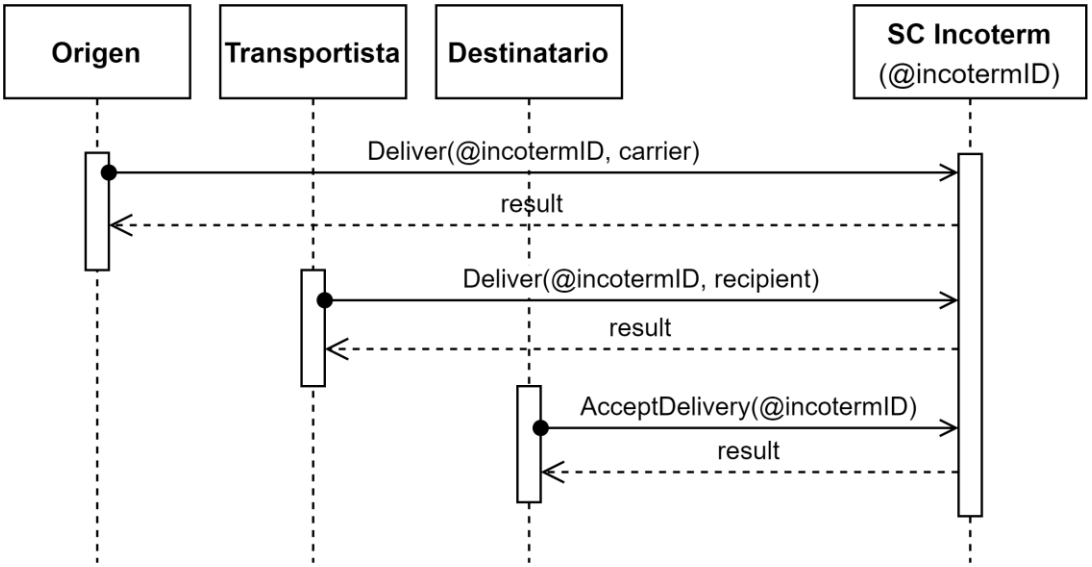


Figura 9. Diagrama de secuencia de un traslado con transportista

5. Implementación

Como ya se ha explicado, se utiliza Hyperledger Fabric como blockchain para el despliegue de los contratos inteligentes. Se utiliza Docker y Docker Compose para el despliegue de los diferentes componentes de la arquitectura de la red. En cuanto a los contratos inteligentes, se implementan mediante el lenguaje de programación Go. La aplicación de escritorio también se implementa mediante el lenguaje de programación Go y se utiliza la librería Fyne para desarrollar la interfaz gráfica.

En este capítulo se explica la configuración de la red blockchain y la implementación de los contratos inteligentes. En cuanto a la aplicación de escritorio, se muestra su funcionamiento en el capítulo siguiente.

5.1. Configuración de la red blockchain

Se han seguido los pasos disponibles en la guía de instalación de Hyperledger Fabric[1] para preparar el entorno de desarrollo. Estos pasos se explican seguidamente:

1. Descargar el repositorio:

```
git clone https://github.com/hyperledger/fabric-samples
```

2. Descargar el archivo bash para instalar los archivos binarios:

```
curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh
```

3. Descargar Fabric 2.4.6. Esto genera los directorios bin y config:

```
./install-fabric.sh --fabric-version 2.4.6 binary
```

4. Como se está utilizando Windows, se debe instalar manualmente JQ:

```
curl -L -o /usr/bin/jq.exe https://github.com/stedolan/jq/releases/latest/download/jq-win64.exe
```

El repositorio de fabric-samples proporciona un ejemplo de red de prueba en el directorio test-network. Partiendo de este ejemplo, se realizan las modificaciones necesarias para configurar la red del proyecto de acuerdo con la arquitectura diseñada.

[1] https://hyperledger-fabric.readthedocs.io/en/latest/install.html

En primer lugar, se modifican los archivos de docker-compose-ca.yaml y de docker-compose-test-net.yaml para que se despliegue una CA y un peer para cada una de las seis organizaciones. En las figuras 10 y 11 se muestra un ejemplo de la configuración para un contenedor de una CA y de un peer respectivamente.

```yaml
ca_org1:
  image: hyperledger/fabric-ca:1.5.5
  environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    - FABRIC_CA_SERVER_CA_NAME=ca-org1
    - FABRIC_CA_SERVER_TLS_ENABLED=true
    - FABRIC_CA_SERVER_PORT=1054
    - FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS=0.0.0.0:11054
  ports:
    - "1054:1054"
    - "11054:11054"
  command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
  volumes:
    - ../organizations/fabric-ca/org1:/etc/hyperledger/fabric-ca-server
  container_name: ca_org1
  networks:
    - test
```

Figura 10. Contenedor Docker de la CA

```yaml
peer0.org1.example.com:
  container_name: peer0.org1.example.com
  image: hyperledger/fabric-peer:2.4.6
  environment:
    #Generic peer variables
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=fabric_test
    - FABRIC_LOGGING_SPEC=INFO
    #- FABRIC_LOGGING_SPEC=DEBUG
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_PROFILE_ENABLED=true
    - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
    - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
    - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    # Peer specific variabes
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_ADDRESS=peer0.org1.example.com:11051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:11051
    - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:1
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:1
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:11051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:11051
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_OPERATIONS_LISTENADDRESS=peer0.org1.example.com:9444
  volumes:
    - /var/run/docker.sock:/host/var/run/docker.sock
    - ../organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp
    - ../organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
    - peer0.org1.example.com:/var/hyperledger/production
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
  command: peer node start
  ports:
    - 11051:11051
    - 9444:9444
  networks:
    - test
```

Figura 11. Contenedor Docker de un peer

Adicionalmente, se establece que el consorcio de la red está compuesto por las seis organizaciones y se configura que un canal privado para estas organizaciones.

```
SixOrgsOrdererGenesis:
    <<: *ChannelDefaults
    Orderer:
        <<: *OrdererDefaults
        Organizations:
            - *OrdererOrg
        Capabilities:
            <<: *OrdererCapabilities
    Consortiums:
        SampleConsortium:
            Organizations:
                - *Org1
                - *Org2
                - *Org3
                - *Org4
                - *Org5
                - *Org6
```

```
SixOrgsChannel:
    Consortium: SampleConsortium
    <<: *ChannelDefaults
    Application:
        <<: *ApplicationDefaults
        Organizations:
            - *Org1
            - *Org2
            - *Org3
            - *Org4
            - *Org5
            - *Org6
        Capabilities:
            <<: *ApplicationCapabilities
```

Figura 12. Configuración del consorcio y del canal

Las políticas de consenso del canal se dejan por defecto. En general, cualquier organización puede leer y escribir datos, mientras que las modificaciones en la configuración del canal, añadir o eliminar organizaciones, desplegar nuevos chaincodes, entre otras operaciones, deben ser aprobadas por la mayoría de las organizaciones del canal.

```
#   /Channel/Application/<PolicyName>
Policies:
    Readers:
        Type: ImplicitMeta
        Rule: "ANY Readers"
    Writers:
        Type: ImplicitMeta
        Rule: "ANY Writers"
    Admins:
        Type: ImplicitMeta
        Rule: "MAJORITY Admins"
    LifecycleEndorsement:
        Type: ImplicitMeta
        Rule: "MAJORITY Endorsement"
    Endorsement:
        Type: ImplicitMeta
        Rule: "MAJORITY Endorsement"
```

Figura 13. Política del canal

Una vez concluida la configuración, se puede iniciar la red, crear el canal y el material criptográfico con las CAs mediante el siguiente comando:

```
./network.sh up createChannel -ca
```

5.2. Contratos inteligentes

Los contratos inteligentes se han desarrollado en el lenguaje de programación Go. Se ha utilizado la librería *fabric-contract-api-go* para realizar las operaciones de lectura y escritura de variables del contrato.

El despliegue de cada chaincode genera un nuevo contenedor Docker por cada organización. Debido a la falta de recursos computacionales para generar un contenedor Docker por cada chaincode, se hace necesario crear un solo chaincode que contenga todos los contratos. La figura 14 contiene un esquema del resultado que se detalla en los siguientes párrafos.

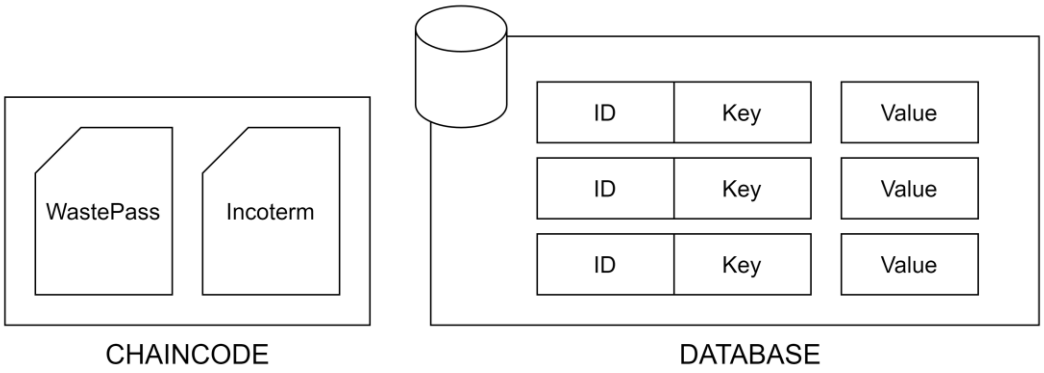


Figura 14. Esquema del chaincode

En primer lugar, se aprovecha una utilidad de Hyperledger Fabric que permite empaquetar diferentes contratos inteligentes dentro de un mismo chaincode. La figura 15 muestra cómo se empaquetan los contratos de WastePass e Incoterm dentro del mismo chaincode.

```
chaincode, err := contractapi.NewChaincode(
  new(wastepass.WastePassContract),
  new(incoterm.IncotermContract),
)
```

Figura 15. Empaquetado de los contratos en un único chaincode

Esto provoca que no se diferencien los datos de dos WastePass ni dos Incoterms distintos. Además, hay que tener en cuenta que todos los datos que se encuentran dentro de un mismo chaincode comparten la misma base de datos. Para solventar esta situación, se asocia el valor de cada variable a una

nueva clave compuesta por el identificador del contrato y el nombre de la variable correspondiente, separados por ":".

```go
// Get complete key
func CreateKey(contract string, objectKey string) string {
  return contract + ":" + objectKey
}
```

Figura 16. Implementación de la función *CreateKey*

Para indicar a cada función el contrato al que se está llamando, se añade un parámetro de entrada a cada una de ellas.

Seguidamente, se explican los aspectos más relevantes de la implementación de cada uno de los contratos. El contrato ERC721 se ha obtenido de la implementación presente en el repositorio de fabric-samples[2].

El contrato ERC4944 hereda del ERC721 y emite el único token NFT del contrato cuyo propietario es la organización desde la que se ha ejecutado la función de inicialización. Junto a lo anterior, se sobrescribe la función *MintWithTokenURI* para evitar que se creen otros NFTs.

```go
func (c *TokenERC4944Contract) Initialize(ctx contractapi.TransactionContextInterface, contract string,
name string, symbol string, tokenURI string) (bool, error) {
  minter, err := erc721.ClientAccountID(ctx)
  if err != nil {
    return false, err
  }

  _, err = erc721.MintWithTokenURI(ctx, contract, minter, "0", tokenURI)
  if err != nil {
    return false, err
  }

  return c.TokenERC721Contract.Initialize(ctx, contract, name, symbol)
}
```

```go
func (c *TokenERC4944Contract) MintWithTokenURI(ctx contractapi.TransactionContextInterface, tokenId string,
tokenURI string) (*erc721.Nft, error) {
  return nil, nil
}
```

Figura 17. Implementación del contrato ERC4944

El contrato WastePass hereda del ERC4944. Este contrato se inicializa con los parámetros del lote de residuos plásticos (*stuff*, *purity* y *weight*) y con el identificador del WastePass anterior (si no hay un WastePass anterior, se le pasa el valor "0"). El contrato crea el lote de residuos plásticos con el identificador de la transacción como el valor del *batch* y se almacena en formato JSON en la variable *tokenURI*. El WastePass anterior, en caso de haberlo, se cancela siempre y cuando el que registra el nuevo WastePass sea

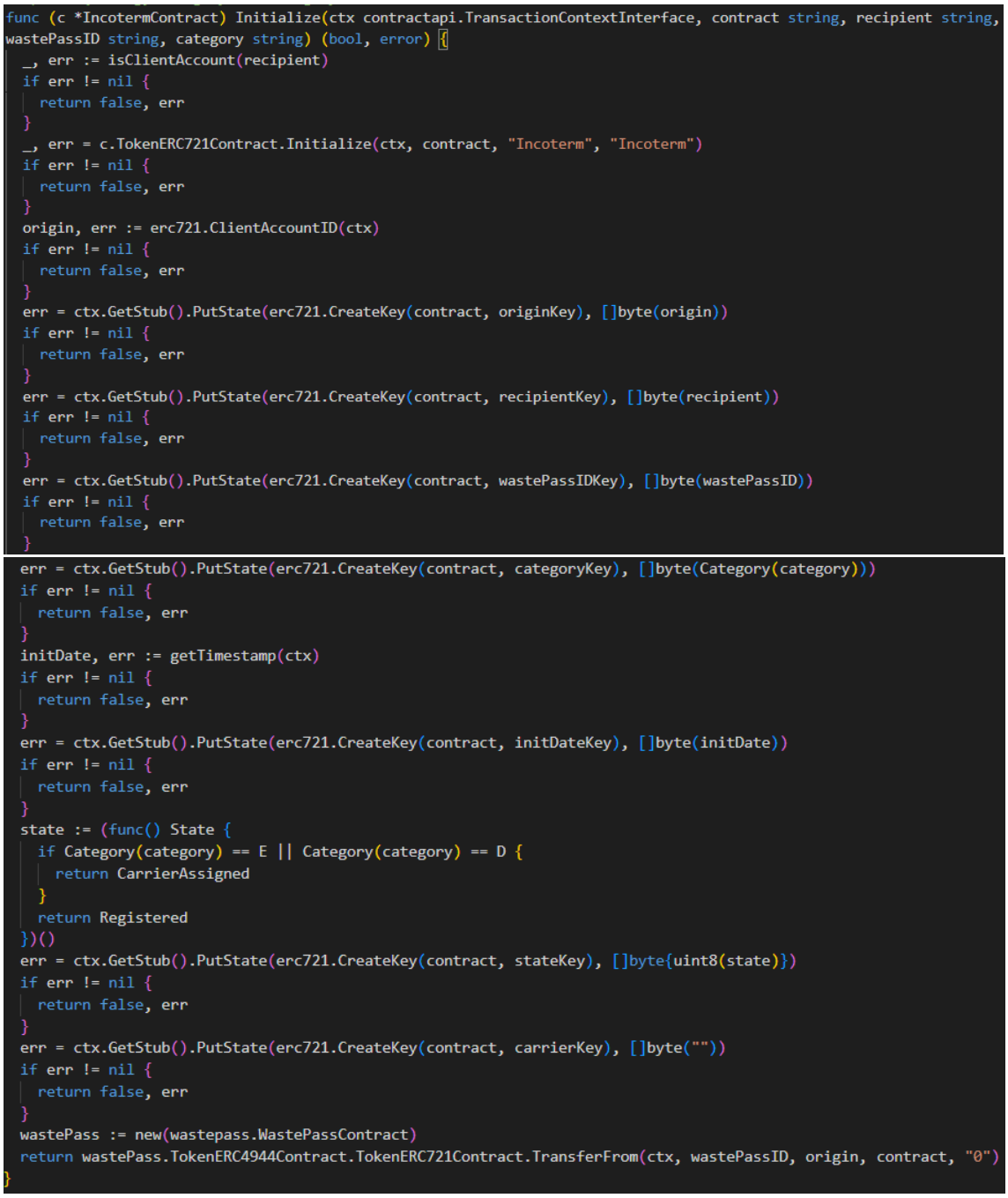
[2] https://github.com/hyperledger/fabric-samples/tree/main/token-erc-721

el propietario del WastePass anterior. Finalmente, se inicializa el ERC4944 con el *tokenURI* y con “WastePass” como nombre y símbolo.

```
func (c *WastePassContract) Initialize(ctx contractapi.TransactionContextInterface, contract string, stuff uint8,
purity float32, weight float32, previousWastePassID string) (bool, error) {
  if (purity >= 0 && purity <= 100) == false {
    return false, fmt.Errorf("purity's range is from 0 to 100")
  }
  plasticBatch := PlasticBatch{
    Batch:  ctx.GetStub().GetTxID(),
    Stuff:  Stuff(stuff),
    Purity: purity,
    Weight: weight,
  }
  tokenURIBytes, err := json.Marshal(plasticBatch)
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, previousWastePassIDKey), []byte(previousWastePassID))
  if err != nil {
    return false, err
  }
  if previousWastePassID != "0" {
    _, err = c.Burn(ctx, previousWastePassID, "0")
    if err != nil {
      return false, fmt.Errorf("failed to create new WastePass: %v", err)
    }
  }
  return c.TokenERC4944Contract.Initialize(ctx, contract, "WastePass", "WastePass", string(tokenURIBytes))
}
```

Figura 18. Función *Initialize* de WastePass

El contrato Incoterm hereda del ERC721 y almacena los NFTs que se van generando a lo largo del traslado. Este contrato se inicializa por el origen, que indica el destinatario, el identificador del WastePass y la categoría del Incoterm. El contrato comprueba que el destinatario proporcionado tenga el formato de una secuencia RDN, inicializa el contrato ERC721 con el string “Incoterm” como nombre y símbolo, almacenan las diferentes variables (origen, destinatario, ID del WastePass, categoría, estado y transportista) y transfiere el NFT único del WastePass al Incoterm para indicar que este contrato es ahora quien gestiona el lote de residuos plásticos.



```
func (c *IncotermContract) Initialize(ctx contractapi.TransactionContextInterface, contract string, recipient string,
wastePassID string, category string) (bool, error) {
  _, err := isClientAccount(recipient)
  if err != nil {
    return false, err
  }
  _, err = c.TokenERC721Contract.Initialize(ctx, contract, "Incoterm", "Incoterm")
  if err != nil {
    return false, err
  }
  origin, err := erc721.ClientAccountID(ctx)
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, originKey), []byte(origin))
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, recipientKey), []byte(recipient))
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, wastePassIDKey), []byte(wastePassID))
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, categoryKey), []byte(Category(category)))
  if err != nil {
    return false, err
  }
  initDate, err := getTimestamp(ctx)
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, initDateKey), []byte(initDate))
  if err != nil {
    return false, err
  }
  state := (func() State {
    if Category(category) == E || Category(category) == D {
      return CarrierAssigned
    }
    return Registered
  })()
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, stateKey), []byte{uint8(state)})
  if err != nil {
    return false, err
  }
  err = ctx.GetStub().PutState(erc721.CreateKey(contract, carrierKey), []byte(""))
  if err != nil {
    return false, err
  }
  wastePass := new(wastepass.WastePassContract)
  return wastePass.TokenERC4944Contract.TokenERC721Contract.TransferFrom(ctx, wastePassID, origin, contract, "0")
}
```

Figura 19. Implementación de la función Initialize de Incoterm

El resto de las funciones principales del Incoterm que se describen son *assignCarrier*, *deliver* y *acceptDelivery*. Las respectivas capturas de código incluyen solamente las operaciones principales, dejando de lado la recuperación de parámetros del estado del chaincode.

La función *assignCarrier* recibe como parámetro el transportista. El contrato recupera las variables requeridas (*origin, recipient, category* y *state*), verifica que la organización que ejecuta la función es el origen o el destinatario en función de si la categoría del Incoterm es C o F respectivamente. Después,

almacena el transportista asignado y actualiza el estado del Incoterm a *CarrierAssigned*.

```
if state != Registered {
  return false, fmt.Errorf("onlyState %d", Registered)
}

if category == E || category == D {
  return false, fmt.Errorf("incoterm E or D not require external carrier")
}

if category == C && origin != caller {
  return false, fmt.Errorf("only sender can assign the carrier in incoterm of category C")
}

if category == F && recipient != caller {
  return false, fmt.Errorf("only recipient can assign the carrier in incoterm of category F")
}

err = ctx.GetStub().PutState(erc721.CreateKey(contract, carrierKey), []byte(carrier))
if err != nil {
  return false, err
}

err = ctx.GetStub().PutState(erc721.CreateKey(contract, stateKey), []byte{uint8(CarrierAssigned)})
if err != nil {
  return false, err
}
```

Figura 20. Implementación de la función *deliver* del Incoterm

La función *deliver* recibe como parámetro el receptor del lote de residuos plásticos (puede ser el transportista o el destinatario final). El contrato recupera las variables requeridas (*origin, recipient, carrier, category* y *state*), verifica cuáles son las organizaciones involucradas en la entrega (de origen a destinatario, de origen a transportista o de transportista a destinatario) y, en función del caso, verifica que la entrega se realiza de acuerdo con la categoría del Incoterm. Finalmente, actualiza el estado del Incoterm.

```
if caller != origin && caller != carrier {
  return false, fmt.Errorf("caller is not the origin or the carrier of the incoterm")
}

var beforeState, afterState State

if caller == origin {
  if category == E || category == D {
    if to != recipient {
      return false, fmt.Errorf("'to' parameter is not the recipient of the incoterm")
    }
    beforeState = CarrierAssigned
    afterState = DeliveredToRecipient
  } else {
    if to != carrier {
      return false, fmt.Errorf("'to' parameter is not the carrier of the incoterm")
    }
    beforeState = CarrierAssigned
    afterState = DeliveredToCarrier
  }
} else {
  if to != recipient {
    return false, fmt.Errorf("'to' parameter is not the recipient of the incoterm")
  }
  beforeState = DeliveredToCarrier
  afterState = DeliveredToRecipient
}

_, err = MintDeliveryTokens(ctx, contract, wastePassID, caller, to)
if err != nil {
  return false, err
}

err = changeState(ctx, contract, beforeState, afterState)
if err != nil {
  return false, err
}
```

Figura 21. Implementación de la función *Deliver* de Incoterm

La función *acceptDelivery* se ejecuta por el destinatario y no recibe ningún parámetro de entrada. El contrato cambia el estado del Incoterm de *DeliveredToRecipient* a *DeliveryAcceptedByRecipient* y transfiere el NFT único del WastePass al destinatario como su nuevo propietario.

```
if caller != recipient {
  return false, fmt.Errorf("caller is not the recipient of the incoterm")
}

err = changeState(ctx, contract, DeliveredToRecipient, DeliveryAcceptedByRecipient)
if err != nil {
  return false, err
}

wastePassID, err := c.GetWastePassID(ctx, contract)
if err != nil {
  return false, err
}

return erc721.TransferFrom(ctx, wastePassID, contract, recipient, "0")
```

Figura 22. Implementación de la función *acceptDelivery* de Incoterm

Una vez explicado el funcionamiento general de los contratos inteligentes, se presenta el comando para desplegar el chaincode. Se establece *plasticTraceability* como nombre del chaincode.

```
./network.sh deployCC -ccn plasticTraceability -ccp ../chaincodes/ -ccl go
```

6. Resultados

Para demostrar el funcionamiento de la solución implementada se presenta un flujo de ejecución con la aplicación de escritorio siguiendo el esquema de la figura 23.

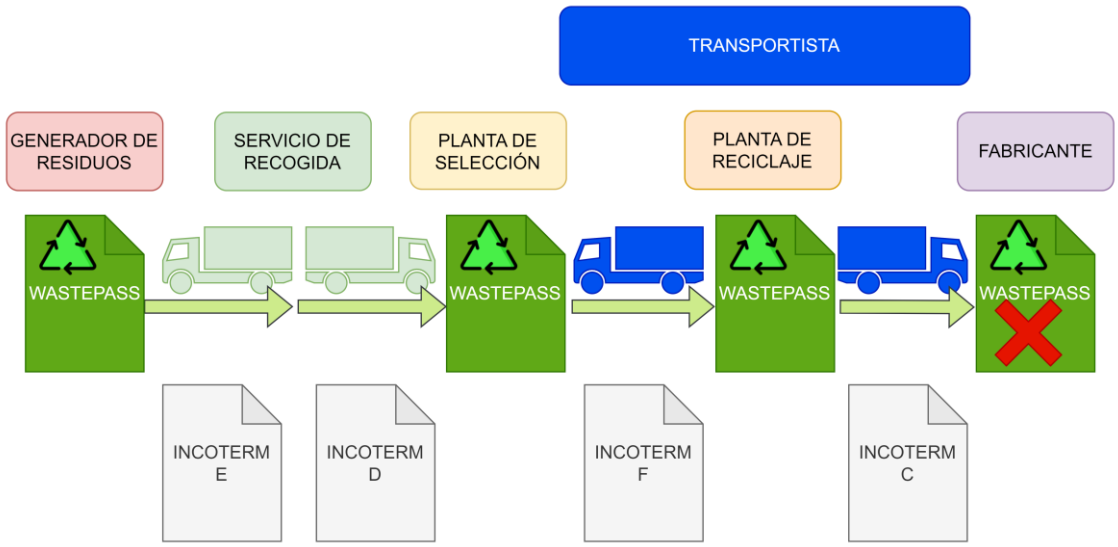


Figura 23. Esquema de la demostración

El generador de residuos registra un nuevo pasaporte de residuos, que se muestra en la tabla. Como el batch es muy extenso (tiene una longitud de 64 caracteres) se acorta convirtiendo su valor a base64.

Figura 24. Registro de un nuevo pasaporte de residuos por el generador de residuos

El generador de residuos solicita el traslado del lote de residuos plásticos recién registrado al servicio de recogida. Como el servicio de recogida los recogerá se aplica la categoría E de Incoterm. En la tabla de pasaportes de residuos, se actualiza el propietario del WastePass al nuevo Incoterm 11DA9936D475, que aparece en el listado de Incoterms con el estado “preparado”.

Figura 25. Solicitud de traslado al servicio de recogida por el generador de residuos

El generador de residuos entrega al servicio de recogida el lote de residuos y el estado del Incoterm se actualiza a “entregado”.

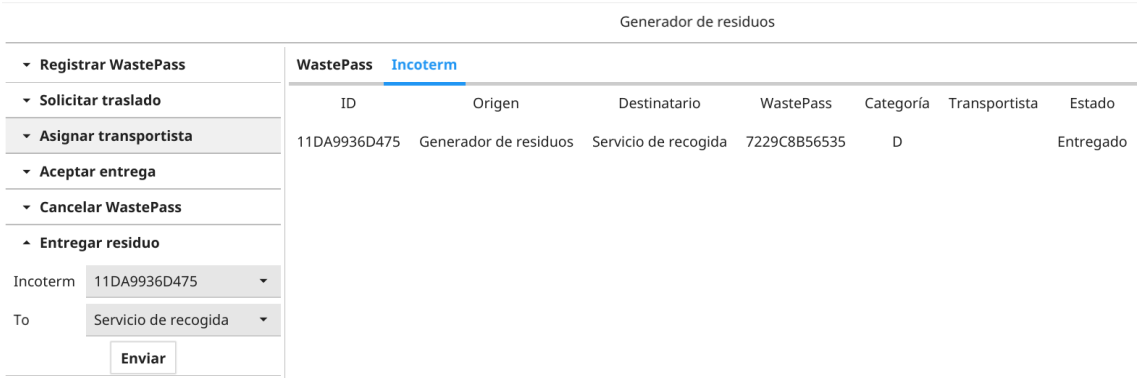


Figura 26. Entrega de residuos del generador de residuos al servicio de recogida

El servicio de recogida confirma que acepta el lote de residuos. El propietario del WastePass pasa a ser el servicio de recogida y el estado del Incoterm se actualiza a “aceptado”.

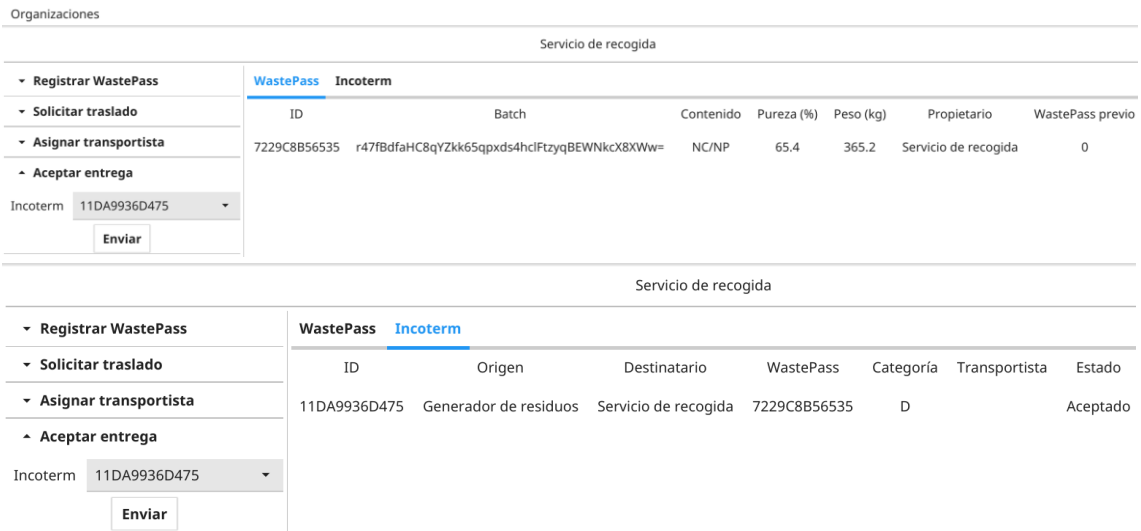


Figura 27. Aceptación de residuos por el servicio de recogida

A continuación, el servicio de recogida solicita el traslado del lote de residuos a la planta de selección. Simplemente se muestra la solicitud de traslado, puesto que el resto de procesos son iguales que los anteriores. La única diferencia es que la categoría del Incoterm es D, por lo que el servicio de recogida se responsabiliza de realizar el traslado.

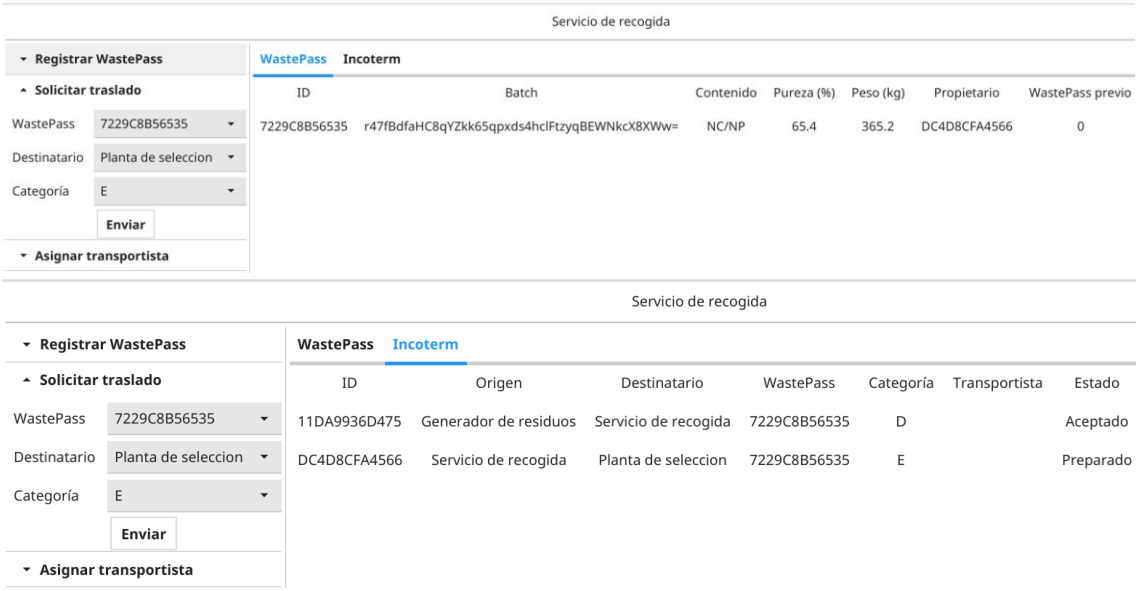


Figura 28. Solicitud de traslado a la planta de selección por el servicio de recogida

Una vez el lote de residuos se encuentra en la planta de selección, se registra un nuevo pasaporte de residuos para indicar el nuevo lote de residuos se ha generado a partir del lote asociado al pasaporte de residuos anterior. Como se puede observar, el propietario del WastePass anterior pasa a ser “0x0” y el nuevo WastePass contiene la referencia del WastePass anterior (7229C8B56535).

Figura 29. Registro de un nuevo pasaporte de residuos por la planta de selección

A continuación, la planta de selección solicita el traslado a la planta de reciclaje. En esta ocasión, se realizará mediante un transportista asignado por la planta de reciclaje, por lo que la categoría es F.

Figura 30. Solicitud de traslado a la planta de reciclaje por la planta de selección

La planta de reciclaje asigna al transportista. Acto seguido, se actualiza el Incoterm con el transportista asignado y el estado pasa a “preparado”.

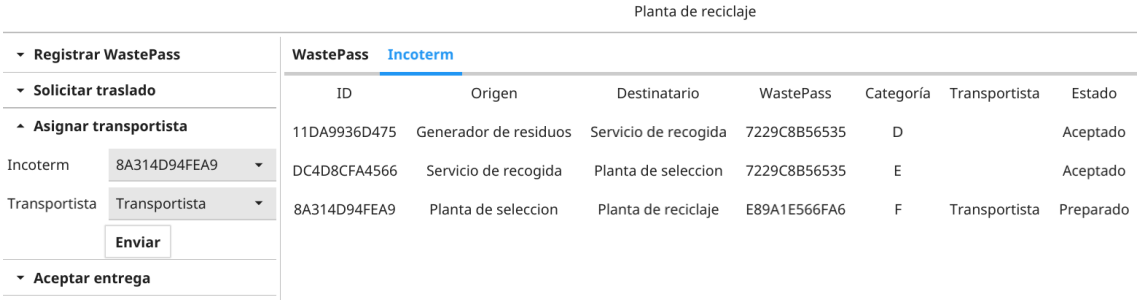


Figura 31. Asignación del transportista por la planta de reciclaje

La planta de selección entrega el lote de residuos al transportista y el estado del Incoterm pasa a “recogido”.

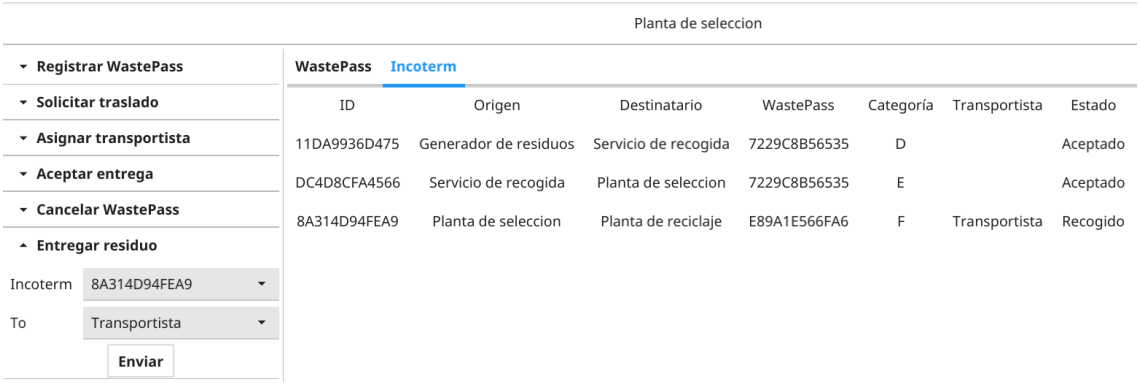


Figura 32. Entrega de residuos de la planta de selección al transportista

El transportista llega a la planta de selección y entrega el lote de residuos. El estado del Incoterm se actualiza a “entregado”.

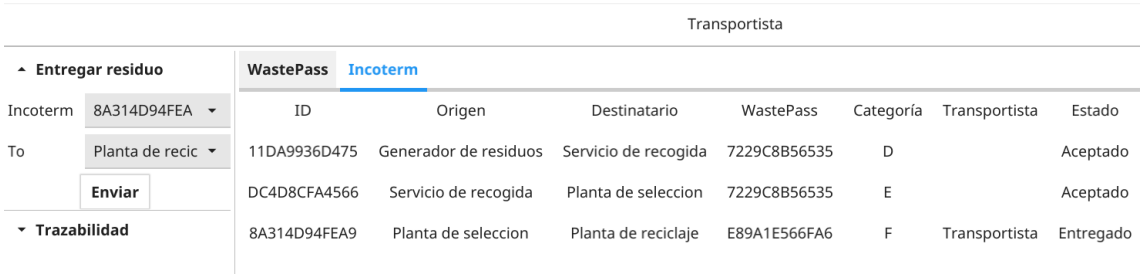


Figura 33. Entrega de residuos del transportista a la planta de reciclaje

Una vez que la planta de reciclaje acepta la entrega, se convierte en el propietario del pasaporte de residuos.

Figura 34. Aceptación de residuos por la planta de reciclaje

La planta de reciclaje completa los procesos de tratamiento de los residuos plásticos y registra un nuevo pasaporte de residuos asociado al lote de residuos plásticos obtenido y al pasaporte de residuos anterior.

Figura 35. Registro de un nuevo pasaporte de residuos por la planta de reciclaje

La planta de reciclaje solicita el traslado al fabricante. En este caso, la categoría del Incoterm es C porque la planta de reciclaje se responsabiliza del traslado. El procedimiento es similar al traslado desde la planta de selección a la planta de reciclaje, por lo que solamente se muestra la solicitud del traslado.

Figura 36. Solicitud del traslado al fabricante por la planta de reciclaje

Una vez el lote de plástico llega a la instalación del fabricante y éste lo acepta, se extrae el documento de la trazabilidad del último pasaporte de residuos (791815C1BAD3), que se adjunta en el Anexo I. Finalmente, cuando se utiliza el lote de plástico para la fabricación de nuevos productos el fabricante ejecuta el proceso de cancelación, que provoca que el propietario del WastePass pase a ser “0x0”.

Figura 37. Cancelación del pasaporte de residuos por el fabricante

7. Análisis de la solución

En este capítulo se analiza la seguridad y la privacidad de la solución, así como su viabilidad en términos económicos y de rendimiento.

7.1. Análisis de seguridad y privacidad

En esta sección se analizan los requisitos de seguridad y privacidad que se garantizan con la solución propuesta. Cabe reseñar que los contratos inteligentes son inalterables una vez que se han desplegado en la blockchain, por lo que en ese estado su funcionamiento no puede ser modifcado. Hay que recordar que los contratos inteligentes en Hyperledger Fabric requieren la aprobación de una mayoría de organizaciones para desplegarse y las organizaciones que quieran utilizarlos deben instalarlos en uno de sus *peers*.

Uno de los requisitos más importantes es el de trazabilidad de cada lote de residuos plásticos registrado en el sistema. Todos los datos de un lote de residuos plásticos se anotan en el pasaporte de residuos y los traslados quedan registrados mediante tokens cuyo propietario es el mismo pasaporte de residuos. Por lo tanto, se garantiza el requisito de trazabilidad. Además, la autenticación del propietario está garantizada porque cada lote de residuos está asociado a su propietario y, antes de realizar cualquier operación sobre éste, se verifica que el que lo está ejecutando sea su propietario.

Otras propiedades como el no repudio, la disponibilidad y la identificación también se garantizan. Al ejecutar una función de un contrato inteligente, se guarda la transacción firmada en la cadena de bloques, por lo que se puede asegurar que la entidad que ha efectuado un determinado proceso posee la clave privada utilizada para la firma. Adicionalmente, la descentralización evita que exista un único punto de falla y, en consecuencia, disminuye el riesgo de que el sistema quede inoperativo o que se pierda información. Además, gracias al uso de Hyperledger Fabric, todas las organizaciones que operan en la red están identificadas con certificados digitales x509.

Otra de las propiedades que se proporciona es la de transferibilidad segura. La transferencia solo puede ser ordenada desde el propietario del lote de residuos plásticos y, una vez que se ha transferido, se actualiza el propietario de este lote. Esto impide al propietario anterior volver a realizar ninguna operación sobre este lote. Tampoco se puede realizar ninguna operación sobre un lote de residuos plásticos cuando ya exista un acuerdo de traslado (Incoterm) vigente debido a que es el propio Incoterm el que actúa

como propietario del WastePass cuando se realiza un traslado de una organización a otra.

En cuanto a la privacidad del sistema, el uso de un canal privado de Hyperledger Fabric garantiza que solo los miembros autorizados de cada organización puedan acceder a la información y ejecutar procesos en su nombre. También garantiza que las transacciones solo se guarden en los nodos de las organizaciones que participan en el canal. Adicionalmente, los procesos internos de cada organización se mantienen privados porque solo se publica la información de los lotes de residuos plásticos y la mínima información necesaria para realizar los traslados pertinentes.

7.2. Análisis de viabilidad

En esta sección se va a analizar la viabilidad de la solución en términos económicos y de rendimiento. Como se utiliza una blockchain privada como es Hyperledger Fabric, los costes económicos por transacción son nulos. No obstante, existen costes económicos derivados de la instalación y el mantenimiento de la infraestructura necesaria (nodos, autoridades de certificación, etc.). En cuanto al rendimiento y a la escalabilidad, estos parámetros dependen de la cantidad de nodos activos en la red, la congestión de los nodos y de la red y las características de hardware (memoria, CPU, etc.) de cada nodo, entre otros datos.

Poniendo el foco en la solución, se realizan algunas mediciones de latencias en la ejecución de los procesos principales. Cabe recordar que estas mediciones tienen en cuenta una red de seis organizaciones con un único nodo *orderer*. El proceso que tarda más es el despliegue del *chaincode* en la red, con un tiempo que oscila los 168 segundos. No obstante, para obtener una aproximación realista habría que dividir este dato entre el número total de organizaciones, debido a que incluye el tiempo de instalación y aprobación del chaincode en cada uno de los seis *peers*. El resultado estimado serían 28 segundos.

La tabla 2 muestra la latencia en la ejecución de cada función, que oscila los 2 segundos por función. Estos valores son aceptables, sobre todo si se comparan con los tiempos de transacción de entre 15 segundos y 5 minutos de Ethereum.

Contratos inteligentes	Funciones	Tiempo de ejecución (s)
WastePass	Initialize	2.1306845
	Burn	2.1220642
Incoterm	Initialize	2.1824204
	AssignCarrier	2.0761357
	Deliver	2.0822342
	AcceptDelivery	2.0823957

Tabla 3. Latencia de la ejecución de cada función

8. Conclusiones y trabajos futuros

Se ha conseguido diseñar una solución para la trazabilidad de residuos plásticos e implementar una PoC partiendo de este diseño, cumpliendo así el objetivo general de este TFM. Además, se han garantizado los requisitos de seguridad y privacidad que se buscaban. Para lograrlo, se ha utilizado EIP-4944, un estándar de nueva generación para tokens compuestos, y el estándar ERC-721, que es el estándar más popular para tokens no fungibles. Hay que destacar que estos estándares son propios de Ethereum, pero en este trabajo se han adaptado a Hyperledger Fabric y, así, se ha obtenido una solución que, además de ofrecer seguridad, también proporciona privacidad.

Agregando a lo anterior, la PoC se ha probado en un entorno local con solo la infraestructura correspondiente a las seis organizaciones que se han definido. Esto ha limitado considerablemente las pruebas para el análisis del rendimiento y de la escalabilidad de la solución. Si bien en esta red de prueba se han obtenido resultados aceptables, el rendimiento y la escalabilidad son parámetros que dependen en gran medida de las especificaciones técnicas de la infraestructura de la red donde se despliegan los contratos inteligentes. Por consiguiente, habría que realizar este análisis sobre red de producción.

La metodología prevista ha sido suficientemente adecuada porque ha permitido tener un prototipo funcional en una fase temprana y, de esta manera, disponer de un mayor grado de flexibilidad para implementar modificaciones y mejoras.

Los impactos positivos expuestos en la sección 1.3 se han logrado. Utilizando una blockchain permisionada como Hyperledger Fabric el consumo energético ha disminuido significativamente. En cuanto a los impactos negativos, se ha mitigado el riesgo de revelar información de la propia organización e información de datos personales del personal de la organización.

Para terminar, se describen un conjunto de posibles mejoras que han quedado pendientes:

- Despliegue de los contratos inteligentes en chaincodes separados.
- Uso de una CA externa para emitir los certificados y realizar controles sobre las operaciones que puede realizar cada organización (por ejemplo, que la administración reguladora emita un certificado a un transportista otorgándole el permiso necesario para transportar residuos).

- Diseño e implementación de procesos seguros para la generación de múltiples pasaportes de residuos a partir de uno solo, así como la agrupación de diferentes pasaportes de residuos en uno solo.
- Mejora del contrato Incoterm para concretar más los tipos, así como incorporar un procedimiento para enlazar los pagos realizados.

Glosario

Autoridad de certificación (CA). Entidad que emite certificados digitales para identificar a los usuarios.

Blockchain. Tecnología emergente que mantiene un registro inmutable y transparente de forma descentralizada en diferentes nodos de la red.

Cartera. Almacén de claves privadas.

Código Identificador de Resina (RIC). Es un sistema de códigos para clasificar el plástico en diferentes tipos.

Contrato inteligente. Programa informático que se ejecuta sobre una blockchain.

Ledger. Es el registro de las transacciones que ocurren en una blockchain.

PoC. Prueba de concepto (*proof of concept*).

Prueba de conocimiento nulo (ZKP). Es un método que permite a una entidad demostrar a otra que una la veracidad de una afirmación sin revelar más información.

RDN. *Relative distinguished names*

TFM. Trabajo de fin de máster.

Tokenización. Representación de activos en formato digital mediante tokens.

Tokens compuestos. Tokens que está compuestos o contienen otros tokens.

Tokens no fungibles (NFT). Tokens que representan activos únicos y diferenciables entre sí.

Bibliografía

[1] Plastics Europe. *Plastics - The Facts 2021,* 2021. Disponible: https://plasticseurope.org/wp-content/uploads/2021/12/Plastics-the-Facts-2021-web-final.pdf

[2] Ahmad, R.W. [Raja Wasim], Salah, K. [Khaled], Jayaraman, R. [Raja] et al. “Blockchain for Waste Management in Smart Cities: A Survey”, IEEE Access, 9, 131520-131541, 2021. Disponible: https://doi.org/10.1007/s12599-020-00656-x.

[3] Sedlmeir, J. [Johannes], Buhl, H.U. [Hans Ulrich], Fridgen, G. [Gilbert] et al. *The Energy Consumption of Blockchain Technology: Beyond Myth.* Business & Information Systems Engineering, 62(6), 599–608, 2020. Disponible: https://doi.org/10.1007/s12599-020-00656-x.

[4] Lenz, R. [Rainer]. “Blockchain Applications for Waste Management - Analysis of Blockchain Use cases in Waste Management and General Guidance for Starting Blockchain Projects”. Disponible: http://dx.doi.org/10.2139/ssrn.3941795.

[5] Blue Room Innovation. “PortNet, blockchain for more sustainable and efficent ports”. Disponible: https://www.blueroominnovation.com/en/portnet-blockchain-puertos/

[6] Vogt, J. [John], Davis, J. [Jonathan]. “The State of Incoterm® Research. Transportation Journal”, vol 59. p. 304 - 324, 2020. Disponible: https://doi.org/10.5325/transportationj.59.3.0304

[7] Víctor Muñoz, Josep Lluis de la Rosa, Andres El-Fakdi, "EIP-4944: Contract with Exactly One Non-fungible Token [DRAFT]," Ethereum Improvement Proposals, no. 4944, March 2022. [Online serial]. Disponible: https://eips.ethereum.org/EIPS/eip-4944.

[8] Hyperledger (2022). Hyperledger Fabric. https://hyperledger-fabric.readthedocs.io/en/latest/

[9] ASTM Internal. “ASTM Plastics Committee Releases Major Revisions to Resin Identification Code (RIC) Standard”, Junio 2013. Disponible: https://newsroom.astm.org/astm-plastics-committee-releases-major-revisions-resin-identification-code-ric-standard

Anexo I. Ejemplo de documento de trazabilidad

En este anexo se presenta el contenido del documento de trazabilidad generado después de efectuar la totalidad de los procedimientos previstos del ciclo de gestión de residuos plásticos. En la primera página aparecen los datos relativos al pasaporte de residuos y al lote de residuos plásticos asociado y en las próximas páginas la información de todos los incoterms generados.

PASAPORTE DE RESIDUOS

INFORMACIÓN SOBRE EL PASAPORTE DE RESIDUOS	
ID	791815C1BAD3
Propietario actual	0x0
Pasaporte de residuos anterior	E89A1E566FA6
INFORMACIÓN SOBRE EL LOTE DE PLÁSTICO	
Batch	aOPmVyelYAQogCYGeGvcVnulHRTZRW7Qx4YQU7wj0Fc=
Contenido	PETE
Pureza (%)	98.6
Cantidad (kg netos)	200.4
INFORMACIÓN SOBRE PASAPORTES DE RESIDUOS ANTERIORES	
Información sobre el pasaporte de residuos E89A1E566FA6	
Batch	ifkqaElxr0RcEfZeNLa4PRuQiHuuQK5R6xWYKkclmWM=
Contenido	PETE
Pureza (%)	76.8
Cantidad (kg netos)	202.6
Información sobre el pasaporte de residuos 7229C8B56535	
Batch	r47fBdfaHC8qYZkk65qpxds4hclFtzyqBEWNkcX8XWw=
Contenido	NC/NP
Pureza (%)	65.4
Cantidad (kg netos)	365.2

RESUMEN DE TRAZABILIDAD		
Origen/Transportista	**Transportista/Destinatario**	**Fecha**
Generador de residuos	Servicio de recogida	2023-01-09 20:49:33
Servicio de recogida	Planta de seleccion	2023-01-09 20:53:27
Planta de seleccion	Transportista	2023-01-09 21:02:00
Transportista	Planta de reciclaje	2023-01-09 21:02:50
Planta de reciclaje	Transportista	2023-01-09 21:07:41
Transportista	Fabricante	2023-01-09 21:09:11

INCOTERM

INFORMACIÓN GENERAL					
Código	11DA9936D475	Categoría	D	Estado	Aceptado
Fecha de inicio	2023-01-09 20:48:09				

INFORMACIÓN RELATIVA AL ORIGEN DEL TRASLADO	
Nombre	Generador de residuos
Localidad	Tarragona
País	ES

INFORMACIÓN RELATIVA AL DESTINO DEL TRASLADO	
Nombre	Servicio de recogida
Localidad	Tarragona
País	ES

INFORMACIÓN DEL TRASLADO		
Origen/Transportista	**Transportista/Destinatario**	**Fecha**
Generador de residuos	Servicio de recogida	2023-01-09 20:49:33

INCOTERM

INFORMACIÓN GENERAL					
Código	DC4D8CFA4566	Categoría	E	Estado	Aceptado
Fecha de inicio	2023-01-09 20:52:07				

INFORMACIÓN RELATIVA AL ORIGEN DEL TRASLADO	
Nombre	Servicio de recogida
Localidad	Tarragona
País	ES

INFORMACIÓN RELATIVA AL DESTINO DEL TRASLADO	
Nombre	Planta de seleccion
Localidad	Barcelona
País	ES

INFORMACIÓN DEL TRASLADO		
Origen/Transportista	**Transportista/Destinatario**	**Fecha**
Servicio de recogida	Planta de seleccion	2023-01-09 20:53:27

INCOTERM

INFORMACIÓN GENERAL					
Código	8A314D94FEA9	Categoría	F	Estado	Aceptado
Fecha de inicio	2023-01-09 21:00:09				

INFORMACIÓN RELATIVA AL ORIGEN DEL TRASLADO	
Nombre	Planta de seleccion
Localidad	Barcelona
País	ES

INFORMACIÓN RELATIVA AL DESTINO DEL TRASLADO	
Nombre	Planta de reciclaje
Localidad	Marsella
País	FR

INFORMACIÓN RELATIVA AL TRANSPORTISTA	
Nombre	Transportista
Localidad	Bruselas
País	BE

INFORMACIÓN DEL TRASLADO		
Origen/Transportista	**Transportista/Destinatario**	**Fecha**
Planta de seleccion	Transportista	2023-01-09 21:02:00
Transportista	Planta de reciclaje	2023-01-09 21:02:50

INCOTERM

INFORMACIÓN GENERAL					
Código	9D5A8E12E4C5	Categoría	C	Estado	Aceptado
Fecha de inicio	2023-01-09 21:06:28				

INFORMACIÓN RELATIVA AL ORIGEN DEL TRASLADO	
Nombre	Planta de reciclaje
Localidad	Marsella
País	FR

INFORMACIÓN RELATIVA AL DESTINO DEL TRASLADO	
Nombre	Fabricante
Localidad	Berlin
País	AL

INFORMACIÓN RELATIVA AL TRANSPORTISTA	
Nombre	Transportista
Localidad	Bruselas
País	BE

INFORMACIÓN DEL TRASLADO

Origen/Transportista	Transportista/Destinatario	Fecha
Planta de reciclaje	Transportista	2023-01-09 21:07:41
Transportista	Fabricante	2023-01-09 21:09:11