



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



FACULTAD
DE INGENIERÍA

TESINA FINAL DE GRADO
Licenciatura en Ciencias de la Computación

**Desarrollo de una aplicación basada en
tecnología blockchain orientada a la
trazabilidad y valorización del vidrio**

Autora: Rocío Belén Corral Mena
Tutor: Dr. Ing. Pablo Javier Vidal

Octubre 2025

Être et Durer
Ser y durar

AGRADECIMIENTOS

Comienzo este trabajo agradeciendo profundamente a todas las personas que, de forma consciente o casual, contribuyeron a su concepción y realización. Este trabajo no habría sido posible sin su apoyo, inspiración y colaboración.

En primer lugar, agradezco a mi tutor de este trabajo, Dr. Ing. Pablo Javier Vidal, por su guía, predisposición y disponibilidad absoluta. Su presencia activa me generó la constancia para llevar este trabajo a término. Agradezco asimismo a la Dra. Ana Carolina Olivera, quien también dio su apoyo, por su buena voluntad y por su ayuda desinteresada en la elaboración y revisión de este trabajo.

Luego, tengo un gran agradecimiento hacia todos los profesores de la facultad, quienes me han formado y apoyado en mi carrera académica y profesional durante estos años. Todos ellos han sido mi mayor motivación para completar la carrera, culminando en este trabajo.

Agradezco también a mis compañeros de la facultad, de trabajo y amigos, que en muchos casos se solapan, por su compañía en este camino, por ser una fuente de inspiración para este trabajo y por su apoyo. Les agradezco especialmente a aquellos amigos que me hicieron dar cuenta que tengo una idea contagiosa y que es la que inspira este trabajo, muchos de mis proyectos y mi forma de ver el mundo.

Por último, mi mayor agradecimiento es hacia mi familia y pareja: mi mayor fuente de apoyo y motivación. Les agradezco aunque lo hagan de forma desinteresada, por su amor, aguante y contención emocional.

Finalmente, este trabajo lo quiero dedicar a todos los profesores de la carrera. Esta es mi forma de retribuirles todo lo que me han enseñado y apoyado a lo largo de estos años. Nuevamente, gracias.

Rocío Belén Corral Mena

Mendoza, Argentina

Octubre 2025

RESUMEN

La trazabilidad permite identificar el origen, las etapas de producción y la distribución de bienes, facilitando la implementación de prácticas de economía circular, donde los residuos se reciclan o reutilizan en lugar de desecharse. En particular, es deseable poder realizar la trazabilidad del vidrio, dado que es un producto que puede ser reciclado o reinsertado en la cadena de suministro de diferentes formas sin perder su calidad.

Para proporcionar un nivel superior de transparencia, seguridad y eficiencia, los sistemas de trazabilidad están comenzando a hacer uso de la tecnología *blockchain*. Esta tecnología permite crear registros inmutables y descentralizados, asegurando la integridad de la información y evitando manipulaciones externas. Además, brinda confianza a los consumidores al garantizar la autenticidad y calidad de los productos, mientras que también permite a las organizaciones que adoptan esta tecnología diferenciarse en el mercado, al demostrar su compromiso con la sostenibilidad y el respeto al medio ambiente.

En este trabajo se desarrolla un prototipo de sistema de trazabilidad de envases de vidrio basado en tecnología *blockchain*, diseñado para registrar y verificar cada etapa de su ciclo de vida, desde la producción hasta su reintroducción en la cadena de valor, facilitando su valorización. Este desarrollo sigue un proceso de ingeniería de software bajo el modelo en V, el cual articula de manera rigurosa las fases de diseño, implementación y pruebas del sistema. A lo largo del presente trabajo, se profundiza en las etapas de análisis de requerimientos, diseño arquitectónico, implementación del prototipo y verificación exhaustiva de sus funcionalidades, con el fin de demostrar la viabilidad y los beneficios de aplicar *blockchain* para una economía circular de vidrio sostenible y transparente.

ABSTRACT

Traceability enables the identification of the origin and various stages of production and distribution of goods, facilitating the implementation of circular economy practices where waste is recycled or reused instead of discarded. In particular, it is desirable to achieve traceability for glass, as it is a product that can be recycled or reinserted into the supply chain in different ways without losing its quality.

To provide a superior level of transparency, security and efficiency, traceability systems are beginning to leverage blockchain technology. This technology allows for the creation of immutable and decentralized records, ensuring data integrity and preventing external manipulation. Furthermore, it fosters consumer confidence by guaranteeing product authenticity and quality, while also enabling organizations that adopt this technology to differentiate themselves in the marketplace by demonstrating their commitment to sustainability and environmental responsibility.

This work develops a prototype blockchain-based glass traceability system, designed to record and verify each stage of its lifecycle, from production to its reintroduction into the value chain, thus facilitating its valorization. This development follows a V-model software engineering process, which articulates the design, implementation and testing phases of the system. The stages of requirements analysis, architectural design, prototype implementation and exhaustive testing of its functionalities are detailed throughout the work, with the aim of demonstrating the viability and benefits of applying blockchain for a transparent and sustainable circular glass economy.

TABLA DE CONTENIDOS

Índice de Figuras	xiii
Índice de Tablas	xvii
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	3
1.3. Estructura general del documento	3
2. Marco teórico	5
2.1. Blockchain	5
2.1.1. Estructura y funcionamiento de una blockchain	7
2.1.2. Contratos inteligentes	13
2.1.3. Desafíos y oportunidades	15
2.2. Economía circular	17
2.2.1. Políticas sustentables orientadas a la economía circular	18
2.2.2. Cadena de suministro	20
2.2.3. Proceso de producción y reciclaje en la economía circular	22
2.2.4. Cadena de suministro del vidrio	25
2.3. Proyectos y trabajos relacionados	27
3. Metodología de trabajo	31
3.1. Planificación del trabajo	31
3.2. Metodología de desarrollo	33
3.2.1. Etapas del proceso de desarrollo	36
3.3. Gestión del proyecto	38
4. Modelado de requerimientos	41
4.1. Definición de dominio	42
4.2. Modelado de casos de uso	46
4.3. Definición de requerimientos	48
4.4. Historias de usuario y planificación	51

5. Diseño de solución	55
5.1. Diseño de arquitectura	56
5.1.1. Capa de datos	58
5.1.2. Capa backend	62
5.1.3. Capa frontend	64
5.2. Diseño de componentes	66
5.2.1. Arquitectura de datos	66
5.2.2. Arquitectura backend	70
5.2.3. Arquitectura frontend	71
6. Implementación	75
6.1. Generación de código	76
6.2. Despliegue	79
6.3. Documentación	80
7. Pruebas	83
7.1. Pruebas unitarias	84
7.2. Pruebas de integración	87
7.3. Pruebas de sistema	88
7.4. Pruebas de aceptación con usuarios	89
8. Conclusiones	93
8.1. Conclusiones del trabajo	93
8.2. Reflexiones finales	94
8.3. Perspectivas futuras	95

Apéndices

A. Contenido externo	101
A.1. Demostración	101
A.2. Código fuente	101
A.3. Documentación técnica	102
A.4. Gestión del proyecto	102
B. Entrevista a Verallia	103
B.1. Transcripción de la entrevista	103
B.2. Análisis y reflexión	107
C. Viaje de investigación	109
D. Flujos de usuario	113
E. Casos de prueba	123
E.1. Pruebas unitarias	123
E.2. Pruebas de integración	125

E.3. Pruebas de sistema	126
E.4. Pruebas de aceptación	129
Bibliografía	133
Glosario	139
Siglas	147

ÍNDICE DE FIGURAS

2.1. Comparación entre el modelo cliente-servidor y el modelo distribuido de <i>blockchain</i>	6
2.2. Estructura básica de una cadena de bloques	8
2.3. Contenido de un bloque en una cadena de bloques	8
2.4. Ejemplo de códigos <i>hash</i> generados a partir de cadenas de texto similares	9
2.5. Esquema de un mecanismo de consenso en una red descentralizada	10
2.6. Proceso de creación de una transacción y un bloque en una <i>blockchain</i>	12
2.7. Ciclo de vida de un contrato inteligente	14
2.8. Comparación entre la economía lineal y la economía circular	18
2.9. Objetivos de Desarrollo Sostenible (ODS) de Naciones Unidas	19
2.10. Componentes de una cadena de suministro	20
2.11. Trazabilidad como herramienta transversal en la cadena de suministro de envases de vino	21
2.12. Ciclo productivo de la economía circular	23
2.13. Usos de la tecnología <i>blockchain</i> en la economía circular	24
2.14. Ciclo de vida de envases de vidrio en un modelo de economía circular	25
3.1. Organización de las actividades del plan de trabajo	32
3.2. Modelo en V	34
3.3. Tablero Kanban	35
3.4. Modelo en V agrupado por etapas	36
3.5. Tablero Kanban en Jira	38

4.1. Etapas del proceso de modelado de requerimientos del prototipo de trazabilidad de vidrio	42
4.2. Etapas del ciclo de vida de los envases de vidrio	43
4.3. <i>Canvas</i> de propuesta de valor para el Productor Primario de vidrio	44
4.4. <i>Canvas</i> de propuesta de valor para el Productor Secundario de bebidas	44
4.5. <i>Canvas</i> de propuesta de valor para el Consumidor	45
4.6. <i>Canvas</i> de propuesta de valor para el Centro de Reciclaje	45
4.7. Diagrama de casos de uso del sistema de trazabilidad de vidrio	47
4.8. Tablero de Jira para la gestión de historias de usuario	53
4.9. Diagrama de Gantt para la planificación del proyecto	54
 5.1. Arquitectura de módulos del sistema de trazabilidad de envases de vidrio basado en <i>blockchain</i>	57
5.2. Funcionamiento de una cadena secundaria sobre Ethereum	61
5.3. Funcionamiento de una API REST	63
5.4. Arquitectura del sistema con librerías elegidas para comunicación entre módulos .	65
5.5. Diagrama de ciclo de vida de los envases de vidrio en el sistema	67
5.6. Diagrama UML del Contrato de Envases (BaseBottlesBatchContract)	67
5.7. Diagrama UML del Contrato de Productos (ProductBottlesBatchContract)	68
5.8. Diagrama UML del Contrato de Reciclaje (RecycleMaterialContract)	68
5.9. Diagrama Entidad-Relación (DER) del modelo de datos	69
5.10. Modelo <i>Clean Architecture</i>	70
5.11. Arquitectura de módulos <i>frontend</i>	72
5.12. Diseño de casos de la barra lateral de navegación de la aplicación para cada rol	72
5.13. Identidad de marca de la aplicación	73
 6.1. Documentación interactiva del <i>endpoint</i> para crear un lote de botellas generada automáticamente a partir de la especificación OpenAPI	81
7.1. Salida de la ejecución de las pruebas unitarias del <i>endpoint</i> de creación de lote en la API <i>backend</i>	85

7.2. Salida de la ejecución de las pruebas de integración del sistema	87
7.3. Caso de prueba documentado para la creación exitosa de un lote de vidrio	88
7.4. Rediseño de la interfaz de usuario de la pantalla de trazabilidad basado en la retroalimentación recibida	90
7.5. Usuarios voluntarios recibiendo las instrucciones del experimento	91
7.6. Usuarios voluntarios interactuando con el prototipo durante prueba guiada	91
C.1. Imágenes capturadas durante el viaje de investigación a Europa	112
C.2. Mensajes asociados a casos de uso inválidos del sistema de depósito (DRS) en Alemania	112
D.1. Flujo de registro de usuarios	114
D.2. Flujo de autenticación de usuarios	115
D.3. Flujo de gestión de usuarios	115
D.4. Flujo de usuario de productor primario	116
D.5. Flujo de usuario de productor secundario	117
D.6. Flujo de usuario de consumidor	118
D.7. Flujo de usuario de reciclador para gestión de envases	119
D.8. Flujo de usuario de reciclador para gestión de lotes de reciclaje	120
D.9. Flujo de seguimiento de envases	121

ÍNDICE DE TABLAS

4.1. Requerimientos funcionales del sistema de trazabilidad de envases de vidrio	48
4.2. Requerimientos no funcionales del sistema de trazabilidad de envases de vidrio	51
5.1. Comparación de plataformas <i>blockchain</i>	60
7.1. Comparación de las etapas de prueba del prototipo de trazabilidad de vidrio	84
7.2. Resumen de las pruebas unitarias implementadas en cada módulo del sistema	86
E.1. Resumen de pruebas unitarias realizadas sobre los contratos inteligentes	124
E.2. Resumen de pruebas unitarias realizadas sobre la API <i>backend</i>	124
E.3. Resumen de pruebas unitarias realizadas sobre la interfaz <i>frontend</i>	125
E.4. Listado de pruebas de integración realizadas sobre el sistema	126
E.5. Resumen de pruebas de sistema realizadas por módulo	127
E.6. Lista de errores hallados e incidencias relevadas en pruebas de sistema	128
E.7. Casos de prueba de seguridad ejecutados sobre el sistema	129
E.8. Resumen de pruebas de carga realizadas sobre el sistema	129
E.9. Listado de participantes en pruebas de aceptación de usuario	130
E.10. Lista de errores hallados e incidencias relevadas en pruebas de aceptación	130

1

INTRODUCCIÓN

Antes de comenzar con la descripción del trabajo realizado, es importante contextualizar el problema abordado y los objetivos que se pretende alcanzar. En este capítulo se introduce el contexto y la motivación del trabajo (Sección 1.1), a continuación se detallan los objetivos (Sección 1.2) y, finalmente, se expone la estructura general del documento con el fin de guiar al lector a través del contenido desarrollado en los capítulos posteriores.

1.1. Motivación

El mundo se enfrenta a un desafío ambiental sin precedentes: la gestión insostenible de los recursos naturales. La producción y consumo masivos de bienes generan un volumen creciente de residuos, lo que pone en riesgo la salud del planeta y el bienestar de las generaciones futuras [34], [36]. En este contexto, la transición hacia una economía circular se presenta como una solución prometedora para mitigar este impacto y construir un futuro más sostenible [4]. Este modelo económico busca maximizar el valor de los recursos a lo largo de su ciclo de vida, minimizando el desperdicio y reintroduciendo los materiales en los sistemas de producción [16], [29]. Sin embargo, el principal desafío para lograr una economía circular radica en la falta de transparencia y trazabilidad dentro de las cadenas de suministro tradicionales.

Esta falta de visibilidad dificulta la capacidad para identificar oportunidades de reutilización y reciclaje, responsabilizar a las industrias por su impacto ambiental y empoderar a los consumidores para que tomen decisiones informadas.

Investigaciones previas han explorado diversas tecnologías para mejorar la trazabilidad de la cadena de suministro, incluidos códigos de barras, identificación por radiofrecuencia (RFID, *Radio Frequency Identification*) y redes de sensores. Estas tecnologías ofrecen cierto nivel de ca-

pacidad de seguimiento; sin embargo, a menudo están limitadas por factores como la falta de estandarización, la fragmentación de información y la vulnerabilidad a manipulaciones [44].

En los últimos años, la tecnología *blockchain* ha surgido como una solución prometedora para abordar estas limitaciones [2], [5]. Sus características principales, como el registro de datos distribuido, la inmutabilidad y la transparencia, la convierten en una plataforma ideal para registrar y rastrear el movimiento de mercancías a lo largo de la cadena de suministro [5]. Diversos estudios han explorado la aplicación de tecnología *blockchain* para la trazabilidad de la cadena de suministro, demostrando su potencial para mejorar la transparencia y la responsabilidad dentro de estos sistemas. Ejemplos de estas aplicaciones incluyen la creación de un registro inmutable del origen de los productos para verificar su autenticidad y combatir la falsificación [9], el rastreo de materiales a lo largo de la cadena de suministro para apoyar una economía circular [5], la optimización de la logística y la gestión de inventario mediante información en tiempo real [46] y la promoción de prácticas sostenibles al identificar productos con menor impacto ambiental [9].

Las investigaciones realizadas hasta el momento reconocen el potencial de *blockchain* para la trazabilidad de la cadena de suministro, pero muchas soluciones propuestas se enfocan únicamente en esta tecnología [5], [9], lo que limita su aplicabilidad en contextos donde se requiere la integración con sistemas de gestión tradicionales y tecnologías complementarias. Además, la mayoría de los estudios se centran en casos de uso específicos, como la industria alimentaria o farmacéutica, dejando una brecha significativa en la aplicación de *blockchain* para mejorar la trazabilidad en otros sectores, como el reciclaje de vidrio.

En Latinoamérica, el vidrio representa el 5 % de los residuos sólidos urbanos [11], y solo el 20 % de este vidrio se recicla [52]. La baja tasa de reciclaje de vidrio en la región se debe a la falta de infraestructura y sistemas de gestión adecuados, así como a la falta de conciencia y educación sobre la importancia del reciclaje. Mejorar la trazabilidad en la cadena de suministro del vidrio facilita su reciclaje, ayudando a promover una economía circular sostenible en la región. Al visibilizar el flujo de materiales, promover prácticas de reciclaje y facilitar la información y procesos a los usuarios, es posible reducir la generación de residuos, disminuir la extracción de materias primas vírgenes y fomentar la reutilización de materiales en la producción de nuevos envases de vidrio.

Teniendo en consideración que la actividad económica principal de la provincia de Mendoza es la producción de vino, la cadena de suministro del vidrio adquiere una relevancia particular al proveer los envases para el embotellado. En este contexto, la falta de trazabilidad en dicha cadena es una problemática local y concreta cuya solución puede tener un impacto significativo en la sostenibilidad de la industria vitivinícola y, por extensión, en la economía regional. Por lo tanto, el presente trabajo se centra específicamente en la cadena de suministro y el reciclaje de los envases de vidrio, dada la importancia de este material recicitable en la economía circular y su papel en el desarrollo de la industria vitivinícola local.

Este trabajo tiene como objetivo desarrollar una solución de trazabilidad de envases de vidrio

basada en la tecnología *blockchain*, con el fin de mejorar la transparencia y la sostenibilidad a lo largo de todo su ciclo de vida. La solución propuesta busca abordar las limitaciones de las tecnologías existentes, al ofrecer una plataforma que permite a los actores de la cadena rastrear y verificar el origen, el movimiento y el estado de los envases. Para lograrlo, se plantea un enfoque abierto que integra la tecnología *blockchain* con el Internet de las Cosas (IoT, *Internet of Things*) y los sistemas de gestión tradicionales. Esta combinación permite aprovechar los datos en tiempo real provenientes de sensores para una visión más completa y confiable del producto, mientras que su compatibilidad con las prácticas comerciales existentes facilita su adopción. En última instancia, se espera que esta implementación represente un modelo factible y práctico para mejorar la trazabilidad de la cadena de suministro, contribuyendo a la transición hacia una economía circular sostenible. De este modo, se busca proporcionar una solución concreta y aplicable en el ecosistema mendocino, que a su vez pueda servir en un futuro como modelo para adaptarse a otras industrias y a una variedad de materiales reciclables.

1.2. Objetivos

El objetivo general de este trabajo consiste en hacer uso de *blockchain* como tecnología de vanguardia para el desarrollo de una aplicación prototípica destinada a mejorar la trazabilidad en modelos de economía circular orientados al reciclaje de vidrio. A partir de este objetivo general, se definen los siguientes objetivos específicos:

- **Objetivo 1:** entender los procesos de adopción de tecnologías tales como *blockchain* y las capacidades actuales en la región para el uso de sistemas de trazabilidad.
- **Objetivo 2:** en lo referido a las Ciencias de la Computación, se busca desarrollar una aplicación prototípica funcional basada en tecnología *blockchain*. Esto permitirá la trazabilidad transparente, segura y en tiempo real de la gestión de residuos, en particular el vidrio, desde su generación hasta su disposición final, con el fin de garantizar el cumplimiento normativo, mejorar la eficiencia operativa y aumentar la confianza entre todos los actores involucrados en el proceso.

1.3. Estructura general del documento

El presente documento se encuentra organizado en capítulos, cada uno de los cuales aborda un aspecto específico del trabajo realizado. En primer lugar, en el Capítulo 2, se introduce el marco teórico, conceptos básicos relacionados con el problema y la tecnología utilizada, para luego continuar con un análisis de las soluciones existentes y los antecedentes académicos relevantes que contextualizan el trabajo. A continuación, en el Capítulo 3 se detalla la planeación del trabajo y la metodología adoptada. En el Capítulo 4 se describe el proceso de modelado de los requerimientos de la solución propuesta. En el Capítulo 5 se describe el diseño del sistema y en el Capítulo 6 se detalla su implementación. Posteriormente, en el Capítulo 7 se presentan

las pruebas realizadas y, finalmente, en el Capítulo 8 se presentan las reflexiones obtenidas, los desafíos encontrados y las perspectivas futuras del proyecto.

Adicionalmente, al final del documento se incluyen una serie de apéndices como lectura opcional. El Apéndice A incluye contenido externo como la demostración del prototipo, el código fuente y la documentación técnica del sistema. El Apéndice B contiene una transcripción de la entrevista realizada a una empresa referente en la industria del vidrio en Mendoza. En el Apéndice C se presenta una minuta de un viaje de investigación realizado para estudiar sistemas de reciclaje y economía circular. El Apéndice D incluye los flujos de usuario detallados para cada uno de los actores del sistema. El Apéndice E presenta los casos de prueba detallados y resultados obtenidos de cada etapa de la verificación del sistema. Finalmente, se incluye un glosario de términos específicos y abreviaturas que pueden resultar útiles para el lector a lo largo del documento.

2

MARCO TEÓRICO

Este capítulo presenta los fundamentos teóricos que sirven de base para este trabajo. Se aborda la estructura y funcionamiento de las cadenas de bloques, destacando sus características de descentralización, transparencia e inmutabilidad, así como los mecanismos de consenso que garantizan su seguridad y eficiencia (Sección 2.1). Además, se analiza el rol de los contratos inteligentes como herramientas para automatizar procesos y gestionar la lógica de negocio en entornos descentralizados (Sección 2.1.2). Asimismo, este capítulo explora los principios de la economía circular, enfatizando su enfoque regenerativo y su capacidad para transformar las cadenas de suministro hacia modelos más sostenibles (Sección 2.2). Se examinan las etapas del proceso de producción y reciclaje, con especial atención a la trazabilidad como habilitadora para garantizar la transparencia y la eficiencia en una economía circular. Luego se incluye un análisis de la cadena de suministro del vidrio en el contexto mendocino, destacando su relevancia estratégica y los desafíos asociados a su implementación en un modelo circular. Finalmente, se explora el estado del arte de la tecnología *blockchain* en la economía circular, identificando las tendencias actuales y las oportunidades de mejora en la trazabilidad y sostenibilidad de la cadena de suministro del vidrio (Sección 2.3).

2.1. Blockchain

Blockchain, o cadena de bloques, es una tecnología emergente y potencialmente disruptiva que permite registrar información de forma segura, transparente y sin intermediarios. Esta tecnología está siendo aplicada para diversos casos de uso y ha sido objeto de creciente atención tanto en el ámbito empresarial como académico desde su aparición en 2008 [32]. La primera y más famosa aplicación de *blockchain* es Bitcoin. Su inventor anónimo, Satoshi Nakamoto, lanzó Bitcoin en 2008 durante la crisis financiera mundial con el objetivo de crear un nuevo tipo de

moneda digital confiable fuera del control de gobiernos, bancos y otras instituciones financieras tradicionales [32]. Desde entonces, la tecnología *blockchain* ha sido adoptada en diversos sectores. Por ejemplo, en el ámbito financiero, se utiliza como un libro contable digital para facilitar transferencias de valor entre pares, sin bancos ni entidades intermedias [9]. Por otro lado, en logística y cadenas de suministro, se aplica para mejorar la trazabilidad y transparencia de los procesos productivos [40]. Esta diversidad de aplicaciones se debe a que la tecnología *blockchain* sirve para registrar información digital de manera segura, transparente e inmutable, lo que la convierte en una herramienta efectiva para abordar problemas de confianza y seguridad en la gestión de datos. Es relevante comprender los motivos de su surgimiento como una tecnología disruptiva en los últimos años, antes de introducir su estructura y funcionamiento.

Para entender la necesidad e impacto de la tecnología *blockchain*, primero es necesario analizar la arquitectura predominante de Internet hasta la actualidad, basada tradicionalmente en un modelo centralizado cliente-servidor (Figura 2.1). En este esquema, los datos son almacenados en servidores administrados por proveedores, quienes actúan como intermediarios de confianza entre los clientes o usuarios. Aunque este modelo ha facilitado el intercambio de información a escala masiva, también trae aparejados ciertos problemas de confianza, seguridad y privacidad [24]. La centralización implica que los usuarios ceden el control y gestión de sus datos a terceros, lo que puede derivar en una dependencia significativa de estas entidades para la integridad y disponibilidad de la información. Ejemplos de esto incluyen la exposición de datos personales privados en ciber-ataques, la interrupción de servicios por fallas en servidores centrales o la censura de contenido por decisiones unilaterales de las plataformas [24]. Por ejemplo, cuando el servidor de Whatsapp deja de funcionar, los usuarios no pueden utilizar la aplicación para comunicarse hasta que el proveedor vuelva a hacerlo funcionar. Otro ejemplo es cuando un proveedor sufre un ciber-ataque y se filtran contraseñas de los usuarios, quienes no tienen conocimiento de las vulnerabilidades que puede tener el proveedor, pero con frecuencia dependen de su servicio.

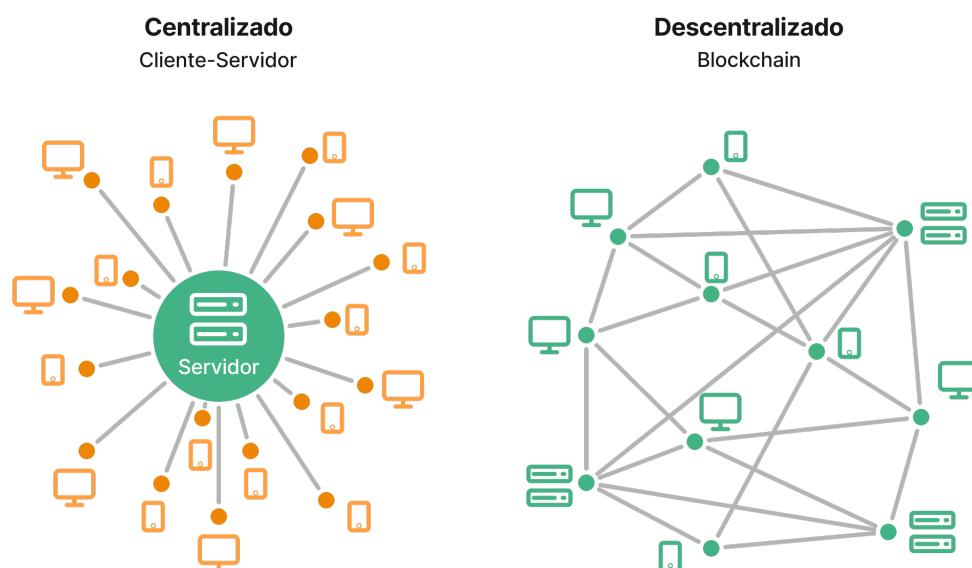


Figura 2.1: Comparación entre el modelo cliente-servidor y el modelo distribuido de blockchain

Ante los desafíos de confianza e integridad en los sistemas centralizados, la tecnología *blockchain* emergió en 2008 como una solución disruptiva. Concebida inicialmente como la base del sistema de criptomonedas Bitcoin [32], la *blockchain* actúa como un libro de contabilidad digital descentralizado que facilita transferencias de valor entre pares sin la necesidad de intermediarios. Esta funcionalidad revolucionaria rápidamente demostró un potencial significativo más allá del ámbito financiero, posicionando a la tecnología *blockchain* como una herramienta para abordar problemas de confianza, transparencia e inmutabilidad en la gestión de datos. Su arquitectura permite registrar información de forma distribuida (Figura 2.1), eliminando la dependencia de proveedores centralizados y habilitando la interacción directa entre múltiples partes [9].

A diferencia de lo que suele suponerse, *blockchain* no se basa en tecnologías radicalmente nuevas, sino que integra de forma innovadora principios existentes de la computación y las matemáticas. Combina conceptos de criptografía (para asegurar la información), redes distribuidas (para la replicación de los datos) y teoría de juegos e incentivos (para coordinar el comportamiento de los participantes y garantizar la seguridad de los datos) [9], [47]. Esta integración produce un sistema seguro, transparente y resistente a manipulaciones, características difíciles de lograr en modelos centralizados. De esta manera, *blockchain* impulsa un nuevo paradigma donde el registro de la información es gestionado colectivamente, lo que permite transacciones y acuerdos entre pares sin depender de un tercero de confianza centralizado.

A continuación, se explorará en detalle la estructura y funcionamiento de la tecnología *blockchain*, sus características distintivas, los mecanismos de consenso que garantizan su seguridad y el papel de los contratos inteligentes como herramientas para la automatización de procesos. Además, se analizarán las ventajas y oportunidades asociadas a su implementación, así como su potencial de uso más allá del ámbito financiero.

2.1.1. Estructura y funcionamiento de una blockchain

La tecnología *blockchain*, o cadena de bloques, es una estructura de datos distribuida y descentralizada donde la información se organiza en transacciones agrupadas en bloques enlazados criptográficamente. Cada bloque posee un código único, conocido como *hash del bloque*, que lo identifica y sirve para enlazarlo al bloque posterior. El *hash* de cada bloque se genera a partir de su contenido y del *hash* del bloque anterior, creando así una cadena continua de bloques interconectados [50]. En la Figura 2.2 se ilustra la estructura simplificada de una *blockchain*, donde cada bloque incluye el *hash* del bloque anterior, formando una cadena de bloques interconectados.

Cada bloque de la cadena consta de un encabezado y un cuerpo [50]. Como se ilustra en la Figura 2.3, el cuerpo guarda la lista de transacciones, mientras que el encabezado contiene metadatos (que pueden variar en cada implementación). Entre los metadatos más relevantes se encuentran el código único del bloque anterior, una marca de tiempo y el código *hash* que identifica únicamente al bloque actual.



Figura 2.2: Estructura básica de una cadena de bloques

El *hash* del bloque se calcula usando funciones *hash* criptográficas, que son algoritmos matemáticos que transforman datos de entrada en una cadena de caracteres de longitud fija. Los códigos *hash* tienen la propiedad de ser rápidos de calcular, difíciles de revertir (es matemáticamente imposible hacerles ingeniería inversa) y únicos para cada conjunto de datos (cualquier cambio en el contenido del bloque generará un código *hash* completamente diferente). Estas características permiten verificar la integridad de los datos almacenados en la cadena, ya que el código *hash* de un bloque se puede recalcular en cualquier momento y comparar con el código almacenado en la cadena [32]. Si los códigos coinciden, se puede confiar en que los datos no han sido alterados; si no coinciden, se ha producido una modificación no autorizada en el contenido del bloque. En la Figura 2.4 se puede ver un ejemplo de los códigos generados usando un algoritmo *hash* a partir de dos cadenas parecidas, donde se observa que incluso un cambio mínimo en el contenido produce un *hash* completamente diferente.

La interconexión criptográfica entre los bloques confiere a *blockchain* su característica de inmutabilidad. Una vez que un bloque es añadido a la cadena, su *hash* se calcula a partir de su contenido y el *hash* del bloque anterior. Cualquier intento de alterar el contenido del bloque invalidaría este *hash* y, por ende, los *hashes* de todos los bloques subsiguientes, rompiendo la integridad criptográfica de la cadena. Este mecanismo permite la detección de cualquier intento de manipulación y la preservación de la integridad histórica del registro [9].

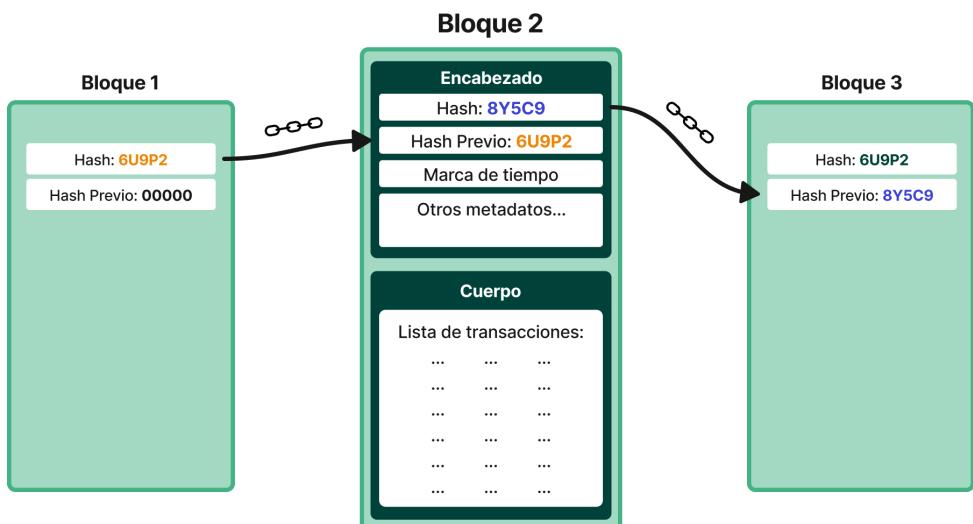


Figura 2.3: Contenido de un bloque en una cadena de bloques

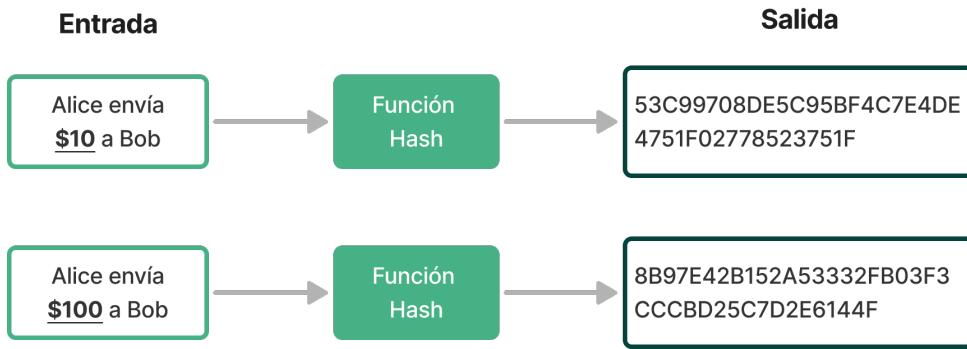


Figura 2.4: Ejemplo de códigos hash generados a partir de cadenas de texto similares

La naturaleza distribuida de la *blockchain* implica que la información no se encuentra almacenada en un único servidor, sino que está distribuida en una red de computadoras interconectadas (conocidas como nodos) y cada nodo de la red mantiene una copia completa y actualizada del registro (de toda la cadena de bloques). Esto asegura su transparencia y resiliencia al no depender de un servidor central [9] propenso a ataques maliciosos y puntos únicos de fallo. Por su parte, la descentralización implica la ausencia de una autoridad central, de modo que la validación y adición de nuevos bloques se rigen por un *mecanismo de consenso* entre todos los nodos participantes de la red.

Precisamente en este marco, los mecanismos de consenso son algoritmos o una serie de reglas que se definen en una red distribuida para que todos los nodos (que en este caso deben guardar exactamente la misma cadena de bloques) se pongan de acuerdo sobre qué información es correcta y válida (Figura 2.5). Sin un mecanismo de consenso, la red sería vulnerable a ataques por parte de nodos maliciosos que puedan esparcir información inválida por la red con fines de beneficio propio, pudiendo perjudicar al resto de la red. Por ejemplo, en la red Bitcoin, cada transacción representa una transferencia de fondos entre cuentas, por lo que un nodo podría difundir transacciones al resto de la red registrando que cientos de usuarios le transfirieron fondos a una cuenta en específico; si el mecanismo de consenso no definiera reglas para validar el origen legítimo de cada transacción, este nodo malicioso podría robar los fondos de los demás usuarios para beneficio exclusivo de quien lo controla [32].

Cada nodo de la red *blockchain* ejecuta un mismo programa computacional (software) que codifica las reglas del mecanismo de consenso, definiendo cómo crear transacciones y bloques válidos, transmitirlos a la red y comprobar la validez de una transacción o bloque recibido de otro nodo antes de agregarlo a su copia local de la cadena (o descartarlo si no es válido) [6]. Para unirse a la red, un nuevo nodo descarga una copia completa de la cadena de bloques existente, lo que le confiere una visión completa del historial de transacciones y el estado actual de la cadena [9]. Posteriormente, el nodo comienza a ejecutar el software del mecanismo de consenso. A partir de entonces, el nodo puede generar nuevas transacciones y transmitirlas a la red para ser recibidas por los demás nodos. Asimismo, el nodo recibe transacciones de otros nodos y las valida mediante el mecanismo de consenso antes de añadirlas a un bloque en su copia local de la cadena [9].

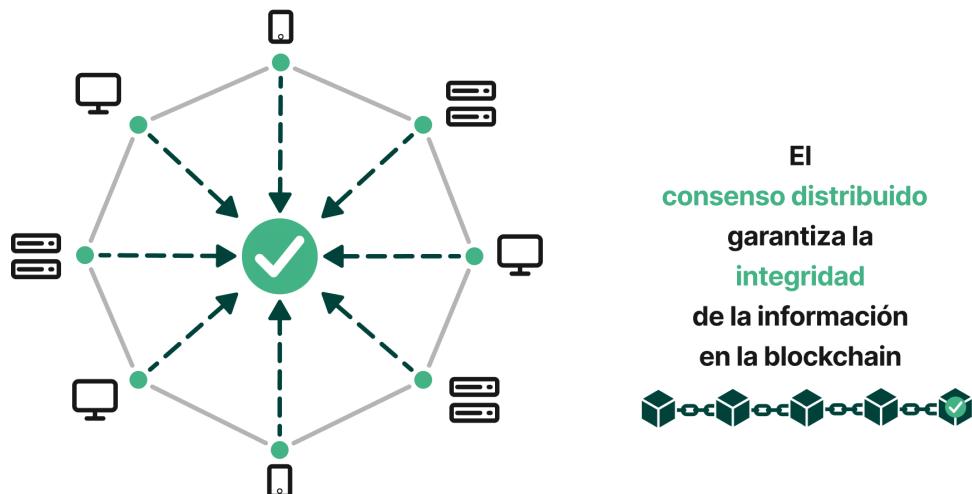


Figura 2.5: Esquema de un mecanismo de consenso en una red descentralizada

En la actualidad, existen múltiples algoritmos de consenso que definen distintas formas de generar un bloque válido (desde un nodo) y comprobar la validez del bloque (recibido desde otro nodo). Cada mecanismo de consenso utiliza diferentes técnicas de teoría de juegos con el objetivo de incentivar a que se unan nuevos nodos a la red mediante recompensas y desincentivar que un nodo actúe de manera maliciosa para obtener un beneficio individual. El mecanismo de consenso garantiza que la red se mantenga segura y resistente a ataques combinando esquemas donde la penalización o pérdida generada por un comportamiento malicioso supere con creces las ganancias que se puedan obtener comportándose de forma maliciosa [32]. Ejemplos prominentes de la diversidad de mecanismos incluyen la Prueba de Trabajo, la Prueba de Participación y la Prueba de Autoridad, cada uno con características particulares.

El mecanismo de Prueba de Trabajo (PoW, *Proof of Work*) es uno de los algoritmos de consenso más conocidos y el primero introducido en las redes *blockchain* por su implementación en Bitcoin [32]. En PoW, todos los nodos compiten para resolver un problema matemático complejo que requiere una gran capacidad computacional. El primer nodo que logra resolver este problema valida la transacción y crea el nuevo bloque incluyendo la solución en su encabezado. El resto de nodos recibe el bloque y comprueba que la solución al problema sea correcta. El nodo que resuelve el reto recibe una compensación en criptomonedas como incentivo por aportar un bloque válido a la red. Aunque la validación de un bloque es de complejidad constante, la alteración de un bloque ya existente implicaría la necesidad de recalcular no solo dicho bloque, sino también todos los bloques subsiguientes. Esto convierte la modificación en un proceso extremadamente costoso en términos de recursos computacionales y energía, lo cual, sumado al rechazo de la red hacia cualquier cadena alterada, desincentiva eficazmente los intentos de manipulación [32]. La desventaja de este algoritmo de consenso es que implica un alto costo energético debido a la carga computacional requerida para resolver el problema [19].

Otro mecanismo ampliamente utilizado es la Prueba de Participación (PoS, *Proof of Stake*), a diferencia de PoW, no requiere de una alta capacidad computacional para la creación de bloques. En PoS, los nodos (conocidos como validadores) “apuestan” una cantidad de criptom-

nidas como garantía de su buen comportamiento. Para generar un bloque válido, el protocolo selecciona pseudo-aleatoriamente un validador para generar el siguiente bloque, con una probabilidad de selección proporcional a la cantidad de criptomonedas que ha apostado. Una vez que el validador seleccionado crea un bloque y lo transmite a la red, los demás nodos validadores de la red simplemente comprueban que el bloque cumpla con las reglas de negocio del protocolo. Si un validador actúa de manera maliciosa, puede perder parte o la totalidad de su participación. La principal ventaja del PoS es su eficiencia energética significativamente mayor en comparación con PoW, ya que no requiere realizar cómputo intensivo. Además, puede ofrecer mayor escalabilidad y tarifas de transacción más bajas. Ethereum es un ejemplo de protocolo *blockchain* que fue implementado originalmente con PoW, pero que se actualizó para utilizar PoS debido a estas ventajas. Sin embargo, una desventaja potencial es el riesgo de centralización si la mayoría de la participación se acumula en pocos nodos, lo que podría darles un control desproporcionado sobre la red [50].

La Prueba de Autoridad (PoA, *Proof of Authority*) es otro mecanismo de consenso donde la validación de bloques se basa en la identidad y reputación de un conjunto pre-aprobado de validadores. Para generar un bloque válido, el protocolo elige una autoridad designada (del conjunto de validadores aprobados y de confianza) que tiene el derecho exclusivo de crear y firmar el nuevo bloque. Para verificar que un bloque es válido, los demás nodos de la red simplemente comprueban la firma digital del validador que lo propuso y que el bloque cumple con las reglas del protocolo. La principal ventaja de PoA es su alta velocidad de transacción, ya que solo un número limitado de validadores de confianza necesita llegar a un consenso. Esto lo hace ideal para redes privadas o consorcios donde la confianza entre los participantes ya existe. Sin embargo, esta misma es su mayor desventaja, ya que la seguridad y el control de la red dependen de un pequeño grupo de entidades conocidas, lo que va en contra del principio de descentralización de muchas *blockchains* públicas [18].

Cada uno de los mecanismos mencionados ofrece distintos niveles de eficiencia de procesamiento, seguridad y descentralización. Durante la validación del bloque, se verifican múltiples aspectos en común en cualquiera de estos mecanismos: la correcta correspondencia del *hash* del bloque anterior con el almacenado en el encabezado del nuevo bloque, la validez de las transacciones de acuerdo a la lógica de negocios propia del protocolo *blockchain* y que el *hash* del bloque propuesto haya sido generado correctamente a partir de la totalidad de su contenido.

En síntesis, la robustez de una *blockchain* reside en la interconexión sinérgica de sus componentes: el encadenamiento criptográfico, la red distribuida y el mecanismo de consenso. No es un único elemento el que garantiza su seguridad e integridad, sino la forma en que estos principios interactúan constantemente. Los mecanismos de consenso, en particular, son los encargados de arbitrar la creación de nuevos bloques de forma segura, resolviendo el problema de la confianza en entornos descentralizados. A su vez, el encadenamiento criptográfico asegura la inmutabilidad de la información, ya que cualquier alteración maliciosa en un bloque de la cadena modificaría su *hash* y rompería la consistencia posterior de la cadena, forzando a cada nodo de la red descentralizada a rechazar el bloque modificado. En consecuencia, para

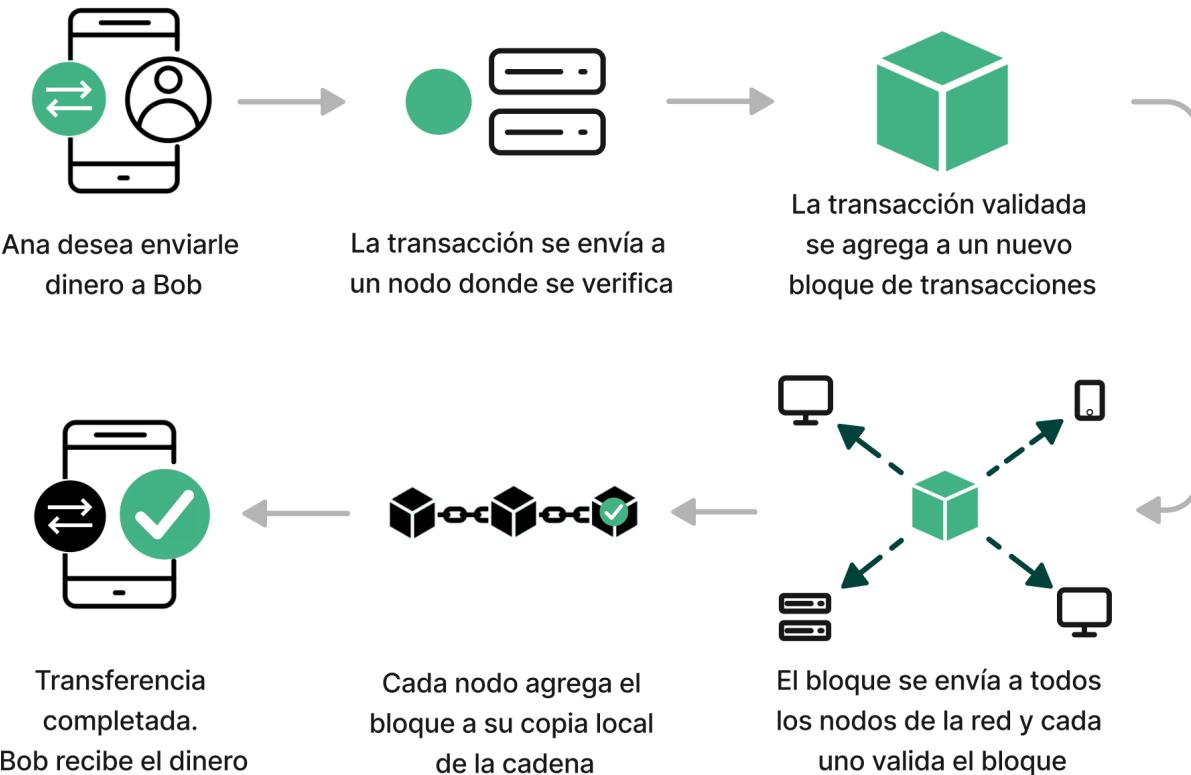


Figura 2.6: Proceso de creación de una transacción y un bloque en una blockchain

recuperar la consistencia de la cadena, sería necesario modificar los *hashes* de todos los bloques posteriores, lo que sería un proceso computacionalmente costoso y fácilmente detectable por el resto de la red. De esta manera, a través de un proceso iterativo de validación, los nodos de la red trabajan de forma coordinada para generar, verificar y distribuir nuevos bloques, conformando un ciclo que asegura la consistencia y la inmutabilidad de la cadena de bloques.

Luego de conocer los elementos fundamentales que conforman una *blockchain*, es posible analizar cómo interactúan para registrar y validar transacciones de manera coordinada en un entorno descentralizado. El proceso se desarrolla de forma secuencial y repetitiva, asegurando que cada bloque añadido a la cadena preserve la integridad del registro global de transacciones. En la Figura 2.6 se presenta un esquema ilustrativo con los pasos del proceso de incorporación de una nueva transacción y su respectivo bloque en una *blockchain*. A continuación, se describen estos pasos en detalle:

1. Un nodo de la red crea y firma una nueva transacción con su clave privada (la firma criptográfica garantiza la autenticidad de la transacción).
2. La transacción se propaga a través de la red distribuida, donde es recibida por cada nodo participante.
3. Cada nodo valida la transacción individualmente, verificando la firma del remitente y asegurándose de que la transacción cumple con la lógica de negocios propia del protocolo *blockchain* (por ejemplo, en Bitcoin, que el remitente cuente con los fondos a transferir). Una vez validada, la transacción se añade a un *pool* de transacciones pendientes. En caso

de ser inválida, simplemente se ignora y se descarta.

4. Cuando un nodo tiene suficiente cantidad de transacciones en su *pool*, procede a seleccionar un conjunto de transacciones pendientes del *pool* para formar un nuevo bloque. Este bloque incluye las transacciones seleccionadas, el *hash* del bloque anterior y otros metadatos (como la marca de tiempo y una serie de datos definida por cada mecanismo). Luego, el nodo calcula el *hash* de este nuevo bloque y, según el algoritmo de consenso, realiza el trabajo necesario para garantizar que sea válido. En el caso de PoW, esto implica resolver un problema criptográfico que requiere una cantidad significativa de potencia computacional. En PoS, el nodo debe demostrar que posee una cantidad suficiente de fondos para participar en la validación del bloque.
5. Una vez que el nodo ha validado el nuevo bloque, lo difunde a la red. Los demás nodos reciben este bloque y verifican su validez (incluyendo el *hash*, las transacciones y la prueba de trabajo/participación). Si el bloque es válido, cada nodo lo añade a su copia local de la cadena de bloques y descarta de su *pool* de pendientes las transacciones incluidas en el bloque. Si el bloque es inválido, es rechazado por cada nodo y no se añade a la cadena.

De esta manera, la cadena de bloques se actualiza de forma continua y descentralizada, asegurando que todos los nodos de la red mantengan una copia idéntica y consistente del registro de transacciones [6]. En su concepción inicial, las transacciones en un bloque se asociaban comúnmente a movimientos financieros [32], pero la flexibilidad inherente de la tecnología *blockchain* permite que los bloques contengan cualquier tipo de información estructurada [6]. Esta versatilidad ha sido el motor para el desarrollo de aplicaciones más complejas, destacando entre ellas los contratos inteligentes [47], que permiten almacenar programas computacionales en la *blockchain*, habilitando la automatización segura de procesos como la trazabilidad, la certificación de origen o la valorización de materiales reciclados.

2.1.2. Contratos inteligentes

A mediados de la década de 1990, cuando el comercio electrónico comenzaba a expandirse y se buscaban mecanismos seguros para realizar transacciones entre personas sin relación previa y sin confianza mutua, el criptógrafo y jurista Nick Szabo propuso el concepto de contrato inteligente [10]. Lo definió como un protocolo informático diseñado para ejecutar de forma automática los términos de un acuerdo, reduciendo la necesidad de intermediarios humanos y de documentos legales tradicionales.

En su concepción original, estos contratos no dependían necesariamente de *blockchain*. Sin embargo, la aparición de esta tecnología a partir de Bitcoin en 2008 y, especialmente, la introducción de Ethereum en 2015, proporcionó por primera vez una infraestructura distribuida, segura e inmutable capaz de materializar la idea de Szabo. Actualmente, un contrato inteligente es un programa almacenado en una *blockchain* que se activa y ejecuta automáticamente cuando se cumplen condiciones predefinidas, garantizando que los acuerdos se lleven a cabo tal como fueron codificados, sin intervención de terceros de confianza [9]. Su función principal es auto-

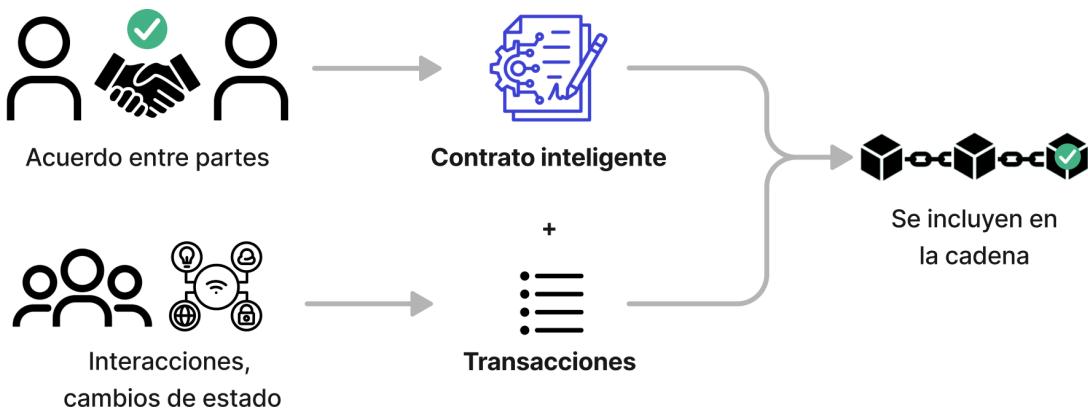


Figura 2.7: Ciclo de vida de un contrato inteligente

matizar procesos en entornos descentralizados, reduciendo la dependencia de intermediarios humanos [55] y mejorando la eficiencia operativa en múltiples sectores [47].

Para codificar las reglas de un contrato inteligente, se emplean lenguajes de programación específicos adaptados a cada plataforma *blockchain* [6]. Un ejemplo prominente es Solidity [48], utilizado en Ethereum, un lenguaje orientado a objetos diseñado específicamente para esta finalidad. Los contratos programados en Solidity pueden interactuar entre sí y con el estado global de la *blockchain*, habilitando la creación de aplicaciones descentralizadas (*dApps*, *Decentralized Applications*) que operan de forma autónoma y sin intermediarios en la red [10].

Un contrato inteligente se concibe como un conjunto de reglas y lógica de negocio codificadas. Cada contrato posee un código (las reglas, por ejemplo, en Solidity) y un estado (la información dinámica almacenada en la *blockchain*) [10]. El código del contrato es inmutable una vez desplegado en la *blockchain* mediante una transacción, garantizando la permanencia de las reglas establecidas. Su estado, sin embargo, puede evolucionar a medida que se interactúa con el contrato a través de transacciones. Es importante destacar que, si bien se describen como auto-ejecutables por su automatismo al cumplir condiciones, su ejecución es llevada a cabo por los nodos de la red que validan las transacciones e integran los cambios de estado en la cadena [10]. Tanto el código como el estado del contrato se almacenan en la *blockchain*, asegurando su transparencia y disponibilidad pública. Por ejemplo, un contrato inteligente podría gestionar un sistema de votación, donde los participantes envían sus votos y el contrato contabiliza automáticamente los resultados al finalizar el periodo de votación. En la Figura 2.7 se ilustra el ciclo de vida de un contrato inteligente, que comienza con un acuerdo entre partes cuyas condiciones se codifican en un contrato inteligente que se sube a la *blockchain* y con el cuál se puede interactuar posteriormente, generando cambios de estado que se registrarán en la cadena mediante transacciones.

Además de las ventajas generales de la tecnología *blockchain*, los contratos inteligentes aportan beneficios y desafíos específicos frente a la automatización de procesos basada en sistemas digitales tradicionales. Entre sus principales ventajas, destacan la capacidad de ejecutar acuerdos de forma automática y verificable sin necesidad de intermediarios, la transparencia y auditabilidad del código y de su historial de transacciones, así como la posibilidad de integrarse con

otros contratos para crear procesos más complejos [6]. Estas propiedades permiten reducir fricciones operativas, agilizar procesos y ofrecer garantías criptográficas de que las reglas se cumplen tal como fueron definidas.

Sin embargo, los contratos inteligentes enfrentan limitaciones inherentes a las infraestructuras *blockchain*, principalmente en términos de escalabilidad [27]. A diferencia de los sistemas centralizados que permiten ejecución paralela y optimización con bases de datos indexadas, los contratos inteligentes operan bajo modelos de ejecución secuencial y replicación completa en cada nodo [48]. Esto impacta directamente su rendimiento y complejiza la implementación de algoritmos avanzados. Desde la perspectiva de la ingeniería de software, el desarrollo de contratos inteligentes introduce restricciones no triviales: el código inmutable, los costos asociados al almacenamiento en la cadena, la ausencia de llamadas externas directas y los modelos de estado global distribuido. Estas particularidades exigen la adopción de nuevas metodologías y prácticas de diseño seguro, control de flujos y validación estática, muchas de las cuales aún se encuentran en proceso de estandarización [11], [48].

En síntesis, los contratos inteligentes constituyen una herramienta computacional que expande las fronteras de la programación distribuida y descentralizada. Si bien su potencial transformador es innegable [48], su desarrollo robusto y seguro representa un desafío activo que abarca múltiples dominios de la computación: desde la teoría de lenguajes formales [25] y la arquitectura de sistemas distribuidos, hasta la verificación de software, la criptografía aplicada y la integración de datos externos confiables [48]. Debido a la aparición de los contratos inteligentes, la tecnología *blockchain* ha trascendido su origen ligado a las criptomonedas para convertirse en un paradigma disruptivo con aplicaciones transversales en múltiples dominios [6], [51]. En la actualidad, los contratos inteligentes se posicionan como un componente fundamental y un impulsor clave de gran parte de las nuevas y complejas soluciones basadas en *blockchain*, especialmente aquellas que buscan automatizar procesos y gestionar la lógica de negocio directamente en la cadena [45]. Si bien representan una tecnología prometedora, aún se encuentran en una etapa incipiente, lo que implica la existencia de numerosos aspectos por perfeccionar [48]. En un contexto más amplio, la tecnología *blockchain*, incluyendo a los contratos inteligentes, ofrece una serie de ventajas fundamentales y limitaciones inherentes que la distinguen de los sistemas de almacenamiento de datos tradicionales.

2.1.3. Desafíos y oportunidades

Las características inherentes de la *blockchain*, detalladas previamente, se traducen en una serie de ventajas que la distinguen de tecnologías tradicionales. La descentralización propia de su diseño y la consecuente eliminación de intermediarios resultan en una mayor confianza [40] y eficiencia operativa al prescindir de autoridades centrales [45]. La arquitectura basada en registros inmutables garantiza transparencia y trazabilidad completa [45], permitiendo un historial verificable de cualquier activo o evento, lo cual es crucial para casos de uso como certificación [6], logística [6], [40] y gestión de residuos [9]. Además, la capacidad de automatización de

aplicaciones mediante contratos inteligentes optimiza la eficiencia y confiabilidad operativa al ejecutar condiciones lógicas de forma autónoma [6]. En conjunto, las propiedades de transparencia, inmutabilidad y seguridad confieren a *blockchain* una aplicabilidad transversal que la consolida como una tecnología habilitadora para la transformación digital en sectores diversos como finanzas, salud, IoT, energía, educación y ciudades inteligentes, entre otros.

Sin embargo, a pesar de sus beneficios, la tecnología *blockchain* también enfrenta desafíos y limitaciones como lo son la escalabilidad y el rendimiento [50]. Las *blockchains* actuales suelen presentar un bajo rendimiento en procesamiento de información en comparación con los sistemas centralizados [5], debido a la necesidad de alcanzar un consenso distribuido entre todos los nodos de la red [50]. Además, existen vulnerabilidades técnicas inherentes a la tecnología, como el ataque del 51 %, el doble gasto y la posibilidad de errores en contratos inteligentes mal programados, que requieren atención constante [19]. La irreversibilidad de las transacciones, si bien es una garantía de seguridad, puede ser problemática ante vulnerabilidades de programación, errores o fraudes, ya que las operaciones registradas no pueden deshacerse [48]. Actualmente, estos desafíos están siendo abordados activamente por la investigación y el desarrollo en la comunidad *blockchain*. La constante evolución de la tecnología y la aparición de nuevas soluciones buscan mitigar estas limitaciones, abriendo el camino para una adopción más amplia [5], [48], [50]. En este contexto de evolución, la tecnología *blockchain* ha demostrado su potencial para transformar diversos sectores y abarcar numerosos casos de uso.

En el sector financiero, *blockchain* ha generado disruptión mediante soluciones para pagos directos (con las llamadas criptomonedas) y transferencias internacionales [6]. A nivel gubernamental, *blockchain* permite la trazabilidad de procesos administrativos y la implementación de sistemas de votación transparentes [51]. En el ámbito de la salud, *blockchain* permite almacenar registros médicos de manera segura y distribuida, mejorando la interoperabilidad entre instituciones y permitiendo a los pacientes tener control de su historial médico [47]. Esta tecnología también se utiliza en la trazabilidad de la cadena de suministro farmacéutica, donde se requiere un alto nivel de confianza y se debe garantizar el cumplimiento normativo [51]. En otros rubros, como la educación, esta tecnología se utiliza para la emisión y verificación de certificados académicos digitales. A su vez, también se está utilizando *blockchain* para crear mercados para el comercio de energía entre pares, mejorando la gestión de certificados de energías renovables y optimizando la trazabilidad de la producción y el consumo energético [47], [51].

En el ámbito de la gestión de la cadena de suministro, *blockchain* proporciona una plataforma confiable para garantizar la trazabilidad, autenticidad y visibilidad en tiempo real de productos y materiales [45], [49]. Empresas como IBM, Maersk y FedEx han implementado soluciones *blockchain* para monitorear inventarios, registrar pagos y reducir disputas logísticas [50]. Otro caso ejemplar es Dervinsa en Argentina, que certifica la calidad de productos derivados de residuos de vinificación, y otras iniciativas que aplican trazabilidad a alimentos y textiles [6]. Dentro de modelos de economía circular, *blockchain* se posiciona como un facilitador para monitorear ciclos de vida de productos y materiales [5], [9]. Diversos tipos de residuos, desde plásticos y vidrio hasta electrónicos y biomédicos, pueden ser gestionados de manera más eficiente.

ciente mediante el uso de contratos inteligentes que automatizan verificaciones e interacciones entre actores de la cadena [5]. Asimismo, han surgido propuestas innovadoras como la generación de pasaportes digitales de productos y esquemas de incentivos sostenibles, promoviendo hábitos de consumo responsables y nuevos modelos de negocio circulares [5].

La necesidad de trazar el flujo de materiales, certificar la autenticidad de los procesos productivos y garantizar la gestión responsable de residuos ha posicionado a *blockchain* como una herramienta protagónica para habilitar modelos circulares sostenibles. En particular, su capacidad para registrar datos inmutables y automatizar interacciones mediante contratos inteligentes permite estructurar sistemas de trazabilidad que mejoran la eficiencia operativa y también fortalecen la confianza entre actores [40], [45]. A continuación, se analizará con mayor profundidad el uso de *blockchain* para trazabilidad de materiales en la cadena de suministro y para la implementación de estrategias de economía circular.

2.2. Economía circular

La economía circular es un enfoque alternativo al modelo económico lineal tradicional, que nace con el objetivo de transformar de manera sostenible la forma en que la sociedad produce, consume y gestiona los recursos naturales [31].

Desde la Revolución Industrial, la economía global ha operado principalmente bajo un modelo lineal de “extraer, producir y consumir”, caracterizado por la explotación de recursos naturales y la generación masiva de residuos [12]. En este modelo económico, los recursos son extraídos de la naturaleza, transformados en productos, consumidos y finalmente desechados al terminar su vida útil. Este enfoque, aunque ha impulsado un crecimiento económico mundial sin precedentes, ha generado sobreexplotación y degradación de ecosistemas. La deforestación, pérdida de biodiversidad, contaminación del agua, generación masiva de residuos y escasez de recursos no renovables son algunas de las consecuencias negativas de este enfoque extractivo a gran escala. A medida que la población mundial y la demanda de recursos continúan creciendo, la insostenibilidad de este modelo a largo plazo se hace cada vez más evidente [4]. En el enfoque lineal, la cadena de suministro de materiales se organiza como un proceso unidireccional: los recursos son extraídos, transformados en productos, consumidos y finalmente desechados. Este modelo ignora el valor residual de los materiales, no contempla mecanismos para reincorporar los productos al ciclo productivo una vez finalizada su vida útil y, en consecuencia, genera una creciente acumulación de residuos. En contraste, la economía circular redefine el papel de la cadena de suministro, transformándola en una red cerrada y regenerativa. Con este enfoque, la cadena se vuelve más dinámica e interdependiente, integrando bucles de retroalimentación entre los diferentes actores de la cadena de valor, incluyendo a consumidores, productores, proveedores y gestores de residuos. El modelo circular propone un sistema en el que los recursos se mantienen en uso durante el mayor tiempo posible, se reciclan y se reutilizan, minimizando la generación de residuos y reduciendo la extracción de materias primas [21]. La diferencia estructural entre la economía lineal y la economía circular



Figura 2.8: Comparación entre la economía lineal y la economía circular

se hace evidente al ilustrar el flujo de recursos en ambos modelos (Figura 2.8).

En la actualidad existen desafíos culturales, normativos y tecnológicos que dificultan la implementación de la economía circular a gran escala. La transición de los sistemas productivos requiere inversión en infraestructura, marcos regulatorios adecuados y políticas de incentivos claros. Asimismo, implica repensar la educación y la formación de trabajadores para adaptarse a nuevas dinámicas laborales [16]. En muchos contextos, como América Latina y el Caribe, también se han identificado limitaciones institucionales y de gobierno que deben ser abordadas para permitir una adopción efectiva del modelo [11]. Esta transición ya se encuentra en marcha. Numerosos países y sectores productivos han comenzado a incorporar principios circulares en sus estrategias de desarrollo, en muchos casos impulsados por marcos regulatorios, acuerdos internacionales y metas vinculadas a la sostenibilidad ambiental [11], [16]. En este contexto, las políticas públicas han asumido un rol central como motores de adopción, ofreciendo instrumentos normativos, fiscales y de gobernanza que facilitan la transformación del sistema económico. A continuación, se analizará un conjunto de políticas sustentables relevantes para avanzar hacia una economía más circular.

2.2.1. Políticas sustentables orientadas a la economía circular

En el proceso de transición hacia modelos de desarrollo más sostenibles, en los últimos años la Unión Europea ha asumido un rol pionero en la implementación de políticas públicas aliñeadas con la economía circular. Iniciativas como el Pacto Verde Europeo y la Ley Europea del Clima han consolidado a Europa como un referente global en materia de sustentabilidad ambiental [20]. Estas políticas no solo promueven la descarbonización de la economía, sino que también introducen principios de circularidad en sectores como la industria, la energía, la movilidad y la gestión de residuos, reconfigurando las cadenas de suministro hacia sistemas más regenerativos, transparentes y trazables.

Sin embargo, el mayor hito internacional en la construcción de una visión compartida sobre sustentabilidad ha sido la adopción de los Objetivos de Desarrollo Sostenible (ODS) de Naciones Unidas en 2015. Este conjunto de 17 objetivos interconectados (Figura 2.9), acompañados por 169 metas y más de 230 indicadores, propone una agenda universal que orienta las políticas públicas hacia un desarrollo económico, social y ambiental equilibrado para 2030 [22].



Figura 2.9: Objetivos de Desarrollo Sostenible (ODS) de Naciones Unidas. Fuente: [33]

El objetivo general de los ODS es erradicar la pobreza, proteger el planeta y garantizar la paz y prosperidad para todas las personas. En relación con la economía circular, se identifican un conjunto de objetivos particularmente relevantes que guían tanto los marcos normativos como las estrategias de innovación en producción, consumo y gestión de residuos:

- **ODS 7: Energía asequible y no contaminante.**¹ Promueve el acceso universal a fuentes de energía limpias, eficientes y modernas, fundamentales para la transición a una economía circular descarbonizada.
- **ODS 11: Ciudades y comunidades sostenibles.**² Plantea la necesidad de gestionar de manera integrada los recursos urbanos, incluyendo residuos, infraestructura y movilidad, en articulación con una trazabilidad eficiente de los flujos materiales.
- **ODS 12: Producción y consumo responsables.**³ Es el núcleo del paradigma circular, impulsando el diseño sostenible de productos, el uso eficiente de recursos, la minimización de residuos y la promoción de modelos de cadena de suministro regenerativos.
- **ODS 13: Acción por el clima.**⁴ Vincula la circularidad con la reducción de emisiones y la adaptación al cambio climático, incentivando políticas que rediseñen los sistemas productivos de alto impacto ambiental.

Los ODS han generado un marco de referencia común que ha influido fuertemente en las agendas de sostenibilidad a nivel global, incluyendo América Latina [30]. Aunque en la región la adopción de políticas circulares aún es incipiente en comparación con Europa, se observan avances significativos en la última década. Por ejemplo, varios países han comenzado a incorporar la responsabilidad extendida del productor, prohibiciones de plásticos de un solo uso y normativas orientadas a la reutilización y reciclado de materiales [11]. Estas políticas buscan

¹ <https://www.un.org/sustainabledevelopment/es/energy/>

² <https://www.un.org/sustainabledevelopment/es/cities/>

³ <https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/>

⁴ <https://www.un.org/sustainabledevelopment/es/climate-change-2/>

reestructurar las cadenas de valor y fomentar prácticas productivas y logísticas compatibles con los principios de circularidad.

En Argentina, la Estrategia Nacional de Consumo y Producción Sostenibles se destaca como el instrumento central para avanzar hacia la economía circular. La estrategia integra medidas normativas, educativas, tecnológicas y financieras, orientadas a fortalecer la sostenibilidad en toda la cadena de producción y consumo. Promueve activamente el uso de tecnologías limpias, la gestión sostenible de recursos y la incorporación de criterios ambientales en compras públicas, reconociendo el rol central de la trazabilidad como mecanismo para garantizar la transparencia, eficiencia y cumplimiento normativo en los sistemas productivos y sus cadenas de suministro [30].

Estas políticas dejan ver que la transformación hacia una economía circular no puede pensarse sin una reconfiguración de las cadenas de suministro, que constituyen la columna vertebral de los sistemas productivos. La implementación de políticas sustentables, tanto en Europa como en América Latina, ha puesto en evidencia la necesidad de contar con mecanismos que permitan monitorear, verificar y optimizar el flujo de materiales a lo largo de todo el ciclo de vida de los productos. En la siguiente sección se abordará con mayor detalle cómo se articula esta relación entre cadenas de suministro y trazabilidad, y cuál es su rol estratégico en la transición hacia un modelo económico circular.

2.2.2. Cadena de suministro

En el contexto de la economía circular, la cadena de suministro asume una nueva lógica de funcionamiento. Pasa de ser una secuencia finita de pasos que culminan con el consumo y disposición del producto, a transformarse en un sistema cíclico, en el cual los productos son diseñados para permanecer en uso el mayor tiempo posible y ser reutilizados, reacondicionados o reciclados [12].

La cadena de suministro constituye el entramado logístico, operativo y estratégico que permite el flujo de materiales, información y recursos desde la extracción de materias primas hasta la llegada de un producto al consumidor final. La Figura 2.10 ilustra este sistema complejo como un conjunto secuencial de etapas que involucra a múltiples actores: proveedores, fabricantes, distribuidores, comerciantes, consumidores y, en el caso del modelo circular, gestores de residuos y reguladores. Su objetivo es garantizar que los bienes y servicios se produzcan y

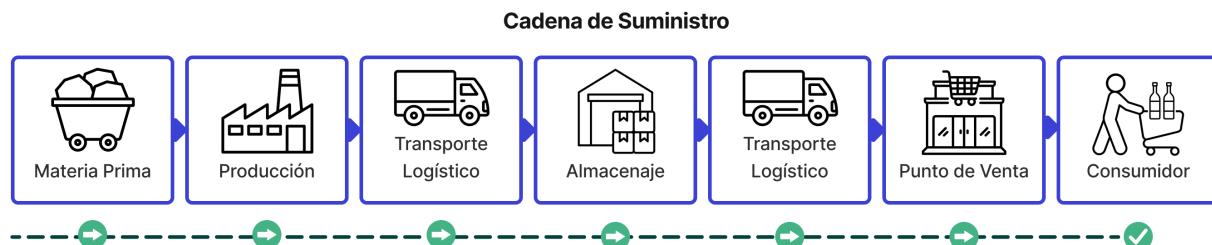


Figura 2.10: Componentes de una cadena de suministro

entreguen de manera eficiente, segura y rentable [42].

A lo largo de los años, las cadenas de suministro se han establecido para maximizar la eficiencia y reducir costos en la producción de productos. Con este objetivo claro, es que se ha dividido el proceso en etapas y se aplican herramientas, procesos y tecnologías diversas para optimizar cada una de ellas, desde la adquisición de materias primas hasta la distribución final. Sin embargo, la maximización de la eficiencia en este modelo lineal, sin preocuparse por el destino del producto luego de su uso, ha generado efectos secundarios adversos en el medioambiente que, como ya se mencionó, han llevado a la necesidad de un modelo sostenible en el tiempo [31].

Para diseñar una cadena de suministro circular, la trazabilidad se posiciona como la herramienta que permite mantener y mejorar la eficiencia y costos, mientras que permite incorporar al final de la cadena las etapas de disposición, reciclaje y reutilización de materiales. La trazabilidad es la capacidad de seguir el recorrido completo de un producto, material o componente a lo largo de toda la cadena de suministro, desde su origen hasta su destino final. Su objetivo principal es reconstruir el historial de producción, transformación y movimiento de un bien, permitiendo conocer su composición, ubicación, responsables y condiciones de manejo en cada etapa del proceso [6]. Por ejemplo, aplicando trazabilidad en la producción de vino se puede conocer la parcela de origen de la uva, la fecha de la vendimia y el proceso de añejamiento, permitiendo al consumidor verificar su autenticidad y al productor identificar rápidamente cualquier problema en un lote. Otro ejemplo frecuente es la trazabilidad de residuos peligrosos, que permite verificar que la disposición final del residuo se hizo correctamente para evitar riesgos de salud o ambientales. La información permite verificar la autenticidad del producto, asegurar estándares de calidad, cumplimiento normativo, eficiencia operativa y sostenibilidad ambiental. Procesos de trazabilidad establecidos permiten también identificar riesgos y oportunidades de mejora en la cadena, permitiendo optimizar la logística, reducir costos y riesgos asociados a errores, fraudes o contaminaciones, y mejorar la capacidad de respuesta ante incidentes o fallas. En la cadena de suministro, la trazabilidad se aplica de forma transversal, es decir, atraviesa e interconecta todas las fases del ciclo: desde el diseño y la fabricación, hasta la distribución, el consumo, la gestión de residuos y el reciclaje [11]. En la Figura 2.11 se ilustra cómo la trazabilidad atraviesa transversalmente todas las etapas del proceso con un ejemplo de la cadena de suministro de envases de vino.

No obstante, la implementación de trazabilidad en la cadena de suministro conlleva desafíos importantes. Las cadenas de suministro tradicionales suelen estar fragmentadas y utilizar sis-



Figura 2.11: Trazabilidad como herramienta transversal en la cadena de suministro de envases de vino

temas de información heterogéneos o poco interoperables [49]. Muchos registros todavía se realizan en papel o en bases de datos centralizadas, lo que aumenta la vulnerabilidad frente a errores humanos, pérdidas de datos o manipulaciones [6]. Además, la ausencia de estándares unificados y la reticencia a compartir datos entre organizaciones limitan la visibilidad total del flujo de productos y materiales. Para abordar estos desafíos, se ha desarrollado un conjunto de tecnologías que fortalecen los sistemas de trazabilidad. Entre las más utilizadas se encuentran los códigos de barras y las etiquetas RFID, que permiten la identificación automática de productos mediante etiquetas físicas; los sensores IoT, que capturan datos en tiempo real sobre condiciones ambientales o de transporte; los sistemas de gestión empresarial y logística digitales, que centralizan y organizan la información operativa; y, más recientemente, la tecnología *blockchain* se está posicionando como solución para unificar a todos los actores de la cadena, aportando una nueva capa de transparencia entre etapas y resolviendo problemas de confianza entre los diferentes actores [56].

La tecnología *blockchain* permite registrar cada transacción o evento de la cadena de suministro en una base de datos digital descentralizada e inalterable. Esto garantiza que todos los actores tengan acceso a un historial común y verificable, reduciendo la necesidad de intermediarios y auditores externos. Combinando contratos inteligentes y plataformas de análisis de datos, la trazabilidad basada en *blockchain* permite no solo conocer lo que ocurrió, sino también automatizar respuestas ante condiciones predefinidas, reduciendo los tiempos de reacción y aumentando la confianza entre las partes [56]. El uso de *blockchain* en este contexto aporta un aumento significativo en la seguridad, transparencia, precisión de datos, eficiencia, responsabilidad y confianza entre actores. Esta tecnología ya se está utilizando para trazabilidad en diversos sectores como la agricultura, alimentos, industria textil y medioambiente. Por ejemplo, en el sector alimenticio, impulsa el valor percibido del producto y la calidad, además de fortalecer la confianza entre las partes interesadas. Para el sector industrial, su uso se enfoca en la planeación y el intercambio de información para una mayor sostenibilidad. Además, es posible combinar tecnología *blockchain* con IoT y otros sistemas digitales ya implementados en la cadena de suministro. Esta combinación puede proporcionar soluciones aún más eficientes para la cadena de suministro, automatizando la recopilación de datos confiables y aumentando los beneficios para las partes interesadas [56]. A continuación, se explorará cómo se articula el proceso de producción y reciclaje en la economía circular y cómo la trazabilidad digital puede potenciar este ciclo.

2.2.3. Proceso de producción y reciclaje en la economía circular

En el marco de la economía circular, los procesos de producción y reciclaje dejan de concebirse como etapas aisladas y unidireccionales para integrarse en un sistema dinámico y regenerativo. Como puede observarse en la Figura 2.12, la cadena de suministro del proceso productivo incluye la gestión de residuos y la reinserción de materiales reciclados en nuevos ciclos productivos [11].

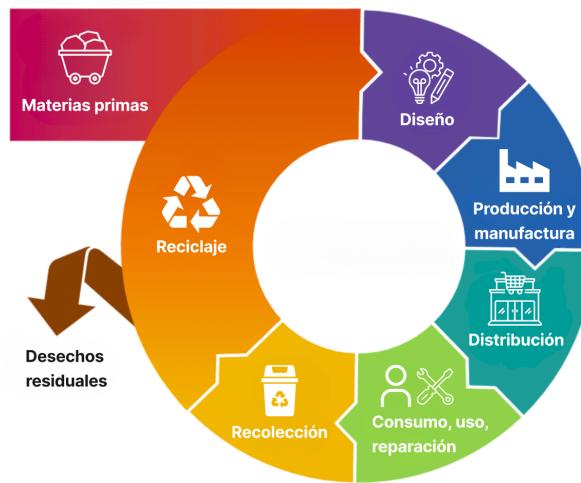


Figura 2.12: Ciclo productivo de la economía circular

El proceso de producción comienza propiamente con la etapa de diseño, donde se decide la composición de los productos considerando criterios de ecoeficiencia, reutilización y reciclabilidad. Aquí intervienen diseñadores, ingenieros y proveedores de materias primas, quienes priorizan materiales reciclados o de bajo impacto ambiental. A continuación, durante la fabricación, los procesos industriales buscan reducir el uso de recursos y minimizar las emisiones, integrando tecnologías limpias y eficientes [31]. En esta fase, los productos terminados o semi elaborados quedan registrados con información detallada sobre su origen, composición y trazabilidad, lo cual permite una futura gestión más eficiente de su reciclaje. Tras su elaboración, los productos son distribuidos a través de canales logísticos que buscan optimizar los costos e impacto ambiental del transporte y almacenamiento.

Una vez que los productos son utilizados por los consumidores, comienza el ciclo inverso de valorización. Cuando estos artículos llegan al fin de su vida útil (y ya no pueden ser reutilizados), se convierten en residuos que deben ser recolectados, transportados, clasificados y reciclados o reacondicionados. Este proceso, conocido de forma genérica como “reciclaje” (sin distinguir si el destino final es reacondicionamiento o reciclaje), involucra a recolectores, centros de aco-gio, plantas de tratamiento, recicladores industriales y fabricantes secundarios [11]. Durante la recolección, tecnologías como sensores IoT, lectores de códigos QR o etiquetas RFID permiten registrar información sobre la identidad del recolector, la cantidad, el tipo y las condiciones del residuo. Esta información permite monitorear flujos de materiales y brindar transparencia en la cadena [56]. En el primer paso del proceso de reciclaje, los residuos son transportados a instalaciones donde se clasifican y segregan según su tipo y calidad. Este paso es fundamental para evitar contaminaciones cruzadas entre materiales distintos y asegurar un reciclaje efectivo. Posteriormente, los materiales seleccionados se someten a procesos de reciclaje o reacondicionamiento, reincorporándolos al sistema productivo como insumos o productos reutilizables. En todo este proceso, tecnologías como *blockchain* pueden registrar cada transacción o transformación del material, documentando la integridad del proceso y fomentando la confianza entre los actores [56].

Aplicaciones de blockchain en las fases de gestión de residuos

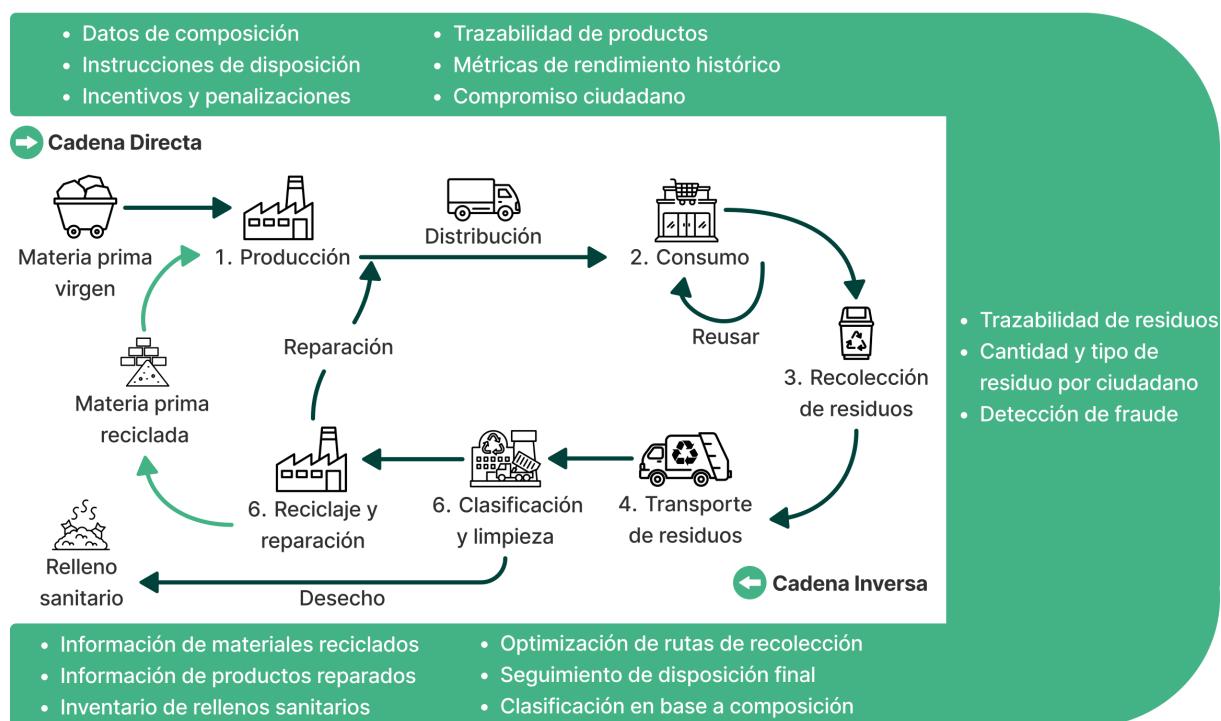


Figura 2.13: Usos de la tecnología blockchain en las etapas de la economía circular [5]

Baralla et al. [5], en su trabajo, ilustran mediante la Figura 2.13 las distintas aplicaciones posibles de la tecnología *blockchain* en cada etapa del ciclo completo de la economía circular. En el sistema esquemático, esta tecnología conecta las etapas en un flujo de información unificado, formando un sistema de trazabilidad digital que permite el seguimiento de los materiales desde su origen hasta su reincorporación al proceso productivo o disposición final, posibilitando la implementación sistemas de incentivos a productores y consumidores por sus prácticas sustentables, calcular métricas de rendimiento y sostenibilidad, hacer trazabilidad de productos y residuos, optimizar rutas de recolección de residuos y proveer información de composición de productos a consumidores y recicladores, entre otras aplicaciones.

El proceso de reciclaje varía en complejidad dependiendo del material. Existen diversos materiales reciclables, cada uno con características particulares. El papel y cartón son fáciles de recolectar y ampliamente reciclados, mientras que los metales (como el aluminio o el cobre) conservan sus propiedades tras múltiples ciclos [11]. Los residuos electrónicos presentan un alto valor por su contenido en metales preciosos, aunque requieren procesos especializados para su desmontaje. Los plásticos representan un desafío por su heterogeneidad, pero pueden reciclarse eficientemente si se rediseñan los envases y se simplifican sus composiciones. Los residuos orgánicos son compostables o pueden aprovecharse energéticamente en caso de no mezclarse con residuos no orgánicos. Finalmente, el vidrio destaca como el material circular por excelencia: puede reciclarse infinitas veces sin perder calidad, su estructura es químicamente estable y su reciclaje requiere menos energía que su producción original [52]. Estas cualidades lo convierten en un insumo ideal para sistemas de economía circular bien diseñados.

2.2.4. Cadena de suministro del vidrio

El vidrio es uno de los materiales más representativos de la economía circular por su capacidad única de ser reciclado indefinidamente sin perder calidad [52]. Esta propiedad lo convierte en un recurso estratégico para reducir la demanda de materias primas vírgenes, minimizar residuos y disminuir la huella de carbono asociada a la producción industrial. A diferencia de otros materiales cuyo reciclaje implica degradación, el vidrio conserva íntegramente sus características físicas y químicas, permitiendo su reintegración al ciclo productivo tantas veces como sea necesario.

En la Figura 2.14 se muestra el ciclo de vida del vidrio en un modelo de economía circular, que abarca desde la extracción de materias primas hasta su reincorporación como materia prima en nuevos productos, exemplificando el caso de los envases de vidrio [54]. El proceso comienza con el diseño del producto, etapa clave para asegurar su durabilidad, reutilización y posterior reciclabilidad. Por ejemplo, en la producción de envases de vidrio, en esta etapa se deciden aspectos como color, forma y composición del envase para optimizar durabilidad, reciclabilidad y aspectos estéticos. Luego del diseño, sigue la producción industrial, donde se funden arena, sosa y caliza a altas temperaturas, frecuentemente combinadas con *calcín* (vidrio reciclado triturado) para reducir el consumo energético y demanda de materiales vírgenes. La especial reciclabilidad del vidrio se hace visible en esta etapa, ya que la calidad del vidrio resultante es la misma sin importar la proporción de *calcín* y de materiales vírgenes utilizados (característica que, por ejemplo, no es igual para el plástico), por lo que los productores de vidrio no encuentran pérdidas potenciales al utilizar materiales reciclados. Luego, los envases fabricados son distribuidos, utilizados para embotellar bebidas (por ejemplo, vino) y adquiridos por los consumidores, quienes (luego de consumir su contenido) pueden reutilizarlos, desartarlos (como basura común) o ingresarlos a circuitos de reciclaje. Para el correcto reciclaje del vidrio (y otros materiales) es importante el circuito de recolección diferenciada, que evita que el vidrio reciclabl se mezcle con otros materiales que lo contaminen e imposibiliten su reciclaje. Los residuos de vidrio son recolectados por empresas especializadas o por los propios consumidores, quienes pueden depositarlos en contenedores específicos para su posterior



Figura 2.14: Ciclo de vida de envases de vidrio en un modelo de economía circular

tratamiento. Una vez recolectado, el vidrio es transportado a plantas de reciclaje donde se clasifica. En la clasificación se separa al vidrio de otros materiales reciclables que puedan haber sido mezclados y se separan los distintos tipos de vidrio (por ejemplo, por color), ya que algunas características, como el color y la composición química, son relevantes para el posterior uso del vidrio reciclado. Luego, el vidrio es triturado y limpiado para eliminar impurezas, como etiquetas o restos de alimentos, dando como resultado lo que se conoce como *calcín*. Esta etapa es crucial, ya que la pureza del material reciclado influye directamente en la calidad del vidrio producido. Finalmente, el *calcín* se funde nuevamente y se convierte en materia prima para nuevos envases, cerrando así el ciclo de vida del vidrio. Este proceso de reciclaje puede repetirse indefinidamente, lo que lo convierte en un modelo ejemplar de economía circular.

La cadena de suministro y reciclaje de envases de vidrio tiene una importancia estratégica en la provincia de Mendoza, por su estrecha vinculación con la industria vitivinícola, uno de los principales motores económicos de la región. La provincia cuenta con una única empresa que produce y recicla envases de vidrio a escala industrial: Verallia⁵. Esta compañía internacional cubre la totalidad de la demanda local de botellas y frascos, fabricando envases para vinos, espumantes, cervezas, licores y alimentos. El proceso de producción en Verallia incluye desde la selección y mezcla de materias primas hasta la formación, inspección y distribución de los envases, con la integración progresiva de vidrio reciclado como parte del insumo.

Verallia ha reconocido públicamente que la mayor dificultad de su industria es la elevada emisión de dióxido de carbono, por lo que ha adoptado una estrategia dual orientada a optimizar el reciclaje y fomentar la reutilización del vidrio. Bajo esta lógica, ha desarrollado el programa “Vidrio, una acción transparente” mediante el cual se promueve la recolección de envases descartados, destinando los ingresos generados al apoyo de organizaciones benéficas [53]. Esta iniciativa, aunque aún incipiente, representa un esfuerzo por avanzar hacia una cadena de suministro más circular y socialmente responsable en la provincia.

Sin embargo, el reciclaje de vidrio en Mendoza enfrenta desafíos estructurales. La tasa de recuperación aún es baja, las métricas oficiales son escasas y las políticas de incentivo son limitadas. La logística de recolección depende en gran medida de la voluntad ciudadana y carece de sistemas obligatorios o premiantes que aseguren su masividad. En este contexto, el rol de actores industriales como Verallia resulta central para impulsar transformaciones sostenibles en la cadena del vidrio, tanto mediante la innovación tecnológica como a través de la articulación público-privada.

Más allá del caso mendocino, el vidrio sigue siendo uno de los materiales más valiosos dentro de una economía circular bien implementada. Su durabilidad, estabilidad química, transparencia y capacidad de reciclaje total lo convierten en un insumo ideal para cerrar ciclos productivos sin pérdidas de calidad ni de valor. Avanzar hacia una cadena del vidrio plenamente circular requiere optimizar cada etapa, desde el diseño y la fabricación hasta la trazabilidad del reciclaje, consolidando sistemas logísticos eficientes, ciudadanos comprometidos y políticas públicas

⁵ <https://ar.verallia.com/>

robustas que garanticen su sostenibilidad a largo plazo. En este contexto, la incorporación de sistemas basados en tecnología *blockchain* ofrece la posibilidad de reforzar la trazabilidad a lo largo de toda la cadena de suministro, facilitando un uso más eficiente y seguro de la información en cada etapa. Esta integración no solo optimiza la operación de cada parte involucrada, sino que también impulsa la adopción de prácticas de economía circular y el uso de *blockchain* en la región. Tomando esta iniciativa como base para el proyecto, a continuación se detallan proyectos y trabajos previos que han explorado la trazabilidad y el reciclaje de vidrio, así como otras iniciativas vinculadas a la economía circular y la sostenibilidad, las cuales aportan soluciones innovadoras para la gestión de residuos y la promoción de prácticas responsables en la cadena de suministro.

2.3. Proyectos y trabajos relacionados

La tecnología *blockchain* se ha posicionado como una herramienta poderosa para mejorar la trazabilidad en cadenas de suministro, ofreciendo registros inmutables y transparentes que permiten rastrear el movimiento de productos desde su origen hasta el consumidor final. Esta característica fortalece la confianza entre los actores y favorece la economía circular, al garantizar la autenticidad e integridad de la información. Numerosos proyectos, tanto académicos como comerciales, han explorado su aplicación en la gestión de residuos y en la trazabilidad de materiales. A continuación, se presentan casos representativos con sus características, beneficios y limitaciones, a fin de identificar lecciones aplicables al desarrollo de un sistema de trazabilidad para el vidrio.

En el ámbito académico, Baralla et al. [5] exploran en su trabajo la trazabilidad de residuos, prevención de fraude e incentivos mediante contratos inteligentes. El modelo planteado (Figura 2.13) permite gestionar tanto la cadena directa (producción y consumo) como la inversa (reciclaje), pero señala la necesidad de enfocarse en categorías específicas de residuos y advierte sobre desafíos de escalabilidad, privacidad y rendimiento en *blockchains* públicas. A su vez, Rahman et al. [38] profundizan en cómo la tecnología *blockchain* permite abordar las vulnerabilidades de los sistemas de gestión de residuos tradicionales que facilitan el fraude, como la falta de transparencia. En este trabajo se introduce el concepto de “Pasaporte Digital de Producto”, un registro digital en la *blockchain* que documenta información clave de un producto a lo largo de su existencia, desde la composición de sus materiales y su origen, hasta su historial de uso y los procesos de reciclaje asociados al final de su vida útil. Este enfoque, según los autores, se potencia al integrarse con tecnología IoT, donde los sensores pueden recolectar datos en tiempo real (por ejemplo, niveles de llenado en contenedores) para que sean registrados de forma segura en la *blockchain*, optimizando así la logística y la eficiencia del sistema.

Dando un paso más allá de la revisión conceptual, Bathaei et al. [7] proponen un marco orientado a la toma de decisiones para que las organizaciones puedan evaluar y priorizar estratégicamente las diferentes soluciones de *blockchain* para la gestión de residuos. Su principal aporte es la aplicación de una metodología mixta que combina el conocimiento de un panel de ex-

pertos con herramientas de decisión multicriterio como el *Best-Worst Method*. A través de este método, el estudio identifica y cuantifica la importancia relativa de los criterios de evaluación, concluyendo que la trazabilidad del material y el cumplimiento regulatorio son los factores más críticos para el éxito de la implementación de *blockchain* en la gestión de residuos. Este enfoque metodológico traslada el debate desde el potencial teórico de la tecnología hacia una guía práctica y validada empíricamente para la toma de decisiones de inversión y desarrollo en la economía circular.

Otro caso académico es el Modelo ZERO de Sandhiya et al. [43] para reciclaje de plásticos, que integra códigos QR, IoT y *blockchain* para proveer un sistema de trazabilidad inmutable y brindar incentivos para reciclar a los consumidores. En este modelo, cada producto lleva un QR único grabado molecularmente, evitando manipulaciones, y se utilizan “contenedores inteligentes” que aceptan solo plásticos válidos, con verificación y clasificación automática en máquinas de reciclaje. El modelo ofrece incentivos monetarios a los consumidores y busca mejorar calidad, transparencia y costos del reciclaje. De forma complementaria, Bhubalan et al. [8] proponen integrar *blockchain* con marcadores moleculares para rediseñar químicamente plásticos y lograr un reciclaje cerrado. Aunque reconocen su potencial técnico, cuestionan la rentabilidad para plásticos de un solo uso debido a los altos costos de implementación. Estos trabajos demuestran que *blockchain* puede integrarse con tecnologías de identificación física para reforzar la trazabilidad, pero el nivel de infraestructura requerida puede limitar su adopción en contextos con recursos más restringidos, como el reciclaje de vidrio en entornos municipales o regionales. A su vez, la lógica de incentivar al consumidor mediante un mecanismo verificable y confiable, presente en el Modelo ZERO, encuentra un ejemplo real y ampliamente adoptado en el programa Pfand [17], el Sistema de Depósito y Retorno (DRS, *Deposit Return Scheme*) de Alemania [35]. Este sistema de DRS no emplea *blockchain*, pero materializa en la práctica el principio fundamental de estos trabajos: ofrecer una recompensa directa y comprobable para asegurar que el material recicitable retorne a la cadena de valor.

La efectividad de este enfoque se confirma con múltiples experiencias regionales e internacionales, que demuestran que la incorporación de incentivos tangibles es un factor clave para fomentar hábitos de reciclaje en los consumidores e impulsar el desarrollo de una economía circular. En España, la aplicación Reciclos [39] ofrece recompensas por el reciclaje de latas y botellas. En Argentina, el proyecto Colmena [15] premia a los usuarios con la criptomoneda JellyCoin, mientras que la aplicación Greenly Points [23] en Mendoza y la organización Reciqlo en Buenos Aires [26] otorgan puntos canjeables por beneficios locales al entregar residuos reciclables en puntos verdes, y este último centra su actividad exclusivamente en la recolección y el reciclaje de vidrio. A nivel global, el proyecto Plastic Bank⁶ de Canadá utiliza la tecnología *blockchain* para ofrecer criptomonedas como incentivo a los recolectores de plástico en regiones empobrecidas, demostrando cómo las recompensas digitales pueden impulsar la participación y aumentar la transparencia en el flujo de residuos [5]. Estas experiencias subrayan que, al ofrecer valor a cambio del material reciclable, se logra una mayor involucración ciudadana y

⁶ <https://plasticbank.com/>

se contribuye de manera significativa a cerrar los ciclos de vida de los productos.

Asimismo, existen múltiples soluciones comerciales que aplican *blockchain* para trazabilidad en cadenas de suministro. Entre ellas se destaca la empresa Signeblock (España) [46] con su producto Gouze, que permite registrar cada paso de los procesos productivos y de distribución en *blockchain*, ofreciendo acceso personalizado a la información mediante un gestor documental unificado. Incorpora digitalización y notarización de procesos, certificación *blockchain* para garantizar inalterabilidad y códigos QR para compartir información del proceso con consumidores. Sus principales limitaciones se relacionan con la interoperabilidad con sistemas existentes y la resistencia organizacional a compartir datos entre actores. Otra solución relevante es Circularise (Países Bajos) [13], una plataforma que ofrece pasaportes digitales de productos para trazabilidad de extremo a extremo y el intercambio seguro de datos. Para cada producto registra origen, composición y datos ambientales respaldados por *blockchain*, facilitando cálculo de huella de carbono. Se integra con sistemas de gestión empresarial existentes y promueve un enfoque abierto e interoperable, utilizando una *blockchain* pública descentralizada para asegurar credibilidad. Por su parte, Circulor (Reino Unido) [14] es una plataforma que proporciona trazabilidad completa desde la fuente hasta el consumidor final, ayudando a demostrar procedencia responsable, reducir emisiones y gestionar riesgos. Se integra con plataformas empresariales existentes, rastrea materias primas y controla el flujo de materiales en la producción. En conjunto, estas plataformas evidencian la viabilidad de *blockchain* para trazabilidad industrial a gran escala, integrando datos ambientales y de origen. No obstante, su enfoque en cadenas de alto valor agregado plantea dudas sobre su adaptación a cadenas de reciclaje de vidrio con márgenes de ganancia reducidos y mayor fragmentación de actores.

Los proyectos y modelos analizados evidencian el vasto potencial de la tecnología *blockchain* para transformar la gestión de residuos y las cadenas de suministro hacia modelos más sostenibles y circulares. Desde sistemas de incentivos simples hasta plataformas complejas de trazabilidad de extremo a extremo, la inmutabilidad, transparencia y descentralización de *blockchain* ofrecen soluciones prometedoras. No obstante, se identifican puntos débiles recurrentes en su aplicación actual, que incluyen la necesidad de un mayor desarrollo técnico y madurez de la tecnología, la superación de barreras organizacionales (como la reticencia a compartir datos), los altos costos de implementación inicial, problemas de escalabilidad en redes públicas y el consumo energético en ciertos algoritmos de consenso, así como la aún incipiente regulación y falta de estándares comunes. Es en este contexto de oportunidades, donde se posiciona el presente trabajo. A pesar de los avances y el reconocimiento de la importancia del reciclaje de vidrio, especialmente en regiones como Mendoza, con una fuerte industria vitivinícola, se observa una fuerte desconexión entre los actores de la cadena de suministro, baja tasa de recuperación (con escasez de métricas oficiales) y débiles políticas de incentivos. Los programas existentes en Argentina, si bien son un paso adelante, aún carecen de mecanismos obligatorios o premiantes que aseguren la masividad de la recolección diferenciada y no garantizan por sí mismos el reciclaje efectivo del material.

Este trabajo busca abordar estas brechas mediante el desarrollo de un prototipo de sistema de

trazabilidad del vidrio basado en tecnología *blockchain*. Enfocado específicamente en la cadena de suministro del vidrio en el contexto mendocino, con el objetivo de integrar a todos los actores del proceso, desde la producción hasta su reintroducción en la cadena de valor. Este sistema se propone permitir el registro y verificación de cada etapa del ciclo de vida del vidrio, mientras que, a su vez, busca superar las limitaciones identificadas en los proyectos preexistentes al ofrecer una solución que facilite la valorización del vidrio, promoviendo una economía circular más transparente, eficiente y sostenible en la región. En este trabajo se hace énfasis en la usabilidad, la integración de datos relevantes y la demostración de los beneficios tangibles para todos los participantes de la cadena.

3

METODOLOGÍA DE TRABAJO

Para la consecución de los objetivos propuestos en este trabajo final, se ha definido una metodología de trabajo que permite planificar y gestionar las diferentes tareas y recursos del proyecto de manera ordenada. Por ello, en este capítulo se presentan los diferentes pasos que se ejecutaron para llegar a cumplir estos objetivos.

Primeramente, se hablará sobre la planificación del trabajo, haciendo hincapié en las actividades necesarias para llevar a cabo el proyecto de forma ordenada y efectiva (Sección 3.1). Posteriormente, se describirá el modelo en V, que es la metodología de desarrollo de software seleccionada como marco de referencia para la organización de las actividades de desarrollo del prototipo tecnológico. Se describirán las etapas del proceso de desarrollo, detallando las actividades y resultados esperados en cada una de ellas a partir de la metodología seleccionada. Finalmente, se profundizará en la gestión del proyecto, haciendo hincapié en las herramientas y prácticas seleccionadas para asegurar una gestión eficiente a lo largo de todas las fases del proceso de desarrollo del prototipo.

3.1. Planificación del trabajo

Definir un plan de trabajo, previo al desarrollo del prototipo tecnológico, es una buena práctica que permite establecer una hoja de ruta concisa que sirve de guía orientativa a lo largo de todo el proceso. El plan de trabajo define los objetivos y actividades necesarios para llevar a cabo un proyecto eficazmente. Para este trabajo en particular, se definió un plan que comprende las actividades necesarias para poder desarrollar un prototipo tecnológico basado en *blockchain*, orientado a la trazabilidad y valorización del vidrio, con el objetivo de contribuir a la economía circular en la región. Este plan sirve de guía para la ejecución de las actividades y la toma

de decisiones a lo largo del proyecto, pero es flexible y puede adaptarse a cambios y nuevas necesidades que puedan surgir durante el desarrollo del trabajo. En la Figura 3.1 se ilustran las actividades que conforman el plan de trabajo junto con su duración estimada y secuencialidad. A continuación se detalla el alcance y los objetivos de cada actividad.

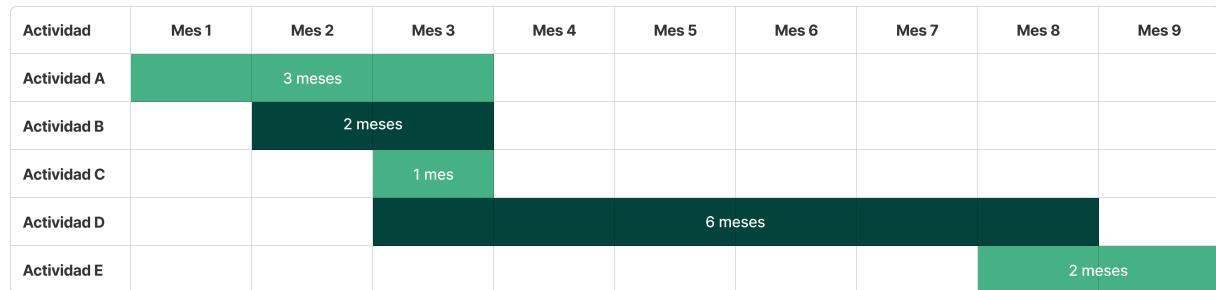


Figura 3.1: Organización de las actividades del plan de trabajo

- **Actividad A:** completar la formación en *blockchain* y las tecnologías y plataformas relacionadas. Los resultados de esta formación se aplican posteriormente en las etapas de diseño de solución e implementación del prototipo tecnológico (comprendidas en la Actividad D).
- **Actividad B:** realizar un estudio pormenorizado del estado actual del arte en todo lo relacionado con *blockchain* en el campo del reciclado. En particular, la búsqueda se orienta al reciclado de vidrio. Se analizan trabajos de la literatura, así como aplicaciones *blockchain* orientadas al reciclaje y la cadena de suministro. Los resultados de este estudio se encuentran documentados en el Capítulo 2: Marco Teórico y en los Apéndices B: Entrevista a Verallia y C: Viaje de Investigación.
- **Actividad C:** definir los procesos de desarrollo del prototipo, haciendo hincapié en la aplicación de los fundamentos de la ingeniería de software y planificar de forma concisa y clara. En la Sección 3.2 se describe la metodología elegida para el proceso de desarrollo del prototipo tecnológico.
- **Actividad D:** desarrollar la aplicación prototipo. Esta actividad comprende las diferentes etapas del proceso de desarrollo de software, desde el análisis de requerimientos, diseño, implementación, validación del prototipo y despliegue. Todos estos pasos se deben ejecutar siguiendo la metodología específica elegida para este trabajo durante la Actividad C, teniendo en cuenta las características particulares del modelo de proceso elegido, con el fin de llevar a cabo el objetivo general de este trabajo.
- **Actividad E:** documentar en una memoria el proceso ejecutado y los resultados del trabajo realizado.

La consecución de estas actividades requiere un marco metodológico que permita gestionar el desarrollo del prototipo de manera eficiente. En la siguiente sección, se detallará la metodología de desarrollo de software elegida para este proyecto y se justificarán los motivos de su selección.

3.2. Metodología de desarrollo

Para desarrollar un software de calidad que cumpla con los objetivos planteados, es necesario seguir una metodología de desarrollo de software que asegure que se sigan buenas prácticas de ingeniería de software para poder entregar un producto funcional, estable, documentado y mantenable dentro de los plazos propuestos. Existen diversas metodologías de desarrollo de software que permiten planificar y ordenar el proceso de desarrollo de software para alcanzar los objetivos del proyecto haciendo un uso eficiente de los recursos disponibles. Cada metodología tiene sus propias características, ventajas y desventajas, y la elección de una metodología apropiada depende de las necesidades específicas del proyecto, tales como el alcance del prototipo, la frecuencia de entrega de resultados, la probabilidad de cambios en los requerimientos durante el desarrollo y la estructura del equipo de trabajo.

Las metodologías de desarrollo de software pueden clasificarse en dos grandes categorías: los modelos prescriptivos o tradicionales, que ofrecen una estructura y un orden definidos para maximizar la previsibilidad y la eficiencia en entornos con requerimientos estables, y los modelos evolutivos o ágiles, que se adaptan mejor a las realidades dinámicas del desarrollo de software moderno al permitir la iteración continua y la flexibilidad frente a los cambios [37]. Cada metodología propone una serie de etapas y prácticas que guían el proceso de desarrollo, como pueden ser la definición de requerimientos, el diseño, la implementación, las pruebas, la documentación, el despliegue y el mantenimiento del software.

Particularmente, en este trabajo se debe elegir una metodología de desarrollo de software que sea apropiada para un equipo unipersonal, ya que el desarrollo del prototipo tecnológico es llevado a cabo por una única persona. A su vez, la metodología debe ser adecuada para proyectos con un alcance definido desde el comienzo y con requerimientos relativamente estables, ya que el objetivo del trabajo es desarrollar un prototipo tecnológico funcional que cumpla con los requerimientos planteados inicialmente. Por último, el proyecto tiene una duración limitada con una única entrega del proyecto completo al final, por lo que la metodología debe permitir una planificación clara y concisa para cumplir con los plazos establecidos, aunque debe ser lo suficientemente flexible para adaptarse a cambios menores que puedan surgir durante el desarrollo del prototipo. Teniendo en cuenta estos factores, se decidió adoptar una metodología híbrida para el desarrollo del prototipo que combina el *modelo en V* (tradicional) con la gestión de tareas de *Kanban* (ágil). El modelo en V aporta un enfoque estructurado para la planificación y documentación del proyecto, mientras que Kanban proporciona flexibilidad para la gestión de tareas y el seguimiento del flujo de trabajo diario en un entorno unipersonal.

El modelo en V (Figura 3.2) es una extensión del *modelo en cascada* que empareja cada fase de desarrollo con una fase de prueba correspondiente [37]. Este modelo propone una estructura de etapas en forma de "V", de modo que a medida que el proyecto avanza hacia abajo en la primera mitad de la V, los requerimientos y componentes del sistema son detallados cada vez más hasta llegar a la codificación, de forma similar al funcionamiento del modelo en cascada, donde cada fase debe completarse en orden sin contemplar posibles errores o cambios. Una vez

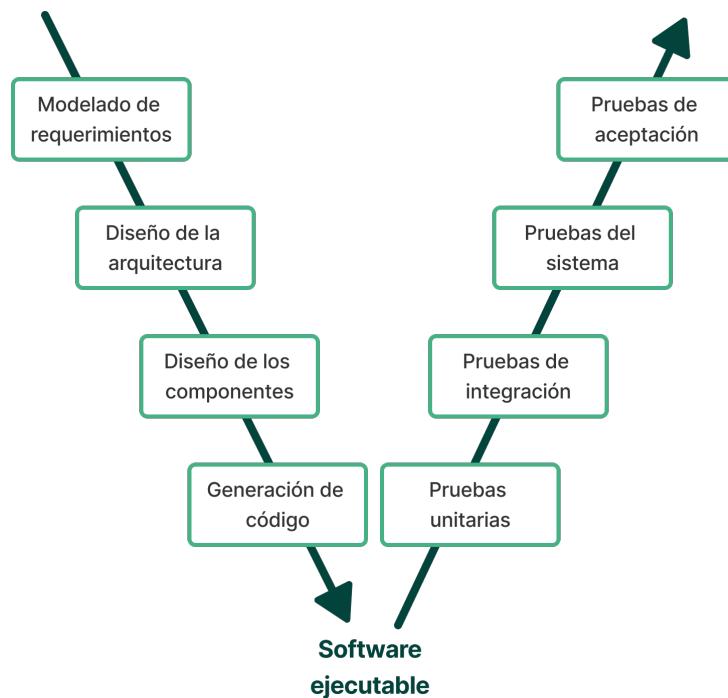


Figura 3.2: Modelo en V. Fuente: [37]

completada la codificación, el proceso asciende por el lado derecho de la V, donde cada etapa de definición anterior es validada a través de pruebas específicas. A diferencia del modelo en cascada, el modelo en V plantea una verificación sistemática en cada etapa, con el fin de detectar y corregir errores de forma temprana y minimizar riesgos al final del proyecto. Este modelo es apropiado para proyectos que requieren altos estándares de calidad y donde los errores tempranos podrían tener consecuencias costosas o críticas. En el caso de un prototipo basado en contratos inteligentes, la inmutabilidad de los contratos desplegados en la *blockchain* hace que la detección temprana de errores sea crítica para garantizar la calidad y la fiabilidad del prototipo final, ya que una vez desplegados, los contratos no pueden ser modificados para resolver errores o vulnerabilidades de seguridad.

Por otro lado, el método Kanban es un enfoque visual ágil para la gestión del flujo de trabajo, cuyo objetivo principal es optimizar la eficiencia al prevenir la sobrecarga de tareas y eliminar cuellos de botella [1]. Este método propone implementar un tablero visual dividido en columnas que representan las diferentes etapas del proceso de desarrollo, por donde se mueven las tareas o tarjetas. En la Figura 3.3 se muestra un ejemplo de un tablero Kanban para un proyecto en curso, donde se pueden ver las tareas en diferentes estados de avance como pendiente, en curso o finalizadas, mientras que las tareas de la columna backlog son aquellas que aún no se han planificado o definido completamente. Los principios de este modelo incluyen limitar el trabajo en curso, visualizar el flujo de tareas y medir su progreso para reconocer oportunidades de mejora. Kanban es una metodología ligera que se adapta bien a la situación de cualquier proyecto, es compatible con otras metodologías de trabajo y su implementación no requiere cambios estructurales en el proceso de trabajo. Sin embargo, su eficacia depende de una disciplina rigurosa y una comunicación constante del equipo para asegurar que el flujo visualizado

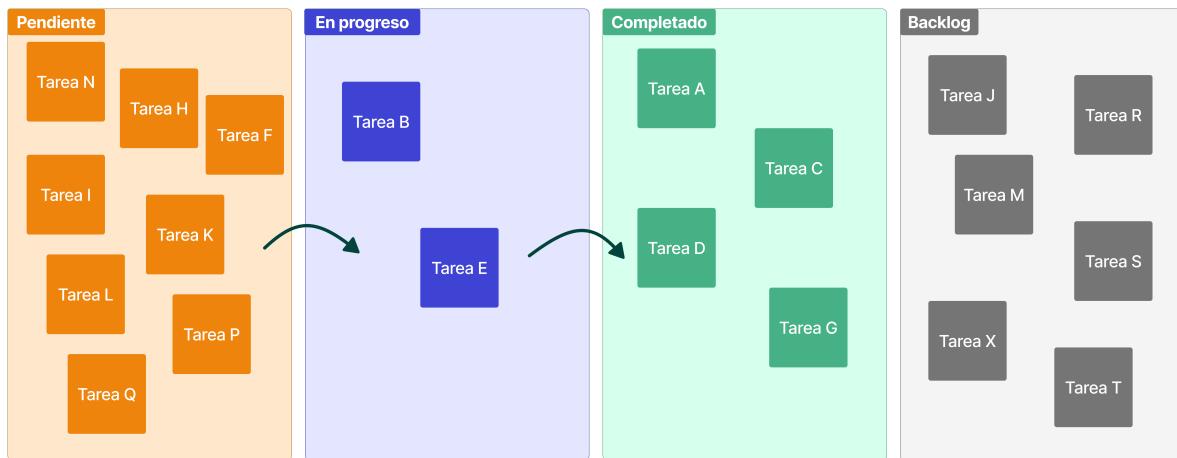


Figura 3.3: Ejemplo de tablero Kanban

en el tablero refleja la realidad actual del estado del proyecto. La naturaleza visual de Kanban facilita el seguimiento del progreso y permite una adaptación ágil a los cambios menores que puedan surgir en el día a día, sin comprometer la estructura general del proyecto [1].

Entre los modelos tradicionales destaca el *modelo espiral*, un enfoque iterativo robusto para gestionar riesgos y adaptarse a entornos inciertos [37]. Este enfoque no se consideró apropiado para este proyecto debido a la complejidad de su proceso evolutivo, ya que el proceso resultaría en una sobrecarga innecesaria para el alcance de este prototipo, dado que los requerimientos del trabajo están claramente definidos y la investigación preliminar ha minimizado la incertidumbre en el proceso. Por su parte, las metodologías ágiles presentan una flexibilidad que las hace adecuadas para proyectos con alta incertidumbre y cambios frecuentes en los requerimientos. Para este proyecto, se analizaron metodologías ágiles como *Scrum*, pero se decidió no utilizar este tipo de metodología ya que su estructura requiere la participación activa de un cliente para guiar reuniones recurrentes para la coordinación del equipo [37], lo que no es aplicable a un proyecto académico individual.

En el contexto del desarrollo de un prototipo tecnológico basado en *blockchain*, la combinación del modelo en V y Kanban representa la opción más estratégica. El modelo en V proporciona la estructura necesaria para un proyecto con requerimientos estables y una necesidad crítica de alta calidad y fiabilidad. La validación rigurosa que propone este modelo, desde el diseño hasta las pruebas finales, asegura que se detecten y corrijan los errores de forma temprana. Por otro lado, Kanban ofrece flexibilidad operativa para gestionar las tareas del día a día de manera visual, lo que facilita el seguimiento del progreso del proyecto y la adaptación a cambios menores en la planificación.

La elección de esta metodología híbrida busca mantener un enfoque sistemático y estructurado, al mismo tiempo que se aprovecha la flexibilidad operativa en el manejo de tareas diarias. El siguiente apartado abordará en detalle cada etapa del proceso de desarrollo en el contexto del modelo en V, desde el modelado de requerimientos hasta las pruebas, para llevar a cabo el desarrollo del prototipo tecnológico.

3.2.1. Etapas del proceso de desarrollo

Utilizando el modelo en V como marco de referencia, el desarrollo del sistema se concibe como un proceso estructurado que se divide en dos grandes fases: la fase descendente, que se enfoca en la definición, el diseño y la implementación, y la fase ascendente, que se centra en la verificación y las pruebas. A continuación, se detallan las actividades y los resultados esperados de cada una de las etapas del proceso de desarrollo, siguiendo el modelo en V, aplicado al prototipo de trazabilidad del vidrio basado en *blockchain*. En la Figura 3.4 se muestran las etapas del proceso de desarrollo en V agrupadas en bloques temáticos, con el fin de ilustrar de manera más clara las interacciones y dependencias entre las diferentes fases de desarrollo de este trabajo, como se describirá a continuación.

Modelado de requerimientos: es la primera fase del proceso, comprende la identificación y documentación detallada de los requerimientos funcionales y no funcionales del prototipo. Los hallazgos se nutren directamente de la investigación realizada como parte de la Actividad B, comprendiendo la revisión del estado del arte en *blockchain*, gestión de residuos y trazabilidad, así como las entrevistas y las observaciones de programas de reciclaje existentes. El objetivo en esta etapa es comprender las necesidades específicas del sistema de trazabilidad del vidrio para definir un conjunto exhaustivo de requerimientos que aseguren que el prototipo aborde los desafíos identificados en la cadena de producción de envases de vidrio en la región. El resultado de esta fase es un documento de requerimientos funcionales y no funcionales que sirve como guía para las siguientes etapas del desarrollo. La ejecución y resultados de esta fase se detallan en el Capítulo 4.

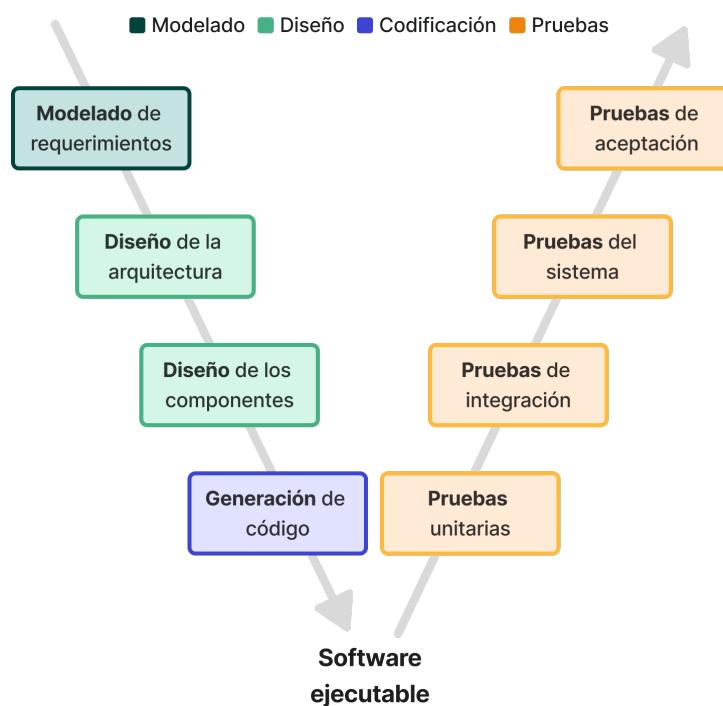


Figura 3.4: Modelo en V agrupado por etapas

Diseño: el diseño del sistema se divide en dos etapas: diseño de arquitectura y diseño de componentes. En la primera etapa, se define la arquitectura general del sistema, incluyendo la selección de tecnologías y plataformas a utilizar, así como la estructura de los módulos principales y la división de responsabilidades entre ellos. En la segunda etapa, se realiza el diseño detallado de cada componente de los módulos del sistema, especificando las interfaces, protocolos de comunicación y estructuras de datos a utilizar. El diseño en etapas y previo a la implementación tiene el objetivo de garantizar que el sistema sea escalable, mantenable y cumpla con los requerimientos definidos en la fase anterior. El resultado es un conjunto de documentos de diseño que guían la implementación del prototipo. En el Capítulo 5 se presentan los detalles de la ejecución de estas dos fases de diseño del sistema.

Generación de código: durante la fase de generación de código o implementación, se lleva a cabo la codificación del prototipo, cada módulo y sus componentes en las tecnologías elegidas y especificaciones detalladas durante la etapa de diseño. Esta etapa es gestionada con el apoyo de Kanban para la organización y seguimiento de las micro-tareas para mantener un flujo de trabajo ágil y adaptable a los cambios menores que puedan surgir durante el desarrollo. El resultado de esta fase es un prototipo funcional que implementa los requerimientos y diseños definidos previamente. En simultáneo con la generación de código se realiza la primera etapa de validación, que consiste en pruebas unitarias de cada componente desarrollado. Estas pruebas aseguran que cada componente funcione correctamente de forma aislada y cumpla con los requerimientos funcionales especificados. Al finalizar esta etapa, el software ejecutable está listo para ser desplegado en un entorno de pruebas. El despliegue del prototipo en un entorno de pruebas accesible públicamente (similar a un entorno productivo real) permite evaluar el comportamiento del sistema en condiciones reales y detectar posibles problemas antes de su uso final. A su vez, el entorno de pruebas es necesario para la realización de pruebas de integración y sistema en las etapas posteriores. Los detalles de la implementación, las pruebas unitarias realizadas y el despliegue se describen en el Capítulo 6.

Pruebas: el proceso de pruebas se lleva a cabo en múltiples etapas. En primera instancia, se implementan pruebas unitarias para verificar cada componente del sistema. Luego, se realizan pruebas de integración automatizadas para verificar que los diferentes módulos del sistema interactúan correctamente entre sí a través de sus interfaces. Estas pruebas aseguran que los datos fluyan adecuadamente dentro del sistema. Posteriormente, se llevan a cabo pruebas de sistema para evaluar el comportamiento del prototipo en su conjunto para validar que el sistema cumple con los requerimientos funcionales y no funcionales definidos. Las pruebas de sistema incluyen pruebas manuales para validar consistencia de datos en el sistema y corroborar que el prototipo cumple con los requerimientos funcionales, junto con pruebas automatizadas mediante código, que permiten verificar los requerimientos no funcionales como rendimiento y seguridad bajo condiciones simuladas. Finalmente, se realizan pruebas de aceptación con un conjunto de usuarios voluntarios para validar que el prototipo cumple con los criterios de aceptación establecidos en la fase de modelado de requerimientos. Los detalles de las pruebas realizadas en estas etapas de pruebas se presentan en el Capítulo 7.

3.3. Gestión del proyecto

Para asegurar una gestión eficiente a lo largo de todas las fases del proceso de desarrollo es necesario implementar herramientas y prácticas de control de proyectos. La gestión del proyecto comprende la planificación, organización y supervisión de todas las actividades relacionadas con el desarrollo del prototipo, asegurando que se cumplan los plazos, se gestionen los recursos de manera efectiva y se mantenga la calidad del trabajo realizado.

Para la gestión de tareas diarias y el seguimiento del progreso del proyecto mediante Kanban, se decidió hacer uso de la herramienta Jira¹. Este software de gestión de proyectos de desarrollo de software provee una funcionalidad de tablero Kanban (Figura 3.5) y permite gestionar el flujo de trabajo, asignar tareas a los miembros del equipo y realizar un seguimiento del progreso durante el desarrollo a través de una aplicación web con una interfaz intuitiva.

A su vez, se determinó utilizar el software Git² para el control de versiones del código fuente, una herramienta que permite un seguimiento detallado y ordenado de los cambios en el código. Utilizando Git es posible revertir cambios, comparar versiones y mantener un historial completo de modificaciones del código fuente, lo que facilita programar de manera ordenada y segura en caso de problemas que puedan surgir durante el desarrollo. De forma complementaria, se utiliza la plataforma GitHub³ para almacenar y gestionar repositorios de código en la nube, lo que permite un acceso remoto y público al código del proyecto. En el Apéndice A se incluye la dirección del repositorio de código fuente en GitHub, entre otros recursos externos relacionados con el proyecto.

The screenshot shows the Jira interface with the 'Tesis LCC' space selected. The left sidebar shows navigation options like 'Para ti', 'Recientes', 'Aplicaciones', 'Espacio', 'Recientes', 'Equipos', and 'Personalizar barra lateral'. The main area displays a Kanban board with three columns: 'POR HACER' (42 items), 'EN CURSO' (2 items), and 'LISTO' (3 items). Each item has a title, a description, an assignee (e.g., SCRUM-1 to SCRUM-12), and a progress bar. The 'EN CURSO' column contains tasks related to user management ('Validar autorización según el rol asignado', 'Ver y editar perfil de usuario'). The 'LISTO' column contains tasks related to user registration ('Registrar usuario con diferentes roles en el sistema', 'Ingresar a la plataforma', 'Mantener sesión de usuario').

Figura 3.5: Tablero Kanban en Jira

¹ <https://www.atlassian.com/es/software/jira>

² <https://git-scm.com>

³ <https://github.com>

Por otro lado, es necesario definir la estrategia de documentación del proyecto de software previamente al inicio del desarrollo. La documentación forma parte del proceso de desarrollo de software y es un entregable en sí mismo, ya que proporciona una referencia clara y confiable sobre el diseño, la implementación y el uso del sistema para los desarrolladores actuales y futuros del proyecto. En este trabajo, se decidió adoptar una estrategia de documentación continua a lo largo de todas las etapas del proceso de desarrollo. Documentar cada fase del modelo en V, desde la definición de requerimientos hasta las pruebas y el despliegue, asegura que toda la información relevante esté disponible para futuras referencias y facilita la comprensión del sistema por parte de otros desarrolladores o partes interesadas. En el caso del código fuente, se combinan dos enfoques: la documentación mediante comentarios en el código y la documentación externa, que comprende documentos separados dentro del repositorio de código fuente, que describen la arquitectura del sistema, instrucciones de instalación, guías de prueba y cualquier otra información relevante para entender y mantener el código del proyecto.

A partir de la selección de la metodología de desarrollo de software y la estrategia de gestión del proyecto, en los próximos capítulos se presentará el proceso de ejecución llevado a cabo en este trabajo para cada una de las etapas planteadas, incluyendo los resultados obtenidos en cada una. En el Capítulo 4 se describirá el proceso de modelado de requerimientos, donde se explica cómo se obtuvieron los requerimientos funcionales y no funcionales, su priorización y la planificación del desarrollo. En el Capítulo 5 se abordará el diseño del sistema, incluyendo la elección de tecnologías, diseño de arquitectura de software, modelado de la interfaz de usuario y definición de los módulos principales del prototipo. Seguidamente, en el Capítulo 6 se detallará la implementación de cada módulo del sistema, integración de los módulos, pruebas unitarias realizadas y despliegue del prototipo en un entorno de pruebas similar a un entorno productivo. En el Capítulo 7 se describirán las pruebas realizadas, tanto las pruebas de integración automatizadas como las pruebas de sistema manuales y las pruebas con usuarios. Finalmente, en el Capítulo 8 se presentarán las conclusiones del trabajo, resultados obtenidos, reflexiones sobre el proceso de desarrollo y oportunidades de mejora de este trabajo con recomendaciones para futuros trabajos relacionados con la trazabilidad y valorización del vidrio mediante tecnologías emergentes como *blockchain*.

4

MODELADO DE REQUERIMIENTOS

El modelado de requerimientos constituye la etapa inicial del lado izquierdo del modelo en V, que enfatiza la importancia de las pruebas en cada fase del ciclo de vida del proyecto de software. El objetivo principal de esta etapa es comprender, documentar y validar las necesidades y expectativas de los actores interesados del sistema, definiendo de forma precisa su comportamiento y funcionalidades. Esta fase se asocia directamente con las pruebas de aceptación del lado derecho de la V, la etapa final del modelo, en la cual se verifica que el sistema cumple con los requerimientos definidos inicialmente.

El modelado de requerimientos incide directamente en las etapas subsiguientes de diseño, implementación y pruebas. Los errores o ambigüedades en la fase de modelado pueden propagarse a lo largo del proyecto, resultando en un aumento del tiempo y los recursos requeridos para corregirlos. Por lo tanto, la inversión de esfuerzo para asegurar que los requerimientos sean claros, completos y factibles reduce el riesgo de inconsistencias y la necesidad de refactorizaciones en fases posteriores, contribuyendo a la ejecución exitosa del proyecto.

En el contexto del desarrollo de un prototipo de aplicación con tecnología *blockchain* para la trazabilidad y valorización de envases de vidrio, el modelado de requerimientos busca garantizar que el prototipo responda a las necesidades específicas de una economía circular sostenible y transparente. Con este objetivo, el proceso de modelado se estructuró en una serie de pasos iterativos para descubrir, definir y refinar los requerimientos del sistema. En la Figura 4.1, se pueden observar las etapas del proceso, las cuales comprenden: definición de dominio, de casos de uso, de requerimientos y de historias de usuario.

El proceso dio inicio con una investigación exhaustiva del dominio del problema para identificar los actores clave y sus interacciones (Sección 4.1). A partir de esta información, se elaboró una propuesta de valor para documentar las necesidades y expectativas de cada actor jun-

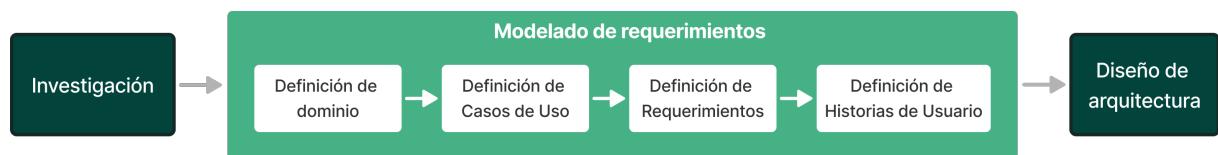


Figura 4.1: Etapas del proceso de modelado de requerimientos del prototipo de trazabilidad de vidrio

to con posibles soluciones. Posteriormente, se modelaron los casos de uso para describir las funcionalidades del sistema desde la perspectiva del usuario (Sección 4.2). Estos casos de uso constituyeron la base para la definición formal de los requerimientos funcionales y no funcionales, incluyendo sus interdependencias (Sección 4.3). Finalmente, se redactaron *historias de usuario* con un nivel de detalle suficiente para definir los criterios de aceptación, lo cual permitió iniciar las etapas de diseño de arquitectura, estimación de esfuerzo y planificación de la implementación.

4.1. Definición de dominio

El modelado de requerimientos inicia con la definición del dominio del problema. En el contexto de este proyecto, que busca la trazabilidad y valorización del vidrio, el objetivo es comprender el entorno en el que el sistema operará, identificando los actores y sus interacciones. El análisis establece la base para la construcción de los requerimientos del sistema.

La definición del dominio se realizó a través de una investigación y revisión de la literatura sobre la trazabilidad del vidrio. En la Sección 2.3, se exploraron los trabajos existentes en el área de *blockchain* aplicada para lograr una economía circular y también se investigaron proyectos existentes que aborden la misma temática o el uso de tecnología para el mismo fin. A su vez, se realizaron entrevistas e investigaciones de campo con expertos en la industria del vidrio y el reciclaje regional, para comprender los procesos actuales, las problemáticas y las expectativas de los actores involucrados (Apéndices B y C). Se examinaron las etapas del ciclo de vida de los envases de vidrio, desde su producción hasta su reintroducción en la cadena de valor, para identificar los puntos donde la tecnología *blockchain* puede aplicarse. La Figura 4.2 ilustra el ciclo de vida de los envases de vidrio, que comienza con la extracción de materia prima para la fabricación de los envases, los cuales son vendidos a productores de bebidas, quienes comercian sus productos a los consumidores. Una vez que estos últimos consumen el producto, el envase de vidrio se convierte en un residuo, que debe ser recolectado, clasificado, limpiado y triturado para luego ser utilizado en la fabricación de nuevos envases.

En este marco, se identificaron los siguientes actores clave:

- **Productor de vidrio (Productor Primario):** fabricante del envase de vidrio, con la responsabilidad de registrar la información inicial del lote de material.
- **Productor de vino (Productor Secundario):** empresa productora de vino que utiliza el envase de vidrio para embotellar sus productos y requiere acceder a la información de

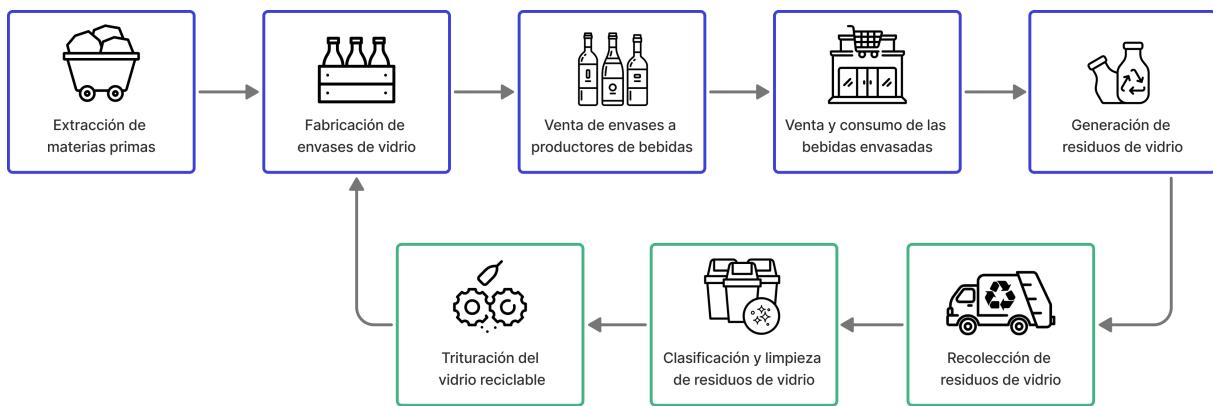


Figura 4.2: Etapas del ciclo de vida de los envases de vidrio

trazabilidad con el fin de garantizar la calidad y seguridad alimentaria.

- **Consumidor:** usuario final que adquiere bebidas envasadas, las consume y puede participar en el proceso de reciclaje.
- **Centro de reciclaje:** entidad que recibe, procesa y recicla el vidrio, utiliza la información de trazabilidad para verificar la calidad del material reciclado y dejar registro de su disposición final.

Dado que el público objetivo del sistema incluye a todos los actores involucrados en la cadena de valor de los envases de vidrio, se planteó realizar una propuesta de valor que incluya a todos ellos, teniendo en cuenta sus procesos, necesidades y expectativas sobre una solución tecnológica. Para ello, se utilizó la herramienta *canvas de propuesta de valor*¹, un diagrama semi-estructurado que permite documentar de manera flexible las necesidades y problemáticas de cada actor, facilitando la comprensión de sus expectativas. Esta herramienta suele ser utilizada por empresas y diseñadores de forma estratégica para analizar, evaluar y ajustar la propuesta de valor de su producto o servicio para alinearla con las necesidades de sus clientes.

Un *canvas* de propuesta de valor se divide en dos secciones principales: el perfil del actor y el mapa de valor de la solución. En el perfil del actor, se detallan sus necesidades, deseos y miedos, lo que proporciona una visión clara de sus motivaciones y los obstáculos que enfrentan en su estado actual. En el mapa de valor de la solución, se definen las funcionalidades y experiencia que ofrecerá la solución, con el objetivo de aplacar los miedos y satisfacer las necesidades y los deseos de los actores. Este análisis conjunto de las expectativas y la solución permite adoptar un enfoque centrado en el usuario desde el inicio del proceso de diseño de solución, asegurando que el sistema responda directamente a las problemáticas identificadas.

La propuesta de valor de la solución de trazabilidad se analiza a través de cuatro *canvas* de propuesta de valor, elaborados para cada uno de los actores clave de la cadena. Este análisis sirve como el primer paso para conceptualizar cómo el sistema basado en *blockchain* puede mitigar los problemas existentes y generar valor tangible para cada participante. El *canvas* de propuesta de valor para el Productor Primario de vidrio (Figura 4.3) y el del Productor Secundario de

¹ <https://www.interaction-design.org/literature/topics/value-proposition-canvas>

bebidas (Figura 4.4) ilustran las necesidades de estos actores, focalizándose en sus responsabilidades y objetivos empresariales, mientras que la solución aborda sus desafíos logísticos y de sostenibilidad. Por su parte, la Figura 4.5 presenta el *canvas* para el Consumidor, centrando el análisis en su rol como punto de partida para el reciclaje. Finalmente, la Figura 4.6 muestra el *canvas* para el Centro de Reciclaje, detallando cómo la solución puede optimizar sus procesos de clasificación y venta de materiales.

El análisis detallado a través del *canvas* de propuesta de valor revela que los principales desafíos se centran en la falta de transparencia y la ineficiencia de los procesos actuales involucrados en el ciclo de vida del vidrio. El Productor Primario necesita un flujo constante de materia prima reciclada de calidad para minimizar sus costos y cumplir con metas de sostenibilidad. A su vez, el Productor Secundario enfrenta el desafío de verificar la procedencia de los envases de vidrio para garantizar la seguridad alimentaria y, de esta manera, acceder a mercados regulados y sostenibles, cumpliendo sus propias metas de sostenibilidad. Por su parte, el Consumidor busca una forma simple de reciclar, con la seguridad de que su esfuerzo es valorado

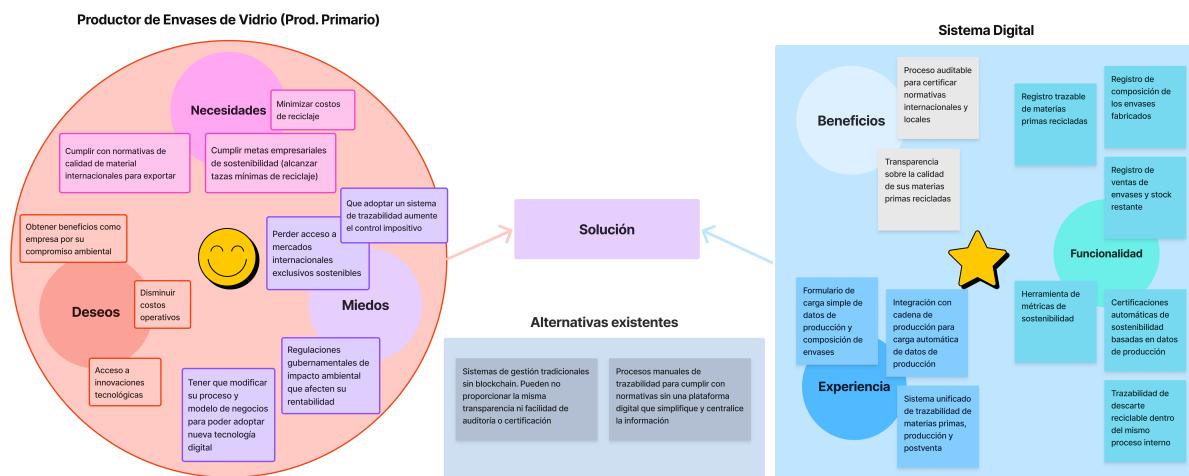


Figura 4.3: Canvas de propuesta de valor para el Productor Primario de vidrio

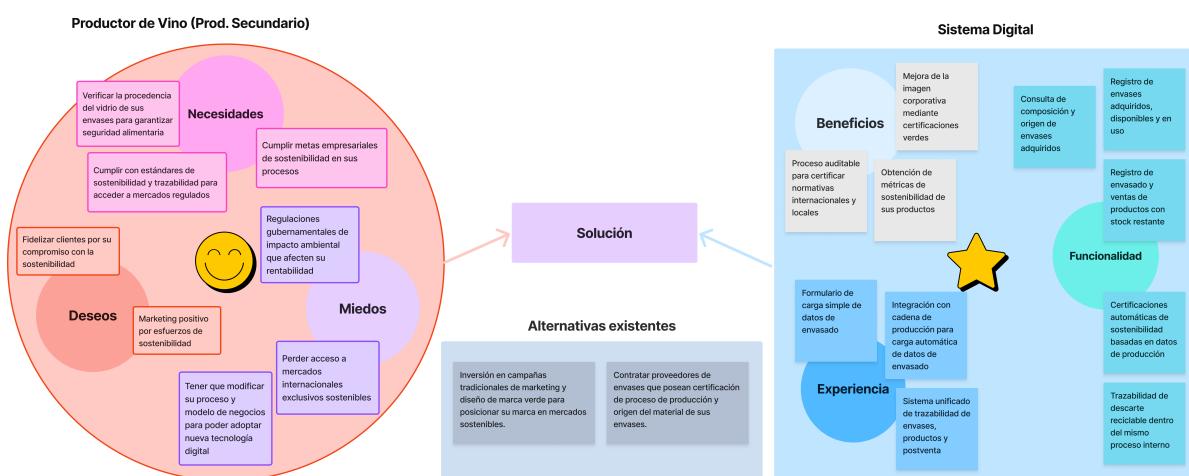


Figura 4.4: Canvas de propuesta de valor para el Productor Secundario de bebidas

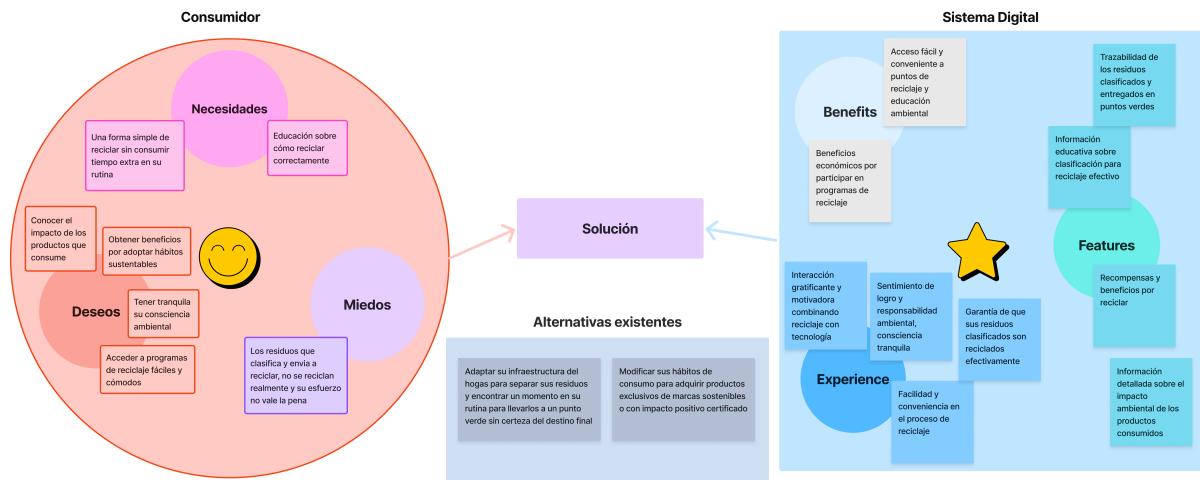


Figura 4.5: Canvas de propuesta de valor para el Consumidor

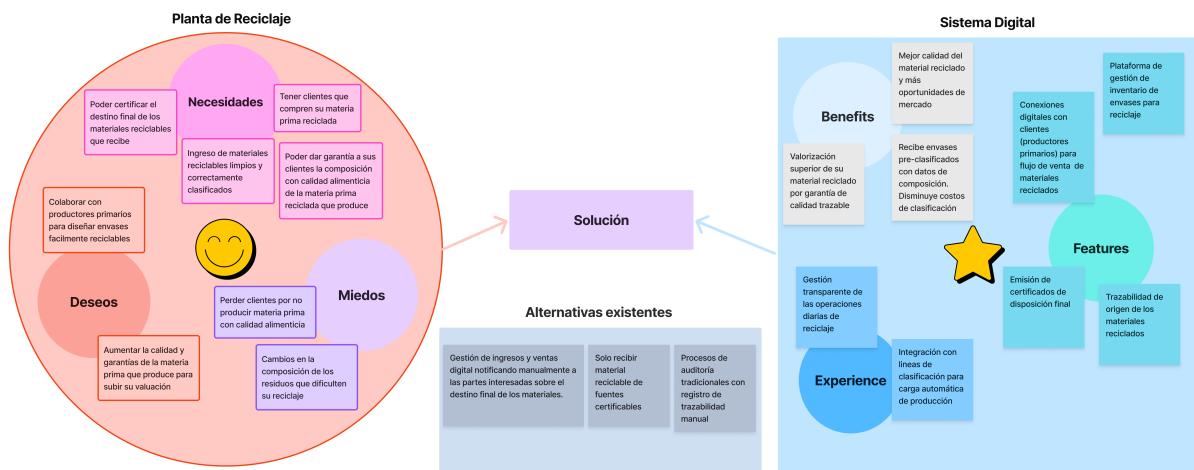


Figura 4.6: Canvas de propuesta de valor para el Centro de Reciclaje

y recompensado, y desea tener la capacidad de conocer el impacto ambiental de los productos que adquiere. Finalmente, el Centro de Reciclaje se enfrenta a altos costos operativos y a falta de calidad en el material recibido, lo que dificulta su valorización. Estas problemáticas y expectativas compartidas se traducen en la necesidad de un sistema unificado y confiable, que permita a los actores verificar la procedencia del material, documentar cada etapa del ciclo de vida y ofrecer incentivos a los usuarios finales. Por lo tanto, los casos de uso del sistema se diseñan para abordar directamente estas necesidades, permitiendo el registro de lotes de producción de envases, consulta de la trazabilidad de cada unidad, gestión del reciclaje de los envases y verificación de la calidad del material reciclado.

La información recopilada en esta fase proporciona una comprensión de las necesidades de los actores y sienta las bases para la siguiente etapa del modelado de requerimientos, donde se definirán los casos de uso del sistema con una perspectiva centrada en el usuario.

4.2. Modelado de casos de uso

Después de definir el dominio y los actores, se procede a la identificación y modelado de los *casos de uso*. Un caso de uso es una técnica de modelado que describe, de manera detallada y estructurada, las interacciones entre un actor (un usuario o sistema externo) y el sistema para lograr un objetivo de negocio específico. Cada caso de uso representa una funcionalidad completa y significativa desde la perspectiva del usuario, garantizando que el sistema final sea útil para sus usuarios.

El valor de esta herramienta de modelado reside en que permite a los equipos de desarrollo y a los interesados en el negocio comprender el “qué” (la funcionalidad), el “quién” (el actor) y el “porqué” (el objetivo) de cada interacción. Además, facilita la comunicación entre los miembros del equipo y sirve como una base sólida para la identificación de los requerimientos funcionales del sistema, asegurando que el diseño del sistema se alinee con las necesidades reales de los usuarios. Los casos de uso de un sistema se pueden documentar mediante un diagrama de casos de uso, que es una representación gráfica que muestra las relaciones entre los actores y cada caso de uso de forma conjunta, proporcionando una visión general del sistema.

Con el objetivo de acotar el alcance de este trabajo, se realiza el modelado de los casos de uso relacionados con la trazabilidad de los envases de vidrio, abarcando las acciones que los actores pueden realizar en el sistema. Estos casos de uso se centran en las funcionalidades esenciales que permiten a los actores registrar, consultar y gestionar la información relacionada con los envases de vidrio a lo largo de su ciclo de vida. Los casos de uso relacionados con la certificación de cada etapa del proceso e integración con sistemas preexistentes se consideran fuera del alcance de este trabajo, pero los casos de uso modelados permiten sentar las bases para futuras extensiones del sistema hacia estas funcionalidades.

En el prototipo de sistema de trazabilidad para envases de vidrio, los casos de uso comprenden tanto las acciones inherentes al ciclo de vida del material como las interacciones propias de una plataforma digital basada en *blockchain*. El primer grupo de casos de uso describe las operaciones fundamentales del proceso de trazabilidad, como el registro de un nuevo lote de envases fabricado por parte del Productor Primario, la consulta del origen del envase por parte del Consumidor y la recepción de envases reciclables por el Centro de Reciclaje. El segundo grupo de casos de uso, por su parte, abarca las funcionalidades básicas del sistema, tales como la autenticación de usuarios y la visualización de datos en la interfaz.

La Figura 4.7 presenta el diagrama de casos de uso elaborado para el sistema de trazabilidad de envases de vidrio para una economía circular, ilustrando las funcionalidades mínimas que debe implementar el sistema y los actores asociados a cada funcionalidad. Cada caso de uso está vinculado a un actor específico, lo que permite visualizar claramente las interacciones y responsabilidades de cada uno. En caso de que un caso de uso sea compartido por varios actores, se puede representar como un caso de uso heredado, indicando que varios actores pueden realizar la misma acción o funcionalidad.

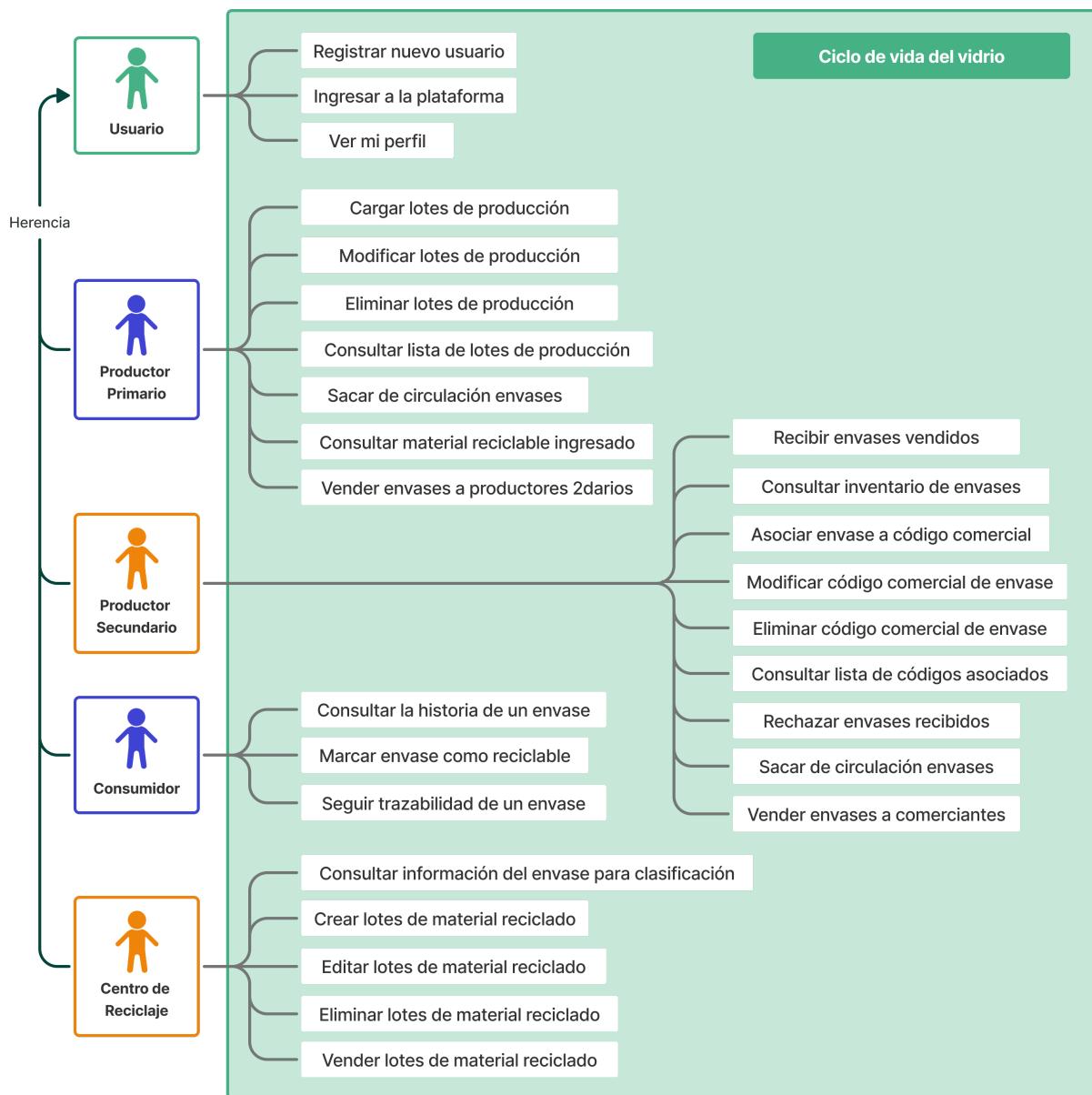


Figura 4.7: Diagrama de casos de uso del sistema de trazabilidad de vidrio

El diagrama muestra que cada actor tiene un conjunto de casos de uso alineado con sus responsabilidades. Por ejemplo, el Productor Primario puede registrar un nuevo lote de envases producidos o registrar la venta de envases a un Productor Secundario, mientras que este puede consultar la trazabilidad de origen de los envases recibidos y registrar su uso asociando los envases a un lote de producto final. Por otro lado, el Consumidor puede consultar el origen de un envase adquirido y el destino de un envase enviado a reciclaje, mientras que el Centro de Reciclaje puede recibir envases reciclables, consultar su composición de materiales y registrar su reciclaje. Los casos de uso de los diferentes actores se interrelacionan de manera cíclica, integrando la información de cada actor de la cadena en un único sistema de información que refleja el flujo de la economía circular de envases de vidrio. Por ejemplo, la acción "Vender envases a productores secundarios" del Productor Primario está ligada a los casos de uso del Productor Secundario, permitiendo rastrear el vidrio a lo largo de su ciclo de vida.

La lista de casos de uso sirve como punto de partida para la definición de los requerimientos funcionales y no funcionales del sistema, donde cada caso de uso se descompone en uno o más requerimientos específicos que describen las funcionalidades que el sistema debe implementar para cumplir con las expectativas de los actores.

4.3. Definición de requerimientos

Después de identificar los casos de uso, se definen los requerimientos funcionales y no funcionales del sistema. Los requerimientos funcionales describen las funcionalidades específicas que el sistema debe ofrecer a los usuarios, mientras que los requerimientos no funcionales establecen las características de calidad que el sistema debe cumplir, tales como rendimiento, seguridad o usabilidad.

Los requerimientos se documentan de manera estructurada, asignando un identificador único a cada requerimiento para su seguimiento durante las etapas posteriores de diseño, implementación y pruebas del sistema. La descripción de cada requerimiento incluye su propósito, las condiciones bajo las cuales se cumple y las dependencias con otros requerimientos. Un ejemplo de requerimiento funcional es que “el sistema debe permitir al Productor de Vidrio registrar un nuevo lote de vidrio, especificando la cantidad y el tipo de vidrio”, mientras que un requerimiento no funcional puede ser que “el sistema debe garantizar la seguridad de los datos del usuario mediante autenticación y autorización”.

A partir de los casos de uso previamente identificados, se definieron 28 requerimientos funcionales y 6 requerimientos no funcionales para el sistema. Estos requerimientos se documentan en la etapa de modelado para su posterior seguimiento durante el desarrollo. Se establecen dependencias entre ellos, lo que permite identificar restricciones en el orden de implementación de las funcionalidades del sistema. Por ejemplo, el registro de un lote de vidrio (RF-006) es una dependencia para que dicho lote pueda ser recibido por un productor secundario para envasar productos (RF-016). La Tabla 4.1 presenta los requerimientos funcionales junto con sus dependencias (*deps.*), mientras que la Tabla 4.2 muestra los requerimientos no funcionales definidos para el prototipo de trazabilidad de envases de vidrio basado en *blockchain*.

Tabla 4.1: Requerimientos funcionales del sistema de trazabilidad de envases de vidrio

ID	Título	Descripción	Deps.
RF-001	Registrar usuarios	<p>El sistema debe permitir registrar usuarios mediante un correo electrónico u otro medio con diferentes roles para hacer uso de las distintas partes del sistema.</p> <p>Los roles disponibles en esta primera etapa son: Productor Primario, Productor Secundario, Consumidor y Reciclador.</p>	-

Continúa en la siguiente página

ID	Título	Descripción	Deps.
RF-002	Ingresar a la plataforma	Todos los usuarios deben poder ingresar a la plataforma con su correo electrónico registrado mediante algún método de autenticación, por ejemplo, con contraseña.	RF-001
RF-003	Mantener sesión de usuario	Cada cuenta de usuarios debe poder mantener abiertas múltiples sesiones en simultáneo. El usuario debe poder cerrar una sesión individual en cualquier dispositivo. La sesión debe mantenerse abierta en el dispositivo a lo largo del tiempo a pesar de que se cierre el navegador o aplicación.	RF-002
RF-004	Validar Autorización	Cada rol de usuario debe tener ciertos permisos y un usuario con un rol dado no debe poder realizar acciones que requieran un permiso del que no goza. Detalle: - Prod. Primario: crear, editar, consultar y eliminar envases. - Prod. Secundario: editar, consultar y eliminar productos. - Consumidor: consultar composición de productos - Reciclador: consultar composición de productos, crear, editar, consultar y eliminar lotes de reciclaje.	RF-002
RF-005	Ver mi perfil de usuario	Cada usuario debe poder consultar la información personal asociada a su cuenta y modificar algunos datos: Email, Nombre Empresa/persona (modificable), Responsable (modificable), Nro teléfono (modificable), Dirección pública <i>blockchain</i> .	RF-002
RF-006	Cargar lotes de producción	El productor puede cargar lotes de producción de envases de vidrio incluyendo la siguiente información: Cantidad de envases, Peso por envase, Color, Composición, Espesor, entre otras.	RF-004
RF-007	Editar lotes de producción	El productor puede editar la información de producción en caso de error hasta antes de comercializar el lote.	RF-006
RF-008	Eliminar lotes de producción	El productor puede eliminar un lote en caso de error hasta antes de comercializar el lote.	RF-006
RF-009	Consultar historial de producción	El productor puede consultar información histórica de todos los lotes que ha creado con sus detalles y puede consultar su trazabilidad posterior.	RF-006
RF-010	Consultar material recicitable ingresado	El productor puede ver la lista de los lotes o conjuntos de materiales reciclables que volvieron a ingresar a su fábrica (como compra a la recicladora o devoluciones desde bodegas).	RF-006, RF-018, RF-025
RF-011	Sacar de circulación envases	El productor puede marcar grupos de botellas de un lote como fuera de circulación o enviado a reciclar (en caso de rotura, falla o desaparición).	RF-006
RF-012	Vender envases a productores secundarios	El productor puede marcar cierta cantidad de envases del lote como vendidos a un productor secundario específico. Los envases dejan de ser propiedad del productor y ya no puede modificarlos ni revenderlos.	RF-006

Continúa en la siguiente página

ID	Título	Descripción	Deps.
RF-013	Asociar envase a código comercial	El productor secundario puede asociar un código de barras (u otro tipo de código visible en la etiqueta impresa de su producto, como QR) a conjuntos de botellas compradas. El código debe garantizar unicidad.	RF-012
RF-014	Modificar código comercial	El productor secundario puede editar la información de códigos asociados a envases hasta antes de vender el producto.	RF-013
RF-015	Eliminar código comercial	El productor secundario puede eliminar códigos asociados a envases hasta antes de vender el producto.	RF-013
RF-016	Consultar inventario de envases nuevos	El productor secundario puede ver la lista de conjuntos de envases que ha comprado pero aún no ha utilizado (no han sido asociados a ningún producto propio ni código). El productor puede confirmar conformidad o rechazar la compra.	RF-012
RF-017	Rechazar envases recibidos	El productor secundario puede desconocer la transacción de transferencia de envases desde el productor en caso de no reconocer la compra o realizar una devolución por algún tipo de error.	RF-012
RF-018	Sacar de circulación envases	El productor secundario puede devolver botellas en caso de no conformidad, fallas de fábrica o rotura. Estos envases pueden transferirse al productor como material recicitable o descartarse.	RF-012
RF-019	Consultar historial de producción	El productor secundario puede consultar el historial de sus productos embotellados o botellas utilizadas. A su vez puede consultar su trazabilidad posterior a la comercialización.	RF-013
RF-020	Vender envases a comerciantes	El productor secundario puede marcar sus productos embotellados como vendidos al comerciante.	RF-013
RF-021	Consultar la historia de un envase	Mediante el código asociado a la botella, el ciudadano debe poder consultar el origen, composición y trazabilidad histórica de cualquier envase.	RF-013
RF-022	Marcar envase como recicitable	El ciudadano puede registrar un envase como ingresado al sistema de reciclaje escaneando su código.	RF-013
RF-023	Dar seguimiento a envases	El ciudadano puede hacer seguimiento del destino y trazabilidad hasta la disposición final de todos los envases que ingresó al sistema de reciclaje.	RF-022
RF-024	Consultar información del envase para clasificación	El reciclador clasificador puede escanear el código de la botella y obtener información relevante sobre su composición para su correcta clasificación.	RF-013
RF-025	Crear lotes de material reciclado	El reciclador puede crear lotes de material reciclado a partir de un conjunto de envases reciclables recibidos de los ciudadanos. Cada lote tiene los siguientes atributos: Peso, Dimensión (si aplica), Material, Composición.	RF-022

Continúa en la siguiente página

ID	Título	Descripción	Deps.
RF-026	Editar lote de material reciclado	El reciclador puede editar la información de un lote en caso de error hasta antes de su comercialización.	RF-025
RF-027	Eliminar lote de material reciclado	El reciclador puede eliminar un lote en caso de error hasta antes de su comercialización.	RF-025
RF-028	Vender lote de material reciclado	El reciclador puede marcar un lote de material reciclable como vendido a un productor y debe especificar el comprador. Se asume en este caso que el material fue efectivamente reciclado, finalizando la trazabilidad.	RF-025

Tabla 4.2: Requerimientos no funcionales del sistema de trazabilidad de envases de vidrio

ID	Título	Descripción
RNF-01	Transparencia	La trazabilidad de un producto debe ser libremente accesible por cualquier usuario autenticado del sistema en todo momento.
RNF-02	Disponibilidad	El sistema debe estar disponible para su uso las 24 horas del día.
RNF-03	Escalabilidad	El sistema debe soportar un número creciente de transacciones.
RNF-04	Mantenibilidad	El sistema debe poder ser mantenible por otros desarrolladores de la industria actual en el futuro
RNF-05	Interoperabilidad	El sistema debe ser integrable con múltiples sistemas de stock y gestión de terceros preexistentes.
RNF-06	Integridad	Los datos de trazabilidad no deben poder ser alterados luego de cargados sin dejar registro público de la modificación.

La lista de requerimientos funcionales y no funcionales sirve como base para la siguiente fase de modelado, en la que se definirán las historias de usuario y se planificará el desarrollo del sistema. A su vez, la lista de requerimientos se utilizará posteriormente en la validación y verificación del sistema, asegurando que todas las funcionalidades implementadas cumplan con las expectativas y necesidades de los usuarios.

4.4. Historias de usuario y planificación

A partir de la definición de los requerimientos funcionales y no funcionales del sistema, se procede a la creación de las historias de usuario. Las historias de usuario permiten documentar las funcionalidades del sistema desde la perspectiva de sus actores, utilizando un formato estandarizado que describe el rol, la acción deseada y el beneficio esperado: “Como [rol], quiero [acción], para [beneficio]”. Por ejemplo, “Como Productor Primario, quiero poder editar la información de un lote de envases antes de su comercialización, para poder corregir cualquier error en los datos de producción y asegurar la precisión en el registro”. Esta forma de do-

cumentar requerimientos facilita la priorización de funcionalidades y la comprensión de las necesidades desde un enfoque centrado en el usuario a la hora de desarrollar el sistema.

Cada historia de usuario se complementa con criterios de aceptación que establecen las condiciones necesarias para su validación, vinculándose directamente con uno o más requerimientos previamente definidos. Esta trazabilidad entre los requerimientos y las historias de usuario es un mecanismo de control que guía el proceso de desarrollo y asegura que la implementación cumpla con las expectativas planteadas. En el contexto del modelo en V, las historias de usuario establecen la base para la fase de pruebas de aceptación, garantizando que el sistema final se alinee con los objetivos del proyecto.

A continuación se presenta un ejemplo de historia de usuario con sus respectivos criterios de aceptación, que ilustra la relación entre las funcionalidades y las necesidades de los actores.

Historia de Usuario: Consultar historial de producción de lotes de envases de vidrio

Como productor,

Quiero poder consultar el historial de todos los lotes de producción con sus detalles y trazabilidad,

Para poder revisar la información de producción y rastrear cada lote en su ciclo de vida.

Criterios de Aceptación:

1. Visualización del historial de lotes:

- El sistema debe mostrar una lista de todos los lotes creados por el productor con la siguiente información:
 - **Código de lote**
 - **Fecha de producción**
 - Cantidad de envases
 - Peso por envase
 - Color

2. Acceso a detalles de cada lote:

- Al seleccionar un lote específico, el sistema debe mostrar los detalles completos, incluyendo:
 - Espesor
 - **Fecha de producción**
 - Observaciones adicionales

En el presente trabajo, se definieron un total de 28 historias de usuario, donde cada historia de usuario se corresponde exactamente con un requerimiento funcional del sistema. Los requerimientos no funcionales se abordan de manera transversal, asegurando que aspectos como el rendimiento, la seguridad y la usabilidad sean considerados en el diseño del sistema.

Todas las actividades

Básica JQL Buscar... Proyecto = Tesis LCC Persona asignada Tipo Estado Más filtros

Borrar filtro Guardar filtro

<input type="checkbox"/>	Actividad	Persona asignada	Informador	Prioridad	Estado	Res.	+
<input type="checkbox"/>	SCRUM-14 RF-009 Consultar historial de prod...	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-13 RF-008 Eliminar lotes de producción	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-12 RF-007 Editar lotes de producción	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-11 RF-006 Cargar lotes de producción...	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-10 RF-005 Ver y editar perfil de usuario	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-9 RF-004 Validar autorización según ...	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-8 RF-003 Mantener sesión de usuario	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-7 RF-003 Mantener sesión de usuario	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	
<input type="checkbox"/>	SCRUM-6 RF-001 Registrar usuario con difere...	Rocío Corral	Rocío Corral	= Medium	FINALIZADA	Listo	

Figura 4.8: Tablero de Jira para la gestión de historias de usuario

Cada historia de usuario se registró en la herramienta de gestión Jira², con el objetivo de facilitar la gestión del proceso de desarrollo siguiendo la metodología Kanban. Como parte del proceso de planificación, se estimó el esfuerzo necesario para implementar la funcionalidad de cada historia de usuario y se registró en la herramienta Jira junto con la tarea correspondiente. La estimación del esfuerzo consideró la complejidad técnica, el tiempo requerido proyectado para implementarlo y las interdependencias entre las funcionalidades, lo que permitió una planificación objetiva del desarrollo.

La planificación del proyecto se realizó mediante un diagrama de Gantt³. Un diagrama de Gantt es una herramienta visual que muestra la secuencia de las tareas, sus dependencias y los plazos de implementación estimados. Este diagrama permite visualizar el cronograma del proyecto, facilitando la identificación de hitos y la gestión de recursos. En este caso, se utilizó para planificar las historias de usuario y su implementación en iteraciones o sprints, asegurando que todas las funcionalidades necesarias sean contempladas y ejecutadas de manera ordenada según sus dependencias.

En la Figura 4.8, se muestra una captura del tablero utilizado para el seguimiento del progreso de cada historia de usuario en la plataforma Jira, mientras que la Figura 4.9 presenta el diagrama de Gantt creado para este proyecto. Con la planificación armada, se estimó que el desarrollo del sistema tendría una duración de 8 semanas, considerando una dedicación de 20 horas semanales, para implementar las 28 historias de usuario definidas. Esta estimación de tiempo corresponde exclusivamente al tiempo de codificación de funcionalidades del prototi-

² <https://www.atlassian.com/es/software/jira>

³ <https://www.atlassian.com/es/agile/project-management/gantt-chart>

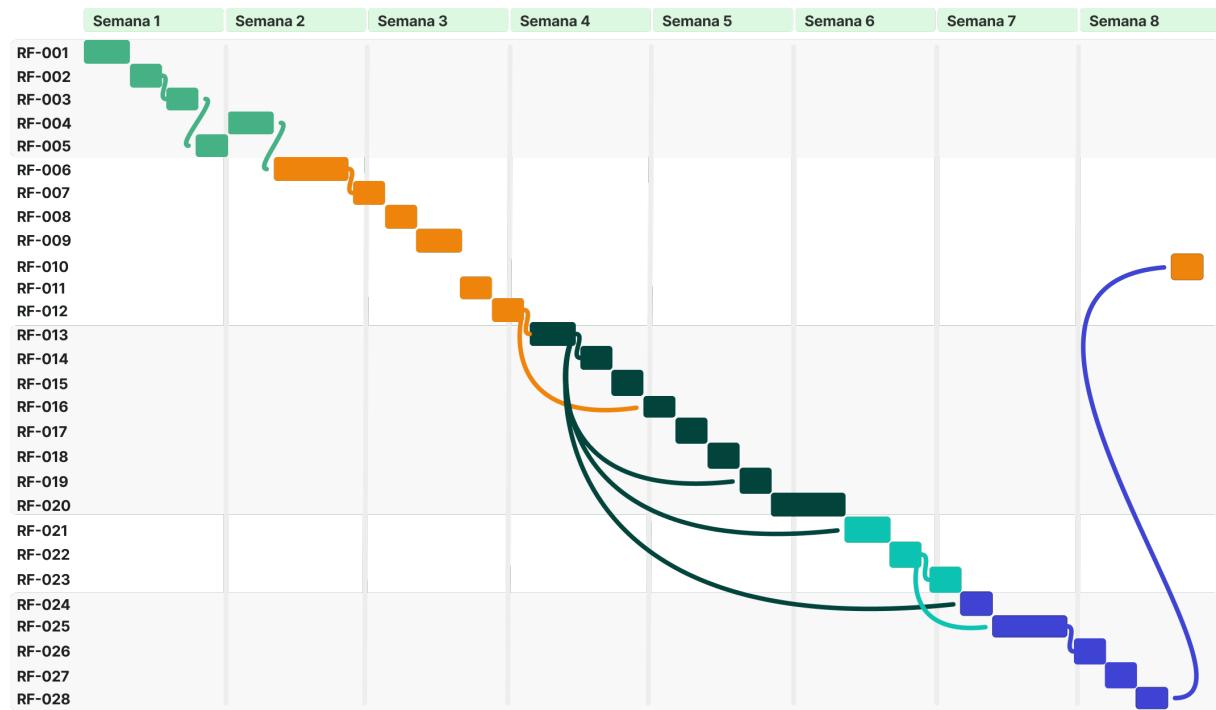


Figura 4.9: Diagrama de Gantt para la planificación del proyecto

po, ya que, posterior a la etapa de generación de código, se procede a la fase de codificación y ejecución de pruebas automatizadas y validación manual del sistema, donde es posible a su vez que se deban realizar ajustes o correcciones de programación en función de los resultados obtenidos.

Concluidas las fases de modelado de requerimientos y planificación, se establece la base para la siguiente etapa del proceso. El conjunto de requerimientos, casos de uso y la planificación detallada con las historias de usuario constituyen la referencia que guiará las fases de diseño, implementación y pruebas del sistema. En el próximo capítulo, se abordará el diseño de la arquitectura y los componentes del sistema, donde se definirán las soluciones tecnológicas y la estructura del software que implementará los requerimientos establecidos.

5

DISEÑO DE SOLUCIÓN

En la fase de modelado de requerimientos se definieron las bases del sistema, estableciendo sus funcionalidades y características que el prototipo debe cumplir. A partir de estas bases, es posible proceder a la etapa de diseño de solución, que es una transición entre la especificación de lo que el sistema debe hacer y el cómo se construirá, transformando los requerimientos abstractos en una arquitectura concreta y un plan de implementación. El diseño permite obtener una visión integral del sistema antes de iniciar la implementación, lo que facilita la identificación de dependencias e interfaces y asegura que todos los componentes y módulos interactúen de manera coherente. Este proceso de planificación anticipada reduce la probabilidad de encontrar inconsistencias o funcionalidades indefinidas durante las fases de desarrollo y pruebas.

El diseño de la solución, en el marco del modelo en V, se aborda en dos etapas: primero se realiza el diseño de arquitectura y luego el diseño de componentes. Cada una de estas etapas se enfoca en un nivel de abstracción distinto del sistema. El diseño de arquitectura comprende la definición de la estructura general del sistema a través de módulos, mientras que el diseño de componentes se ocupa de los detalles internos de cada módulo.

En la primera etapa, diseño de arquitectura, el sistema se divide en subsistemas o módulos lógicos y se definen sus interacciones, las responsabilidades de cada uno y las tecnologías subyacentes. Este enfoque permite establecer las bases del sistema, abarcando tanto los requerimientos funcionales como los no funcionales. Las decisiones de diseño tomadas en esta etapa se validan posteriormente mediante pruebas de sistema, las cuales se encargan de verificar que todos los módulos trabajen conjuntamente y que el sistema de forma integral satisfaga los requerimientos especificados.

Por otro lado, la segunda etapa es el diseño de componentes, donde se profundiza en los detalles de la arquitectura interna de cada módulo. Esto incluye la especificación de clases, interfa-

ces, flujos de datos y la organización de la lógica de negocio. Los requerimientos funcionales, guiados por las historias de usuario, se traducen en documentos de arquitectura de software específicos que posteriormente se implementan en la fase de codificación. Las decisiones de diseño tomadas en esta etapa se verifican a través de las pruebas de integración, que aseguran que los componentes individuales interactúen de forma correcta entre sí.

Los requerimientos funcionales, definidos en el capítulo anterior, son el fundamento para el diseño de la solución y se utilizan en esta etapa a través de las historias de usuario, que guían el diseño de los módulos y componentes. De manera complementaria, los requerimientos no funcionales también se contemplan en la fase de diseño, particularmente en el diseño de arquitectura, donde se establecen las bases para garantizar atributos como la seguridad, el rendimiento y la escalabilidad, incluso si no están directamente asociados a una historia de usuario.

A continuación, se explicará cada una de las etapas del diseño de solución, comenzando con el diseño de arquitectura en la Sección 5.1, para luego avanzar con el diseño de componentes en la Sección 5.2.

5.1. Diseño de arquitectura

La fase de diseño de arquitectura representa el primer paso en la traducción de los requerimientos del sistema hacia un sistema de software funcional. El diseño permite una visión global de la solución, asegurando que todos los componentes y módulos trabajen juntos de manera coherente previo a la implementación. A partir de los requerimientos previamente definidos, se establece un marco de trabajo de alto nivel que estructura la solución en componentes lógicos, delineando sus responsabilidades, las interacciones entre ellos y las tecnologías que los soportan. La salida de esta etapa es la arquitectura del sistema, la cual servirá de base para las decisiones de diseño a un nivel más granular. En el contexto del modelo en V, las decisiones tomadas en esta etapa se validan en la fase de pruebas de sistema, donde se verifica que la arquitectura en su conjunto cumple con las especificaciones definidas en los requerimientos funcionales y no funcionales del sistema.

Para el prototipo tecnológico de trazabilidad de vidrio basado en *blockchain*, la arquitectura del sistema se concibió siguiendo un enfoque de tres capas lógicas para asegurar una clara separación de responsabilidades y modularidad. Este patrón, común en el desarrollo de aplicaciones web, permite que cada capa se desarrolle, mantenga y escale de forma independiente. En la Figura 5.1 se ilustra el diseño de la arquitectura de módulos del sistema elaborado siguiendo este enfoque. La arquitectura se compone de la capa de presentación (*frontend*), la capa de lógica de negocio (*backend*) y la capa de datos (*blockchain* y base de datos relacional). La comunicación entre estas capas se define a través de interfaces estandarizadas, lo que promueve una baja dependencia y puede facilitar futuras extensiones e integraciones, por ejemplo, con sistemas de gestión externos o con dispositivos IoT para automatizar la captura de datos en los procesos productivos.

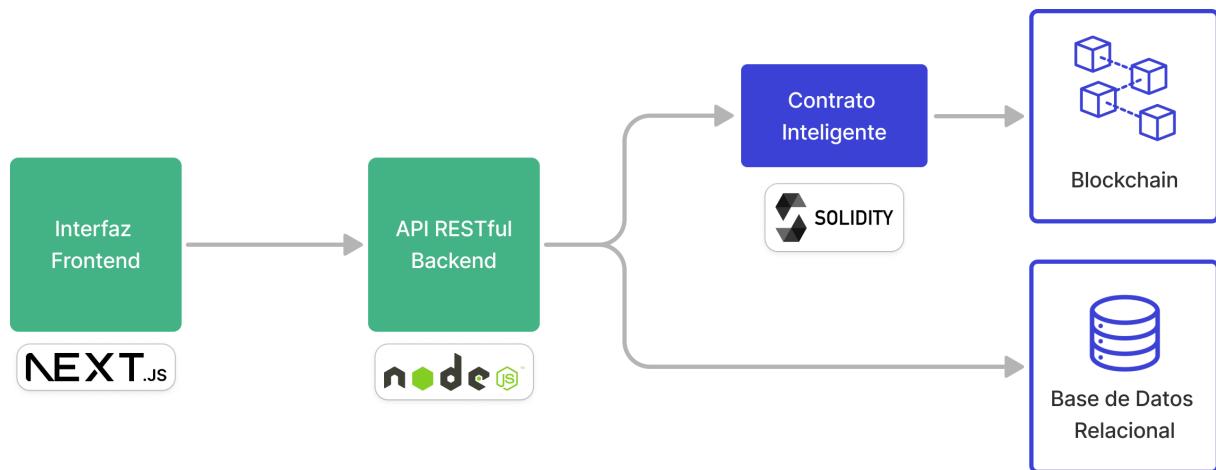


Figura 5.1: Arquitectura de módulos del sistema de trazabilidad de envases de vidrio basado en blockchain

A su vez, dentro de cada capa del sistema, es necesario definir un criterio para la delimitación de los módulos lógicos dentro de la misma capa. El criterio aplicado en este trabajo se basa en el principio de *cohesión funcional*, el cual plantea agrupar las funcionalidades y responsabilidades del sistema a partir de cada rol de usuario identificado. Siguiendo este criterio, para este trabajo se decidió implementar un módulo específico para cada actor (productor primario, productor secundario, consumidor y centro de reciclaje), así como módulos compartidos para la lógica de negocio transversal, como la autenticación de usuarios o trazabilidad de envases de vidrio. Esta división de responsabilidades reduce el acoplamiento entre los módulos y simplifica el mantenimiento del sistema, en el caso de requerir cambios o actualizaciones en el futuro.

Luego de definir las capas que componen la arquitectura del sistema, es posible proceder a seleccionar las tecnologías más adecuadas para cada uno de los módulos definidos. La elección de tecnologías para cada capa busca satisfacer una serie de criterios técnicos y de negocio, incluyendo la compatibilidad con otras tecnologías, el rendimiento, la escalabilidad y el soporte de la comunidad. El diseño de la arquitectura tecnológica del sistema, incluyendo la definición de tecnologías, se realizó siguiendo un enfoque “de adentro hacia afuera”, comenzando por la representación de los datos para luego diseñar su acceso (*backend*) y finalmente la forma en la cual los datos se muestran (*frontend*). Esta perspectiva se elige para priorizar la integridad y la persistencia de la información, que son pilares fundamentales de un sistema basado en *blockchain*. Al abordar primero la capa de datos y la lógica de negocio (es decir, los contratos inteligentes), se puede garantizar que la estructura central del sistema sea robusta para disminuir el riesgo de inconsistencias en una etapa avanzada de la implementación. Este enfoque, en lugar de uno “de afuera hacia adentro” (del *frontend* a los datos), asegura que la lógica de negocio sea el foco principal del diseño, permitiendo que la interfaz de usuario y otras capas de presentación se adapten a la funcionalidad central, y no al revés.

A continuación, se describen las tecnologías seleccionadas, los patrones de diseño aplicados y las decisiones arquitectónicas tomadas para cada capa del sistema. Comenzando por la capa de datos, se explicará la arquitectura desde adentro hacia afuera, siguiendo el flujo de datos desde su representación y almacenamiento hasta la presentación al usuario final.

5.1.1. Capa de datos

La capa de datos constituye el cimiento de todo sistema de software. Se encarga de la persistencia, gestión y recuperación de toda la información del sistema. Las bases de datos relacionales son el tipo de base de datos más utilizado actualmente para gestionar información estructurada. Los datos se organizan en tablas, con filas y columnas, y se pueden establecer relaciones entre las distintas tablas mediante claves únicas. Cada columna tiene un nombre y un tipo de dato asociado, mientras que cada fila representa un registro único dentro de la tabla y su contenido debe cumplir con el tipo de dato definido para cada columna. Por ejemplo, se podría definir una tabla “Usuario”, para almacenar información sobre los usuarios del sistema, incluyendo columnas como “nombre” de tipo texto, “correo electrónico” de tipo texto y “fecha de registro” de tipo fecha, entre otras columnas. Cada fila en esta tabla representaría un usuario específico, con su nombre, correo electrónico y la fecha en que se registró en el sistema, entre otros datos. Las principales ventajas de las bases de datos relacionales radican en su capacidad para garantizar la uniformidad de los datos a través de esquemas definidos para cada tabla, así como la facilidad para realizar consultas y análisis complejos de manera eficiente. Sin embargo, su naturaleza centralizada las hace susceptibles a manipulaciones manuales difíciles de detectar, lo que justifica la necesidad de una tecnología complementaria, como la *blockchain*, para registrar información que requiere garantías de integridad.

En este prototipo, la tecnología *blockchain* resulta apropiada para registrar la información clave del sistema, que requiere inmutabilidad y transparencia para fomentar la confianza entre los distintos actores. Ejemplos de este tipo de información pueden ser la propiedad de los lotes de envases de vidrio o la trazabilidad de los envases reciclados. Por este motivo, se tomó la decisión estratégica de combinar la *blockchain* con una base de datos relacional con el objetivo de resolver la necesidad de equilibrio entre la seguridad y la transparencia, con el rendimiento y la escalabilidad. La *blockchain*, por su naturaleza, provee una fuente de información segura y transparente, pero puede presentar limitaciones en cuanto a la velocidad de las transacciones y el volumen de datos que permite consultar eficientemente. Para abordar estas limitaciones, es que se decidió utilizar la *blockchain* como base de datos principal del sistema, mientras que se decidió utilizar una base de datos relacional complementaria para almacenar datos auxiliares e indexar los datos de acceso frecuente, como los perfiles de usuario, los metadatos de los lotes (*id*, propietario, estado, etc.) y otros datos de soporte que no requieren inmutabilidad. La relación entre ambas fuentes de datos se maneja mediante identificadores únicos que se almacenan en la *blockchain*, sirviendo como una referencia a los datos detallados en la base de datos relacional. La capa *backend* es responsable de mantener la consistencia entre ambas fuentes de información (*blockchain* y base de datos relacional) a la hora de persistir nueva información o actualizar información existente. Este enfoque híbrido permite proveer una recuperación de información ágil y una experiencia de usuario fluida sin poner en compromiso la integridad de los datos sensibles.

Luego de definir la estrategia de almacenamiento y gestión de datos, es posible proceder a seleccionar las tecnologías específicas que se utilizarán, tanto para la base de datos relacional

como para la plataforma *blockchain*, ya que existen múltiples opciones en el mercado.

En el caso de la base de datos relacional, se optó por el uso de MariaDB¹, una base de datos de código abierto elegida por su sencillez y familiaridad, ya que el uso que se le dará en este trabajo es complementario y no hace falta utilizar una alternativa más compleja. MariaDB cuenta con un amplio soporte, librerías y documentación disponible para conectarse a ella de forma estandarizada desde cualquier lenguaje o *framework* utilizado en la capa *backend*.

Por otro lado, para la tecnología *blockchain*, la elección de una plataforma adecuada puede determinar la complejidad y tiempo de implementación del sistema. La tecnología *blockchain* no se implementa desde cero, sino que se utilizan plataformas ya desarrolladas y probadas que ofrecen características y funcionalidades específicas. Estas plataformas, conocidas como protocolos *blockchain*, varían en aspectos como su mecanismo de consenso, lenguaje de programación y comunidad de desarrolladores. Para este trabajo, se analizaron cinco de las plataformas líderes en la industria para su análisis y comparación: Hyperledger Fabric, Ethereum, Polkadot, VeChain y Cardano. Estas plataformas se seleccionaron por su relevancia y características técnicas, evaluando su idoneidad para el caso de uso específico de trazabilidad en la cadena de suministro del vidrio.

Hyperledger Fabric² es una plataforma de código abierto diseñada para uso empresarial, que forma parte de la Fundación Linux [3]. Se caracteriza por su arquitectura modular y configurable, ideal para una amplia gama de casos de uso en la industria. A diferencia de las redes públicas, es una plataforma permisionada, lo que significa que los participantes se conocen y confían entre sí. Admite contratos inteligentes en lenguajes de programación comunes como Java³ y Node.js⁴, lo que reduce la curva de aprendizaje. Hyperledger no requiere una criptomoneda nativa, lo que elimina ciertos riesgos de ataque y permite que la plataforma se implemente con costos operativos similares a los de cualquier otro sistema distribuido.

Ethereum⁵ es una plataforma de código abierto y pública que permite a los desarrolladores crear contratos inteligentes y aplicaciones descentralizadas (dApps). Se considera una computadora mundial descentralizada, alimentada por su criptomoneda nativa, Ether. Ethereum fue pionera en la creación de contratos inteligentes y ha mantenido un liderazgo en la industria desde su lanzamiento en 2015. Los contratos se escriben en Solidity⁶, un lenguaje de programación de dominio específico que se ejecuta en la red de Ethereum. Este lenguaje de programación en la actualidad es el más utilizado para el desarrollo de contratos inteligentes. Aunque Ethereum se lanzó con un protocolo de consenso de Prueba de Trabajo (PoW), la plataforma ha migrado a la Prueba de Participación (PoS) para mejorar su eficiencia energética y escalabilidad.

Polkadot⁷ es una plataforma de código abierto que busca facilitar la interoperabilidad entre

¹ <https://mariadb.org/documentation/>

² <https://hlf.readthedocs.io/en/latest/>

³ <https://www.java.com/en/>

⁴ <https://nodejs.org/en>

⁵ <https://ethereum.org/en/developers/docs/>

⁶ <https://www.soliditylang.org/>

⁷ <https://docs.polkadot.com/>

diferentes *blockchains*. Su objetivo es crear una red escalable y segura que pueda soportar una amplia gama de aplicaciones descentralizadas. Su arquitectura se basa en una cadena principal (*relay chain*) y múltiples cadenas que se conectan a ella (*parachains*), permitiendo que las *blockchains* se comuniquen entre sí de manera eficiente a través de la *relay chain*. Utiliza un protocolo de consenso derivado de PoS, llamado Prueba de Participación Nominada (NPoS, *Nominated Proof of Stake*) y su criptomoneda nativa es DOT. Las aplicaciones se desarrollan con Substrate⁸, un *framework* modular escrito en Rust⁹, que también es compatible con contratos inteligentes escritos en Solidity.

VeChain¹⁰ es una plataforma de código abierto dedicada específicamente a la trazabilidad y la autenticación de productos en la cadena de suministro. Combina tecnología *blockchain* con RFID e IoT para rastrear productos desde la producción hasta el consumidor final. Es una plataforma permisionada, donde los participantes se conocen y confían mutuamente, lo que permite una mayor privacidad y confidencialidad. Utiliza una arquitectura de dos *tokens* (VET y VTHO) y un protocolo de consenso de Prueba de Autoridad (PoA). Al ser compatible con Solidity, facilita la migración de aplicaciones existentes de Ethereum.

Cardano¹¹ es una plataforma de *blockchain* de código abierto que se enfoca en la creación de una red escalable, segura y sostenible. Se distingue por su enfoque científico y riguroso, utilizando evidencia formal y revisión por pares para garantizar la seguridad y confiabilidad de la plataforma. Para programar aplicaciones se utiliza el lenguaje de programación funcional Haskell, que permite la verificación formal de los contratos inteligentes. También se pueden desarrollar contratos utilizando Plutus¹², un lenguaje de dominio específico basado en Haskell¹³. La red utiliza un protocolo de consenso PoS y su criptomoneda nativa es ADA.

En la Tabla 5.1, se presenta un resumen comparativo de los protocolos *blockchain* analizados, destacando los aspectos relevantes de cada uno para la selección del más adecuado para este trabajo.

Tabla 5.1: Comparación de plataformas *blockchain*

Tecnología	Hyperledger	Ethereum	Polkadot	VeChain	Cardano
Consenso	Flexible	PoW - PoS	NPoS	PoA	PoS
Lenguaje	Java, Go, Node.js	Solidity	Rust, Solidity	Solidity	Haskell
Interoperabilidad	Limitada	Limitada	Alta	Limitada	Limitada
Adopción	Alta	Muy alta	Media	Media	Media
Comunidad	Grande	Grande	Grande	Mediana	Grande

⁸ <https://polkadot.com/platform/sdk/>

⁹ <https://rust-lang.org/>

¹⁰ <https://docs.vechain.org/>

¹¹ <https://docs.cardano.org/>

¹² <https://docs.cardano.org/developer-resources/smart-contracts/plutus>

¹³ <https://www.haskell.org/>

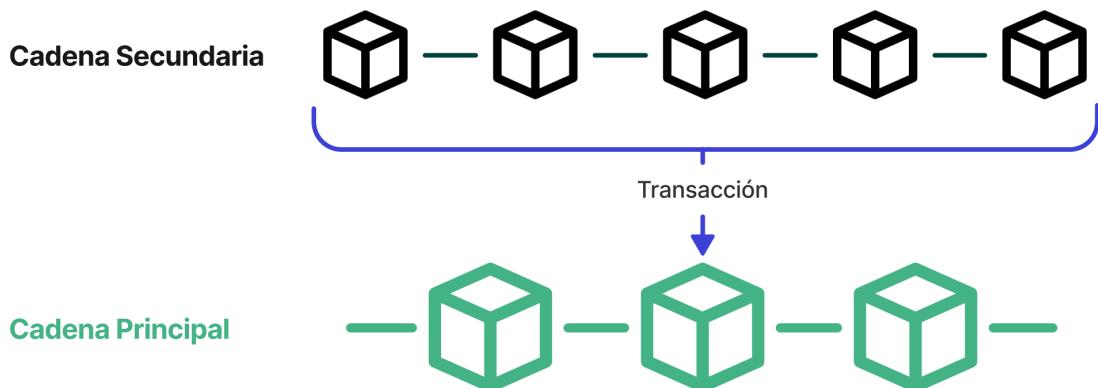


Figura 5.2: Funcionamiento de una cadena secundaria sobre Ethereum

Luego de realizar el análisis comparativo entre protocolos *blockchain*, se llega a la determinación de que Ethereum resulta ser la tecnología más adecuada para este trabajo por múltiples razones. En primer lugar, su naturaleza pública está alineada con el objetivo del proyecto, ya que permite a cualquier persona unirse a la red y verificar el estado de la cadena de forma transparente, sin necesidad de permisos, como puede ocurrir en las plataformas permisionadas como Hyperledger. A su vez, otro factor influyente en esta elección es la adopción del mecanismo de consenso PoS en la red Ethereum, que es más eficiente energéticamente que PoW, de modo que se alinea directamente con los objetivos de sostenibilidad ambiental del proyecto. Además, Ethereum posee la mayor comunidad de desarrolladores entre las plataformas analizadas y una alta adopción en la industria, lo que garantiza un soporte continuo y una amplia gama de herramientas y recursos a su disposición. Su lenguaje de programación, Solidity, es de alto nivel y fácil de aprender en comparación a lenguajes como Rust y Haskell, permitiendo la creación de aplicaciones complejas de manera eficiente. Finalmente, existe una amplia variedad de herramientas que permiten conectar otras tecnologías y sistemas con Ethereum, facilitando la integración con la base de datos relacional y la capa *backend* del sistema.

Sin embargo, Ethereum presenta algunas desventajas, como los altos costos de transacción y alta latencia de red, que pueden afectar la experiencia del usuario y la viabilidad económica del sistema. Afortunadamente, en la actualidad existen múltiples soluciones para mitigar estos problemas. En particular, para mitigar los altos costos y la latencia de la red de Ethereum para esta aplicación de trazabilidad que puede alcanzar un volumen de datos considerable, se decidió realizar el despliegue de los contratos en una cadena secundaria de Ethereum, que permite procesar las transacciones en una cadena complementaria de alto rendimiento y bajo costo para luego sincronizarla con la cadena principal mediante una única transacción (Figura 5.2), lo que reduce costos y aumenta la escalabilidad, sin comprometer la seguridad ni la integridad de los datos de la *blockchain* Ethereum. A su vez, se eligió hacer uso del *framework* Hardhat¹⁴ para el desarrollo y despliegue de los contratos inteligentes, ya que es una herramienta ampliamente adoptada que facilita la implementación y ejecución de pruebas unitarias y de integración sobre contratos inteligentes en Solidity.

¹⁴<https://hardhat.org>

5.1.2. Capa backend

La capa de lógica de negocio, generalmente conocida como *backend* o API (*Application Programming Interface*), actúa como el cerebro del sistema. Su propósito principal es implementar y ejecutar las reglas de negocio del sistema, orquestar la interacción entre las distintas capas (capa *frontend* y capa de datos), y exponer una interfaz estandarizada a través de la cual otros componentes puedan interactuar con la funcionalidad del sistema, sin depender de su implementación interna.

Para este prototipo, se ha optado por implementar una arquitectura desacoplada, donde la capa de presentación (*frontend*) y la capa de lógica de negocio se desarrollan de forma independiente. Este tipo de arquitectura desacoplada resulta necesaria para un sistema de las características de este trabajo. A diferencia de una arquitectura monolítica, este enfoque promueve la modularidad y la escalabilidad, permitiendo que la interfaz de usuario pueda evolucionar o ser reemplazada sin afectar la lógica de negocio central de la aplicación. Este diseño responde directamente al requerimiento no funcional de interoperabilidad (RNF-05), ya que la API de trazabilidad está pensada para ser el punto de integración principal no solo para el *frontend* del prototipo, sino también para sistemas de gestión preexistentes, dispositivos IoT y otras aplicaciones de terceros que pudieran surgir en el futuro para la trazabilidad de procesos de la cadena de suministro y reciclaje de los envases de vidrio.

Para la implementación de la capa *backend* se tomó la decisión de utilizar la tecnología Node.js¹⁵, un entorno de ejecución del lenguaje JavaScript¹⁶ que permite crear servidores, aplicaciones web, herramientas de línea de comandos y *scripts*. A su vez, JavaScript es un lenguaje de programación de alto nivel, interpretado y dinámico, que es ampliamente conocido y utilizado para el desarrollo web tanto en *frontend* como en *backend*. Su curva de aprendizaje es relativamente baja y es el lenguaje que se utiliza para dar dinamismo a las páginas web. Sin embargo, su uso también se ha extendido al lado del servidor gracias a entornos de ejecución como Node.js, lo que ha permitido a los desarrolladores crear aplicaciones web completas utilizando un único lenguaje de programación. Su popularidad y amplia adopción en la industria, se deben a su flexibilidad y a la gran cantidad de librerías y *frameworks* que se han desarrollado para este lenguaje, que permiten construir tanto aplicaciones web simples, como sistemas complejos y escalables.

Existen múltiples motivos que justifican la elección del entorno Node.js para el desarrollo de la capa *backend* de este trabajo. En primer lugar, considerando el alcance limitado del trabajo, JavaScript resulta ser un lenguaje propicio debido a su baja curva de aprendizaje y su flexibilidad para ser utilizado tanto en la capa *backend* como en la capa *frontend*, lo que facilita el proceso de desarrollo al permitir que el mismo desarrollador trabaje en ambas capas sin necesidad de cambiar de lenguaje. Existen múltiples lenguajes y *frameworks* ampliamente utilizados en la industria para desarrollar la capa *backend* de aplicaciones web, como Java con Spring Boot,

¹⁵<https://nodejs.org/es>

¹⁶<https://developer.mozilla.org/es/docs/Web/JavaScript>

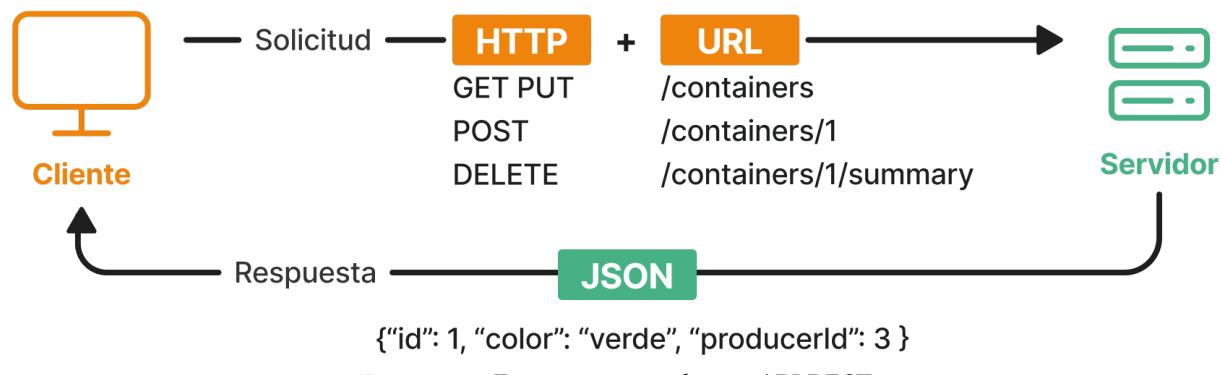


Figura 5.3: Funcionamiento de una API REST

Python con Django o Flask y Ruby on Rails, pero la posibilidad de utilizar el mismo lenguaje tanto en *frontend* como en *backend*, simplifica el proceso de desarrollo en equipos pequeños o unipersonales y ayuda a facilitar el mantenimiento del código en el futuro. A su vez, Node.js ofrece un excelente soporte para la interacción con la red de Ethereum, mediante una serie de librerías ampliamente adoptadas y probadas en la comunidad, como pueden ser Ethers.js y Web3.js. Por último, la amplia adopción de Node.js en la industria y la disponibilidad de herramientas para pruebas unitarias facilitan el desarrollo y el mantenimiento del sistema.

Por otro lado, la comunicación del *backend* con las demás capas del sistema debe implementarse a través de interfaces bien definidas. La API *backend* implementa el estándar *API REST* para la comunicación con la capa de presentación. El estándar REST (*Representational State Transfer*), ilustrado en la Figura 5.3, define una arquitectura para la comunicación entre sistemas que se basa en el uso del protocolo HTTP para intercambiar información sin mantener un estado entre intercambios. Una API REST utiliza los métodos estándar de HTTP (como *GET*, *POST*, *PUT* y *DELETE*) para realizar operaciones sobre los recursos del sistema y adopta el formato JSON para el intercambio de datos. La API se estructura en un conjunto de *endpoints*, cada uno representando una acción sobre un recurso específico del sistema, ya sea crear, leer, actualizar o eliminar el recurso. Por ejemplo, el *frontend* del sistema podría enviar una solicitud *GET* al *endpoint* “/api/containers” para solicitar la lista de envases de vidrio producidos por cierto productor primario. En este caso, el *backend* respondería con la información solicitada en formato JSON y el *frontend* podría utilizar esta información para actualizar la interfaz de usuario, mostrando el listado de los envases de vidrio correspondientes al usuario.

Dentro del entorno de Node.js, existen múltiples librerías y *frameworks* que facilitan el desarrollo de APIs REST. La opción más popular y ampliamente adoptada es Express.js¹⁷, un *framework* minimalista y flexible que proporciona las utilidades esenciales para la construcción de APIs REST. Entre sus principales ventajas se encuentran su simplicidad, flexibilidad y extensibilidad. Gracias a estas características, Express.js es utilizado para construir API REST directamente, pero también ha sido utilizado como base de múltiples *frameworks* de desarrollo de APIs REST más complejos, como Nest.js y Sails.js, entre otros. Estos *frameworks* extienden las funcionalidades de Express.js y facilitan el desarrollo de APIs en Node.js, añadiendo fun-

¹⁷<https://expressjs.com/>

cionalidades adicionales que mejoran la mantenibilidad y escalabilidad de las APIs REST. Sin embargo, estos *frameworks* suelen imponer una estructura más rígida y una curva de aprendizaje más pronunciada que Express.js. Por este motivo, se decidió utilizar Express.js como base para el desarrollo de la API del sistema, ya que no impone restricciones adicionales sobre el modelo arquitectónico y permite una implementación eficiente y escalable, sin sacrificar la flexibilidad necesaria para adaptarse a los requerimientos específicos del sistema.

Finalmente, para la interacción con la capa de datos, el *backend* se conecta a la base de datos relacional mediante librerías de conexión estandarizadas y a la *blockchain* a través de librerías de interacción con contratos inteligentes, lo que permite un acceso unificado a los datos, abstractando la complejidad de cada fuente de datos y proporcionando una interfaz coherente para la capa *frontend* del sistema.

5.1.3. Capa frontend

La capa de presentación, conocida habitualmente como *frontend*, es la interfaz de usuario web que permite a los distintos actores de la cadena de suministro interactuar con el sistema de trazabilidad del vidrio. Su objetivo es traducir la lógica de negocio y los datos que provienen de la API en una experiencia visual y funcional, permitiendo que los usuarios interactúen con el prototipo de manera intuitiva. El *frontend* es la cara visible del sistema completo.

Para este prototipo, se tomó la decisión de implementar de forma desacoplada la interfaz de la lógica de negocio, con el objetivo de reforzar la mantenibilidad y flexibilidad del sistema. Esta separación es propicia para que la interfaz de usuario pueda evolucionar de manera independiente de la lógica, adaptándose a nuevas necesidades, experiencias de usuario o tecnologías sin afectar el funcionamiento del sistema. Con este esquema, es factible que en un futuro, cada actor de la cadena tenga acceso a una interfaz a medida de sus necesidades. Por ejemplo, para un productor de envases de vidrio, se podría desarrollar una aplicación de escritorio con métricas de negocio sobre su producción y venta, mientras que para los consumidores, se podría crear una aplicación móvil que muestre puntos de reciclaje y ofrezca incentivos por reciclar. Aunque el prototipo desarrollado para este trabajo final presenta una interfaz web unificada para todos los actores, el diseño modular facilita la creación de estas aplicaciones independientes en el futuro.

La elección tecnológica para esta capa estuvo focalizada en encontrar *frameworks* y librerías que promuevan la modularidad y la eficiencia durante el desarrollo. En primer lugar, se determinó utilizar React¹⁸ como librería base para la construcción de la interfaz, debido a su modelo de desarrollo basado en componentes, que promueve la reutilización de código. A su vez, para potenciar React, se optó por hacer uso del *framework* Next.js¹⁹, que agrega funcionalidades extra a React para facilitar aún más el desarrollo de aplicaciones web modulares y de alto rendimiento, combinando técnicas como generación de sitios estáticos y renderizado en el servidor. Por otro

¹⁸<https://es.react.dev/>

¹⁹<https://nextjs.org/docs>

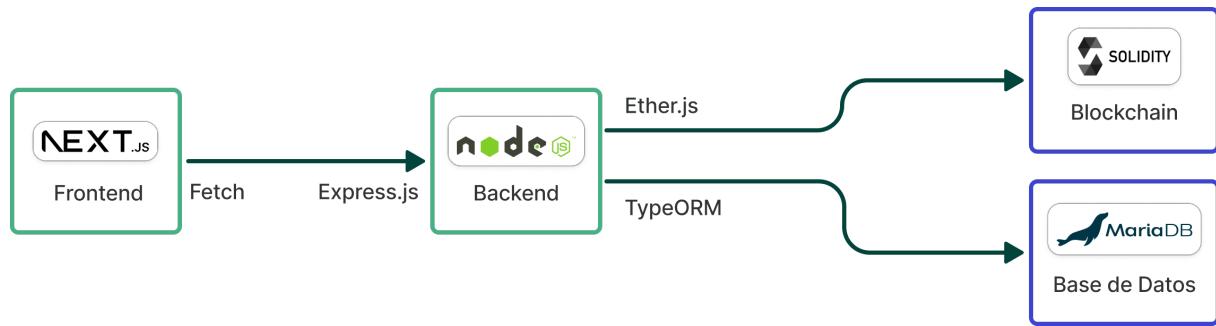


Figura 5.4: Arquitectura del sistema con librerías elegidas para comunicación entre módulos

lado, para el diseño de las interfaces web se utiliza el lenguaje de estilado CSS²⁰, que en este prototipo se complementó con el uso de la librería Tailwind CSS²¹ que promueve una estética moderna y una sintaxis simplificada dentro del código.

La comunicación del *frontend* con el *backend* se realiza exclusivamente a través de llamadas a la API REST. La interfaz no contiene la lógica de negocio, sino que actúa como un cliente ligero que envía datos al *backend* (por ejemplo, al registrar un nuevo lote) y recibe la respuesta, la cual es luego presentada al usuario. Este modelo garantiza que el *frontend* se enfoca en la interacción y la visualización, mientras la lógica crítica reside en el *backend*. Debido a que el *frontend* se enfoca únicamente en la visualización de información, es necesario implementar un sistema de autenticación y autorización unificado entre *frontend* y *backend* que permita a cada usuario visualizar e interactuar únicamente con las funcionalidades propias de su rol. Por este motivo, para gestionar la autenticación de usuarios en este proyecto, se determinó hacer uso del servicio Firebase Authentication²², que gestiona la autenticación de usuarios e implementa de forma abstracta el estándar de autorización OAuth 2.0²³, asegurando que solo los actores con los permisos adecuados puedan acceder a cada recurso del sistema.

Por último, en la Figura 5.4 se puede observar la arquitectura general del sistema, donde Next.js se utiliza para el desarrollo del *frontend*, que se comunica con el *backend*, que se implementa en Node.js y utiliza el *framework* Express.js para recibir las solicitudes, mientras que emplea librerías específicas para comunicarse con los contratos inteligentes en Solidity y con la base de datos relacional de MariaDB. Luego de definir la arquitectura de la aplicación web, incluyendo la definición de capas, la comunicación entre ellas y los lenguajes de programación a utilizarse, es posible comenzar con la etapa de diseño de componentes. En esta segunda etapa de diseño, se define la arquitectura interna de cada capa o módulo definido en la etapa anterior. En la próxima sección, se detallará el diseño de componentes de este sistema, que abarca la estructura interna del *frontend*, la API y la capa de datos.

²⁰<https://developer.mozilla.org/en-US/docs/Web/CSS>

²¹<https://tailwindcss.com/>

²²<https://firebase.google.com/docs/auth>

²³<https://oauth.net/2/>

5.2. Diseño de componentes

La segunda etapa de diseño, el diseño de componentes, tiene como objetivo detallar la arquitectura interna de los módulos definidos en la fase anterior. Su propósito es traducir la arquitectura de alto nivel en un plan de construcción específico, que incluye la estructura de la interfaz de usuario, la arquitectura de la API y el modelo de datos. Esta fase toma como punto de partida los requerimientos del sistema y la arquitectura de módulos previamente definida, y su resultado es un conjunto de especificaciones detalladas que guiarán la implementación de cada módulo del software. De esta forma, se busca asegurar la cohesión interna de cada módulo y su correcta interacción con los demás. A su vez, cada decisión de diseño tomada en esta etapa se verificará posteriormente en la fase de pruebas de integración, que valida la correcta comunicación entre los componentes del sistema.

A continuación, se presentarán las decisiones de diseño tomadas para cada uno de los módulos definidos en la arquitectura del sistema: la capa de datos, la capa de lógica de negocio (*backend*) y la capa de presentación (*frontend*).

5.2.1. Arquitectura de datos

El diseño de la arquitectura de datos representa un componente central de este trabajo, ya que debe integrar de manera transparente y eficiente la naturaleza descentralizada de la *blockchain* con la eficiencia y escalabilidad de una base de datos relacional para cumplir con los requerimientos no funcionales del sistema (RNF-01: Transparencia, RNF-03: Escalabilidad y RNF-06: Integridad). La solución definida durante la etapa de diseño de arquitectura propone un modelo de datos híbrido para optimizar el almacenamiento y la recuperación de información. En este esquema, la *blockchain* se utiliza como una capa de confianza inmutable, mientras que la base de datos relacional se encarga de la gestión de datos auxiliares que no requieren la inmutabilidad de la cadena de bloques. El diseño de componentes para esta arquitectura, por lo tanto, implica definir la estructura de los contratos inteligentes en la *blockchain* y el esquema de la base de datos relacional, asegurando que ambos sistemas interactúen de manera eficiente y coherente.

En primera instancia, se definió la representación de los datos, tanto en la *blockchain* como en la base de datos relacional. Con base en los requerimientos del sistema y la investigación sobre el ciclo de vida del vidrio, se optó por representar cada envase o conjunto de envases con una estructura distinta en cada etapa de su vida útil. Esta decisión se tomó debido a que los envases tienen metadatos, propietarios y agrupaciones diferentes en cada fase del proceso. El flujo de estados de los envases de vidrio, como se observa en la Figura 5.5, comienza con un lote de envases producido por el productor primario, que luego se vende a múltiples productores secundarios para crear lotes de productos envasados. Finalmente, los consumidores pueden llevar cada envase vacío a centros de reciclaje, donde se agrupan en lotes de reciclaje para su posterior reprocesamiento. Para mantener la trazabilidad completa del ciclo de vida, cada

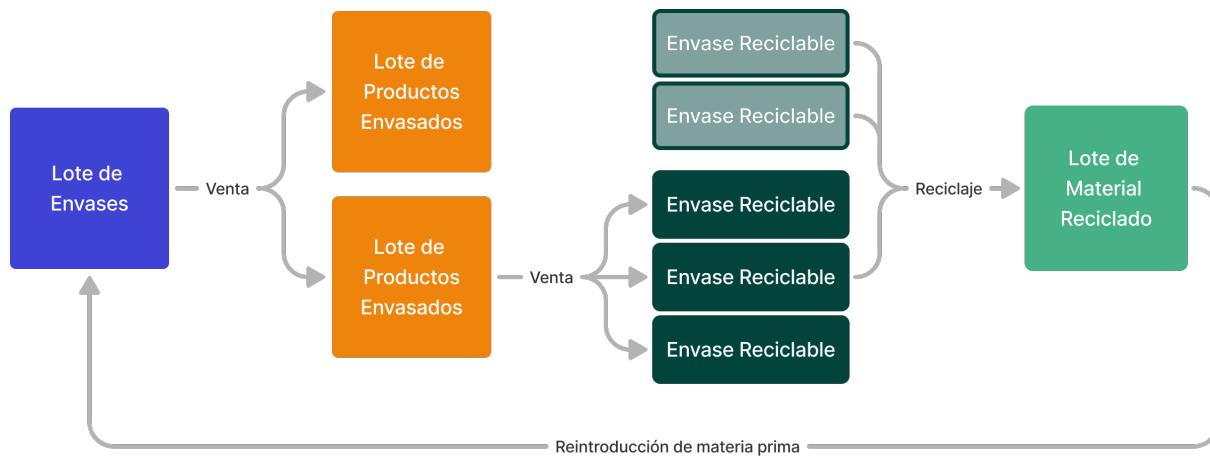


Figura 5.5: Diagrama de ciclo de vida de los envases de vidrio en el sistema

elemento posee una referencia a su ID (identificador) de origen, lo que permite rastrear, a partir de un envase reciclado, el lote original en el que fue producido, su productor y sus metadatos.

A partir de la representación de los datos, se diseñaron los contratos inteligentes en la *block-chain* para almacenar únicamente la información necesaria para garantizar la trazabilidad de los envases y la integridad de los datos, como un identificador único de cada lote de vidrio, sus metadatos esenciales de producción, el propietario actual y un historial de las transferencias de propiedad y de estado. Al igual que en un sistema de programación orientada a objetos (POO), los contratos en Solidity implementan atributos que guardan el estado y métodos que permiten interactuar con él. En lugar de un único contrato monolítico, se decidió implementar tres contratos que interactúan entre sí para lograr mayor modularidad, mantenibilidad y desacoplamiento. A continuación, se detalla la arquitectura de cada uno de ellos, incluyendo sus responsabilidades y la interacción entre ellos. Para cada contrato, se presenta un diagrama de clases UML (*Unified Modeling Language*) que ilustra su estructura interna, junto con las estructuras de datos relacionadas. Un diagrama de clases UML es una herramienta visual que se usa en la ingeniería de software para modelar la estructura de un sistema orientado a objetos. Describe las clases con sus atributos (datos) y métodos (funciones), y las relaciones entre

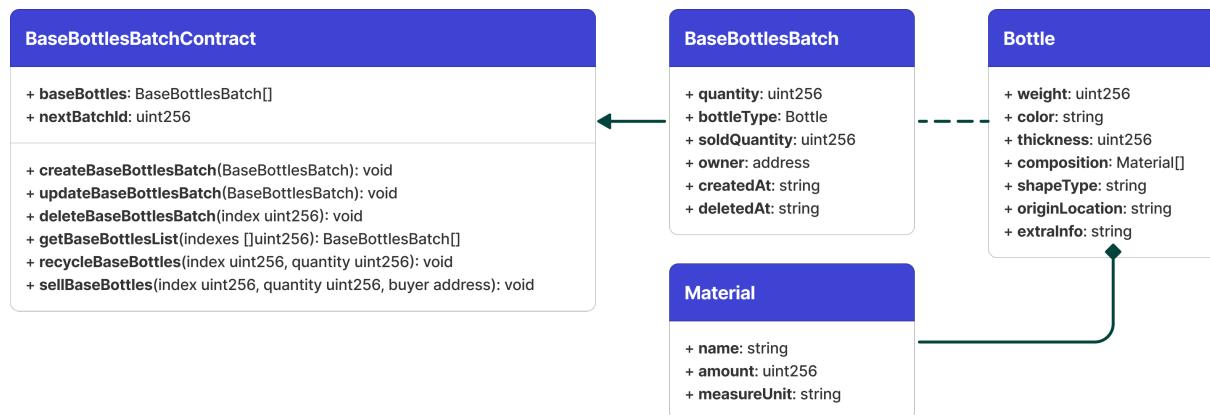


Figura 5.6: Diagrama UML del Contrato de Envases (BaseBottlesBatchContract)

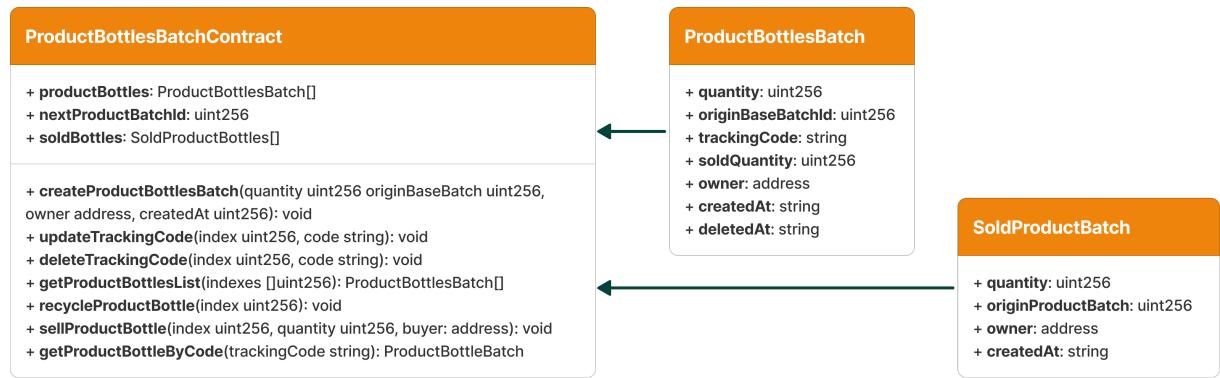


Figura 5.7: Diagrama UML del Contrato de Productos (*ProductBottlesBatchContract*)

ellas. Permite visualizar la composición y las interacciones del sistema, facilitando la comunicación y el entendimiento entre los desarrolladores. De forma práctica, actúa como un mapa conceptual que detalla los componentes principales del software antes de su implementación. Dentro del diagrama, los nombres de clases y propiedades se describen en inglés, siguiendo las convenciones de nomenclatura estándar vigentes en programación en entornos profesionales. Posteriormente, en la etapa de codificación, todo el código será implementado en inglés siguiendo este estándar, por lo que se utilizarán los nombres definidos en los diagramas UML para mantener la coherencia y facilitar la comprensión del código.

Contrato de envases: BaseBottlesBatchContract (Figura 5.6) es el punto de partida del ciclo de vida del vidrio, encargado de gestionar la producción inicial de los envases. Sus responsabilidades incluyen registrar lotes de botellas recién fabricadas por el productor primario y sus metadatos (como la cantidad y composición), así como gestionar la transferencia de propiedad a productores secundarios. Sus métodos permiten crear, actualizar y eliminar lotes, además de consultar la información histórica de los mismos. La información de este contrato es consumida por el contrato *ProductBottlesBatchContract*, proporcionando una referencia al lote original para el siguiente eslabón de la cadena de trazabilidad.

Contrato de productos: ProductBottlesBatchContract (Figura 5.7) gestiona la segunda fase

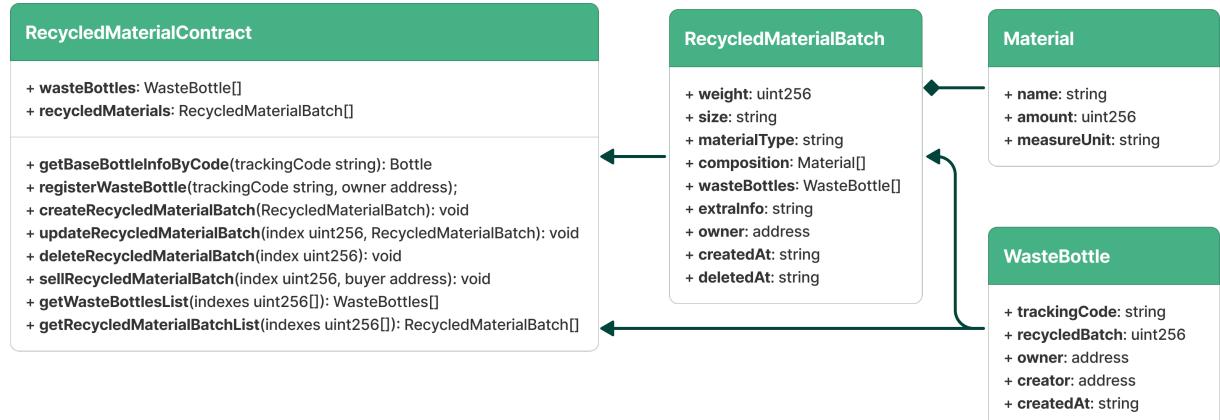


Figura 5.8: Diagrama UML del Contrato de Reciclaje (*RecycledMaterialContract*)

del ciclo de vida del vidrio, comprende el envasado y la comercialización de estos productos. Este contrato registra los lotes de productos terminados, asociando un código de seguimiento a cada uno y referenciando el lote de envases original del contrato *BaseBottlesBatchContract*. Sus funciones permiten crear lotes de productos, registrar su venta y marcar aquellos envases que se convierten en residuos. La información de seguimiento generada en este contrato juega un rol central en el sistema de trazabilidad, ya que el código de seguimiento introducido en este contrato actúa como el eslabón intermedio que permite vincular el lote de origen de un envase de *BaseBottlesBatchContract* con el envase en *RecycleMaterialContract* cuando el consumidor lo desecha para su posterior reciclaje.

Contrato de reciclaje: RecycleMaterialContract (Figura 5.8) cubre la gestión del final de la vida útil de los envases, desde su recolección como residuo hasta su procesamiento como material reciclado. Este contrato almacena los registros de los envases que han sido entregados para reciclaje, permitiendo crear nuevos lotes de material reciclado a partir de ellos. Sus métodos permiten registrar envases de desecho, crear lotes de material reciclado (agrupando envases previamente registrados) y transferir la propiedad de estos lotes a los productores primarios, cerrando de esta forma el ciclo de vida circular del vidrio. La información de seguimiento de los envases provista por el contrato *ProductBottlesBatchContract* es consumida por este contrato, que a su vez genera nuevos lotes de material que pueden ser reutilizados por el contrato *BaseBottlesBatchContract*.

De forma complementaria a la arquitectura de contratos inteligentes, el diseño de la base de datos relacional almacena la información de los usuarios (relacionada con los requerimientos funcionales de autenticación y autorización) y una referencia al ID y propietario de cada lote y envase registrado en la *blockchain*. La relación entre los datos de la *blockchain* y la base de datos relacional se establece mediante el identificador único del lote, que sirve como clave de enlace y permite realizar consultas de metadatos detallados de cada lote. Adicionalmente, en la base de datos relacional se guarda una referencia a cada envase reciclado por los consumidores. Esto permite que cada consumidor pueda acceder al listado de envases que ha reciclado pre-

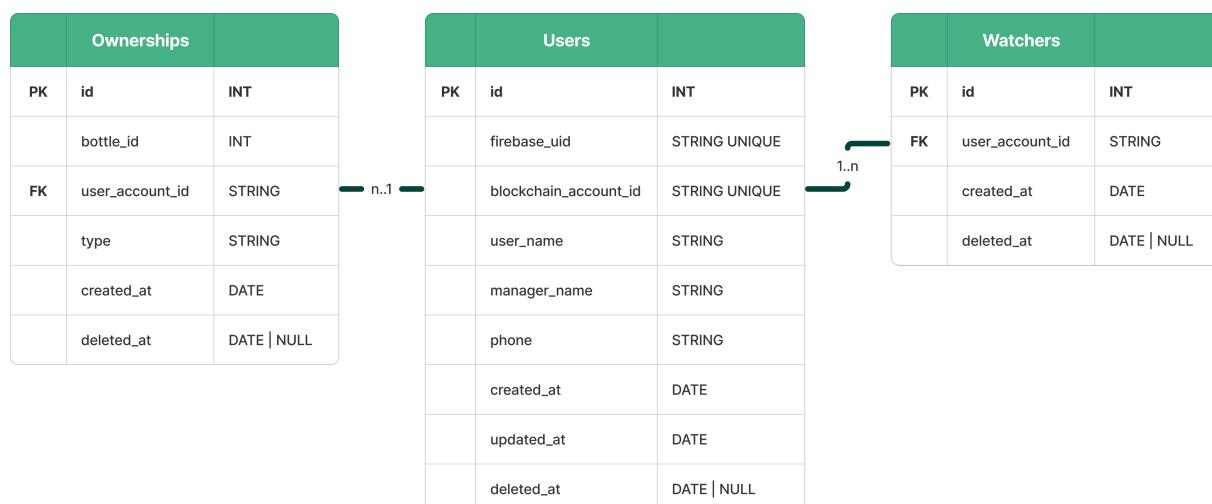


Figura 5.9: Diagrama Entidad-Relación (DER) del modelo de datos

vamente y consultar si efectivamente ha sido procesado, cumpliendo así con el requerimiento funcional asociado (RF-023). En la Figura 5.9 se presenta un diagrama (DER) que ilustra la relación entre las entidades de la base de datos.

5.2.2. Arquitectura backend

En el diseño de arquitectura de la capa *backend* se adoptó el patrón *Clean Architecture* (Arquitectura Limpia) [28], un modelo de diseño que prioriza la separación de las reglas de negocio de las dependencias externas. En este esquema, la implementación de la API se estructura en tres capas principales: *Routers*, *Handlers* y *Repositories*. Los *routers* reciben las solicitudes HTTP y las dirigen a los *handlers* correspondientes, luego los *handlers* contienen la lógica de negocio y orquestan las operaciones, por último, los *repositories* se encargan de la interacción con las fuentes de datos (ya sea la base de datos relacional o la *blockchain*). La comunicación entre estas capas es unidireccional, lo que significa que las capas externas solo pueden acceder a las capas más internas, reforzando así la independencia de la lógica de negocio. La Figura 5.10 ilustra cómo las interfaces externas (como el *frontend* y otros sistemas) interactúan únicamente con los controladores (*routers*), los cuales a su vez interactúan con los casos de uso (*handlers*), que finalmente se comunican con las entidades (*repositories*) para acceder al dominio (datos).

La elección del patrón *Clean Architecture* se justifica por su capacidad para generar un sistema altamente desacoplado, lo que se traduce en una mayor mantenibilidad del sistema y flexibilidad ante futuros cambios. La lógica de negocio, situada en el núcleo de la arquitectura (*Handlers*), se mantiene independiente de las tecnologías de implementación (infraestructura), la presentación de los datos (*Routers*) y las bases de datos (*Repositories*). Esto es particularmente ventajoso en un sistema de trazabilidad como el planteado en este trabajo, donde la lógica de negocio debe ser estable, pero la interfaz de usuario y las integraciones con otros sistemas

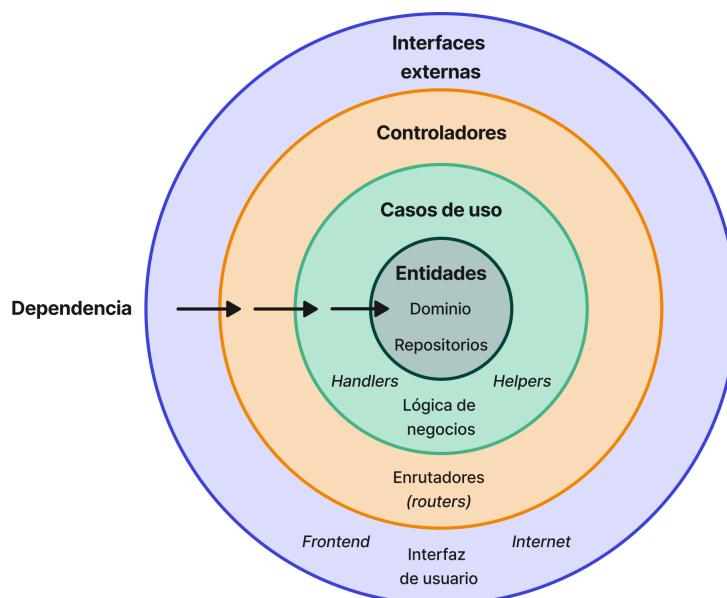


Figura 5.10: Modelo Clean Architecture

(como plataformas de gestión o dispositivos IoT) pueden evolucionar. A su vez, en este proyecto la arquitectura de la API se ha dividido en módulos de dominio (por ejemplo, gestión de usuarios o trazabilidad de lotes), cada uno de los cuales expone un conjunto de *endpoints* a través de una API REST.

En particular, para orquestar la comunicación entre la API, la base de datos relacional y la *blockchain*, se implementó un patrón de repositorios que unifica las operaciones de lectura y escritura. De esta forma, el *handler* puede manejar todos los datos de manera uniforme, sin importar si el repositorio los obtuvo de la *blockchain* o de la base de datos relacional, ya que esta lógica de acceso a datos se abstrae en el repositorio. Por ejemplo, al registrar un nuevo lote, el *handler* valida la información y luego instruye al repositorio de la *blockchain* para registrar la transacción y al repositorio de la base de datos relacional para almacenar la referencia del lote. En este caso, el patrón de diseño *Clean Architecture* permite abstraer la complejidad de la arquitectura híbrida, proporcionando una interfaz de programación unificada a la capa de lógica de negocio.

5.2.3. Arquitectura frontend

La interfaz de usuario del sistema se diseñó como una aplicación web, con el objetivo de proporcionar una experiencia de usuario fluida, accesible desde cualquier dispositivo con conexión a Internet y sin necesidad de instalar software adicional. Debido al alcance limitado del trabajo, se decidió implementar una interfaz diseñada para computadoras de escritorio, dado que es el caso de uso más frecuente en sistemas de gestión y trazabilidad. La aplicación podrá accederse en dispositivos móviles, pero no se ha priorizado la implementación adaptada para estos dispositivos, por lo que la interfaz puede resultar menos amigable en pantallas pequeñas. La interfaz se estructuró mediante una arquitectura basada en componentes, un patrón de diseño que promueve la creación de elementos reutilizables, modulares e independientes, que es el patrón recomendado por librerías como React y frameworks como Next.js.

Para la estructuración interna del código, se eligió implementar una arquitectura Modelo Vista-Controlador (MVC), que establece una clara separación de responsabilidades: la vista implementa la interfaz de usuario, el controlador maneja la lógica y las interacciones, y el modelo (en este caso, un servicio) se comunica con el *backend*. Esta metodología, combinada con la arquitectura de componentes, facilita una construcción rápida y consistente de cada vista, al mismo tiempo que mejora la mantenibilidad del código a largo plazo, ya que cada componente puede ser actualizado sin afectar otras partes del sistema. En la Figura 5.11 se ilustra la arquitectura de componentes y módulos del *frontend*, donde los usuarios navegan por las distintas vistas de la aplicación y cada una está compuesta por un módulo con un componente de vista, un controlador que implementa la lógica y un servicio que actúa como modelo, el cual se encarga de enviar solicitudes a la API *backend* para interactuar con los datos.

La estructura de la interfaz de usuario se organizó en módulos funcionales por cada rol de usuario, lo cual se alinea con la división de responsabilidades del diseño de arquitectura de

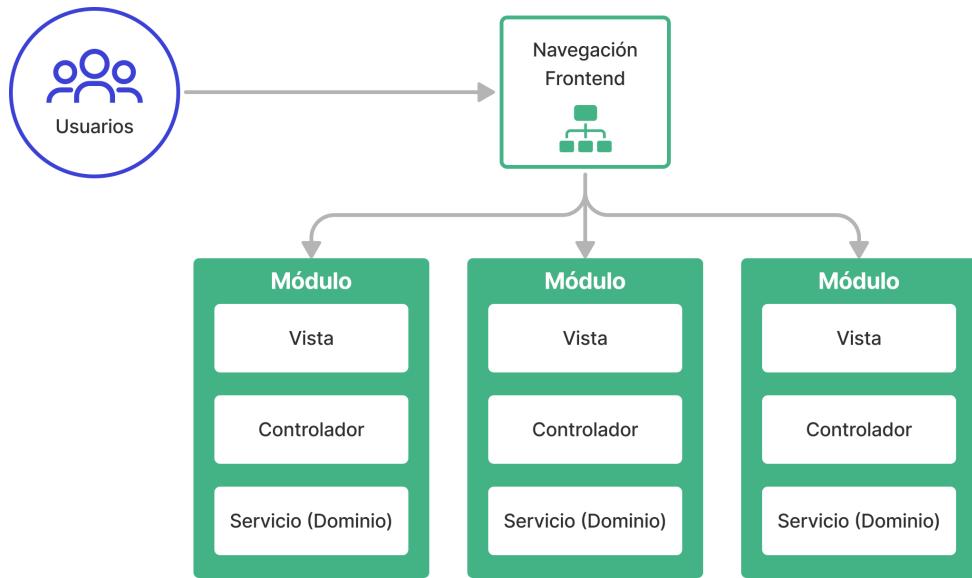


Figura 5.11: Arquitectura de módulos frontend

backend y capa de datos. Por ejemplo, se definieron vistas específicas para el registro y la gestión de lotes por parte de los productores, junto con una interfaz de consulta para los consumidores. En la Figura 5.12 se puede observar el diseño de la navegación de la aplicación, que consiste en una barra lateral comprimida que se expande al posicionar el cursor sobre ella, mostrando accesos directos a las pantallas disponibles según el rol del usuario autenticado. El contenido de la barra lateral varía dinámicamente según el rol del usuario autenticado para permitir un acceso rápido a las funcionalidades relevantes para cada tipo de usuario. En el Apéndice D se encuentran documentados los flujos de usuario para cada rol con imágenes de sus respectivas



Figura 5.12: Diseño de casos de la barra lateral de navegación de la aplicación para cada rol



Figura 5.13: Identidad de marca de la aplicación

pantallas y las acciones que conectan el flujo.

Por otro lado, para asegurar una identidad visual consistente en la aplicación, se definió un sistema de diseño con una paleta de colores basada en tonalidades de verde con el objetivo de transmitir el compromiso ambiental del proyecto mediante la economía circular, así como también tipografías e iconografía que complementan esta estética. En la Figura 5.13 se puede observar una muestra de la identidad de marca de la aplicación, a través de su paleta de colores, que posee un color primario (verde), un color secundario (verde oscuro) y colores complementarios (azul y anaranjado) para resaltar elementos importantes en la interfaz.

Con la definición de las especificaciones detalladas de la arquitectura de componentes para el *frontend*, el *backend* y la capa de datos, el proceso de desarrollo posterior puede ser más fluido, organizado y predecible. En el siguiente capítulo se abordará el proceso de implementación del prototipo tecnológico, a través del cual se materializarán los diseños aquí descritos en un software funcional que cumpla con los requerimientos establecidos.



IMPLEMENTACIÓN

La fase de implementación representa el proceso de traducción del diseño de software a código ejecutable, constituyendo el puente entre la teoría y la práctica. En el marco del modelo en V, este proceso es una de las etapas finales de la fase descendente, que a su vez marca el inicio de la fase ascendente, ya que la implementación de cada módulo va acompañada de la ejecución de pruebas unitarias. Este enfoque iterativo de desarrollo y validación temprana busca asegurar que la funcionalidad de cada componente se verifique de forma continua para minimizar la aparición de errores cuando el software se despliegue en un entorno real. El proceso de implementación implica tanto la escritura de código, como también la integración de los distintos módulos del sistema, la documentación del código y la preparación del entorno para el despliegue.

La implementación del software se llevó a cabo siguiendo la planificación elaborada a partir de las historias de usuario junto con el diseño del sistema. Durante la ejecución de esta etapa, se utilizó la herramienta Jira para gestionar las tareas en curso y el progreso del desarrollo. El cronograma original enfrentó desviaciones debido a la superposición de actividades académicas y compromisos imprevistos, pero la flexibilidad en la gestión del proyecto permitió la adaptación, posibilitando el cumplimiento de los objetivos del trabajo. En esta fase de desarrollo, se implementa e integra cada uno de los módulos definidos en el proceso de diseño, asegurando su funcionamiento de forma aislada y en conjunto.

El proceso de desarrollo se estructuró para seguir un flujo de trabajo lógico. En primer lugar, se crearon los contratos inteligentes, que conforman la capa más interna del prototipo y definen la lógica de las transacciones en la *blockchain*. Posteriormente, se construyó la API, que actúa como intermediario para interactuar con los contratos. Finalmente, se desarrolló la interfaz de usuario, que sirve como la capa de presentación. Este enfoque se adoptó con el objetivo de garantizar que cada componente estuviera operativo y probado antes de proceder a la siguiente

capa que interactúa con él. A nivel de dominio, el desarrollo siguió secuencialmente el ciclo de vida del vidrio (productor primario, secundario, consumidor, reciclador) para mantener la coherencia del sistema. En la siguiente sección se detalla el proceso de generación de código llevado a cabo durante la implementación del prototipo (Sección 6.1). Seguidamente, se describe el proceso de despliegue del sistema en un entorno de pruebas para ser utilizado en las etapas de validación posteriores (Sección 6.2). Finalmente, se aborda la estrategia de documentación del software desarrollado (Sección 6.3).

6.1. Generación de código

La implementación del prototipo se realizó en un entorno de desarrollo local, siguiendo un flujo de trabajo que priorizó la separación por capas para gestionar las dependencias del sistema. Como la lógica de negocio central recae en los contratos inteligentes, su implementación fue la primera en abordarse. La capa de datos no solo define la lógica de las transacciones en la *blockchain*, sino que también establece el modelo de datos base que utilizan las capas superiores. Una vez que los contratos estuvieron completamente desarrollados y probados en base a las especificaciones de requerimientos y diseño, se procedió a la construcción de la API. Esta capa actúa como un intermediario entre los contratos y el *frontend*, siendo responsable de traducir las peticiones de la interfaz de usuario en transacciones y llamadas a los contratos. Finalmente, se construyó la interfaz de usuario, que se conecta a la API para poder presentar la información al usuario y permitir la interacción con el sistema.

Para cada funcionalidad del sistema, es posible identificar un flujo de trabajo común que se repite en cada módulo. En primer lugar, se desarrolla la función correspondiente en los contratos inteligentes, como se puede ver en el Código 1 con la función para crear un lote de botellas de vidrio, que recibe los parámetros necesarios, crea un nuevo lote y lo almacena en la *blockchain*, emitiendo un evento para notificar su creación. A continuación, se escribe un *endpoint* en la API que llama a esta función del contrato inteligente, tal como se puede observar en el Código 2, que contiene la implementación del repositorio de este *endpoint*. Finalmente, se implementa la interfaz de usuario que permite a los usuarios finales interactuar con esta funcionalidad a través de un formulario con un botón de envío, como se muestra en el Código 3 con el componente de React que maneja la creación del lote de botellas. Este componente recoge los datos del formulario, llama a un servicio que envía una solicitud a la API y maneja la respuesta para proporcionar mensajes de retroalimentación al usuario. Durante la ejecución de este código, el flujo comienza en el *frontend*, que envía una solicitud a la API cuando el usuario presiona el botón para crear un lote. La API procesa esta solicitud y llama a la función correspondiente en el contrato inteligente, que ejecuta la lógica de negocio y actualiza el estado en la *blockchain*, emitiendo eventos que la API puede escuchar para actualizar la base de datos relacional con la información del nuevo lote creado.

El proceso de desarrollo se concibió de manera iterativa, donde la escritura de código se alternó con la creación de pruebas unitarias. Este método permitió verificar el correcto funcionamiento

Código 1: Función para la creación de un lote de botellas de vidrio en la blockchain (Solidity)

```

1  function createBaseBottlesBatch(uint256 quantity, Bottle memory bottleType, address owner,
2    ↳ string memory createdAt) external onlyContractOwner {
3    BaseBottlesBatch memory newBatch = BaseBottlesBatch({
4      id: nextBatchId,
5      quantity: quantity,
6      bottleType: bottleType,
7      owner: owner,
8      soldQuantity: 0,
9      createdAt: createdAt,
10     deletedAt: ""
11   });
12
13   baseBottlesBatches[nextBatchId] = newBatch;
14   emit BaseBatchCreated(nextBatchId, owner);
15   nextBatchId++;
16 }
```

Código 2: Función del repositorio de la API para la creación de un lote de botellas de vidrio (Node.js)

```

1  async function CreateBaseBottlesBatch(
2    batch: BaseBottlesBatch,
3  ): IResult<number> {
4    const createdAt = new Date().toISOString();
5    const result = await this.callContractMethod(
6      'createBaseBottlesBatch',
7      batch.quantity,
8      batch.bottleType,
9      batch.owner,
10     createdAt,
11   );
12   if (!result.ok) return result;
13
14   // Get created batch id from events emmited or default to 0.
15   const batchId = result.data.find((event) => event.name === 'BaseBatchCreated')?.batchId ?? 0;
16
17   return { ok: true, status: StatusCodes.OK, data: batchId };
18 }
```

de cada componente de forma individual, asegurando que las funciones y módulos cumplieran con las especificaciones de diseño. Gracias al diseño de sistema realizado previamente, la implementación de cada módulo se llevó a cabo de manera sistemática sin bloqueos, pero esto no implicó que no surgieran desafíos técnicos durante la integración de los componentes. Un ejemplo destacable durante la implementación fue el desafío de adaptar la API a la naturaleza inherente de los contratos inteligentes, los cuales no retornan datos de forma nativa, sino que emiten eventos notificando cambios en su estado. Esta particularidad técnica de la *blockchain*

Código 3: Función para la creación de un lote de botellas de vidrio en el frontend (Next.js)

```

1  function CreateBaseBottlesBatchButton = ({ form }: Props {
2    async function handleSubmit() {
3      const payload = data;
4      const { ok } = await createBatchService(payload);
5      if (ok) {
6        toast.success('Lote de botellas creado correctamente');
7      } else {
8        toast.error('Ocurrió un error al crear el lote de botellas');
9      }
10    };
11
12    async function createBatchService(data: BottleBatch) {
13      try {
14        const res = await fetch(`/producer/batch`, {
15          method: 'POST',
16          body: JSON.stringify(data),
17        });
18        const data = await res.json();
19        return { ok: true, data: data.batchId };
20      } catch {
21        return { ok: false, data: null };
22      }
23    }
24
25    return (
26      <button type="submit" onClick={handleSubmit}>Crear Lote</button>
27    );
28  }

```

requirió que la capa de la API fuera adaptada para escuchar estos eventos, capturando información como los identificadores únicos de los lotes de vidrio recién creados para su posterior almacenamiento en la base de datos relacional. Esta solución técnica permitió demostrar la viabilidad de la arquitectura híbrida propuesta, asegurando la sincronización de la información entre la *blockchain* y la base de datos complementaria.

A nivel de dominio, la implementación de las funcionalidades siguió el ciclo de vida del vidrio para mantener una mayor coherencia. El desarrollo se inició con las funcionalidades del productor primario, continuó con las del productor secundario, luego con las del consumidor y, finalmente, con las del centro de reciclaje, cerrando así el ciclo de trazabilidad. Una vez que se completaron las funcionalidades para cada actor, se desarrolló la funcionalidad de seguimiento de extremo a extremo, que permite visualizar el historial completo de un envase desde su producción hasta su revalorización. Este enfoque permitió que el flujo del proceso de trazabilidad se construyera de manera lógica y progresiva. Una vez que todas las funcionalidades del prototipo fueron implementadas a nivel de código, se procedió a realizar el despliegue del prototipo en un entorno de pruebas, como se detalla en la siguiente sección.

6.2. Despliegue

Una vez que cada módulo del sistema fue implementado y verificado con pruebas unitarias en un entorno local, se procedió a realizar un despliegue del sistema en un entorno de pruebas de características similares a un entorno productivo real. El objetivo principal de esta acción fue demostrar la operatividad del prototipo y proveer un entorno estable y accesible en línea para utilizar en las etapas de pruebas de sistema y pruebas de aceptación. Para ello, se eligieron plataformas gratuitas que permitieran la exposición pública de los componentes del sistema, lo cual facilitó la validación por parte de terceros y la demostración de la viabilidad del proyecto dentro del alcance de un trabajo académico.

En primer lugar, para la modularización y gestión del despliegue, se configuraron contenedores de Docker¹ para cada uno de los componentes del sistema. El uso de esta tecnología permitió empaquetar la aplicación y sus dependencias en unidades portables y autónomas, para poder garantizar la reproducibilidad del trabajo en cualquier entorno y facilitar la futura transición del prototipo a un entorno productivo evitando problemas de compatibilidad debido a diferencias en la configuración del entorno. Por ejemplo, el contenedor de la API incluye el servidor Node.js junto con las dependencias del proyecto y puede construirse y ejecutarse utilizando los comandos listados en el Código 4 utilizando variables de entorno para configurar parámetros como el puerto para el servicio y las credenciales de la base de datos.

Código 4: Comandos para construir y ejecutar el contenedor de la API con Docker

```
1 # Build the Docker image
2 docker build -t $DOCKER_TAG .
3
4 # Run the Docker container
5 docker run -d -e PORT -e DB_HOST -e DB_PORT -e DB_USER -e DB_PASS \
6 -e FIREBASE_CONFIG -e PRIVATE_KEY -e PROVIDER_URL -e BASE_BATCH_CONTRACT_ADDRESS \
7 -e PRODUCT_BATCH_CONTRACT_ADDRESS -e RECYCLING_CONTRACT_ADDRESS \
8 -p $EXTERNAL_PORT:$PORT --name $DOCKER_TAG $DOCKER_TAG
```

Posteriormente, el despliegue se llevó a cabo de forma diferente para cada tecnología. La API de *backend* se desplegó en una plataforma de alojamiento web², los contratos inteligentes se publicaron en una red de pruebas de Ethereum³ y la interfaz de usuario se puso a disposición en un servicio de hospedaje web estático⁴. Esta configuración logró que el prototipo fuera accesible en línea y a su vez permitió la ejecución de pruebas de integración y aceptación de usuarios en un entorno que replicaba las condiciones de uso finales. Si bien el prototipo se implementó en un entorno de pruebas, fue diseñado con una arquitectura escalable, con el objetivo de facilitar una transición sin fricciones a un entorno productivo en el futuro.

¹ <https://docker.com/>

² <https://cloud.google.com/>

³ <https://sepolia-optimism.etherscan.io/>

⁴ <https://vercel.com/>

Finalmente, todos los detalles del proceso de despliegue, incluyendo las instrucciones para la configuración del entorno local y la replicación del despliegue en producción, se documentaron exhaustivamente en cada repositorio del proyecto. Esta documentación asegura que otros desarrolladores o investigadores puedan reproducir el entorno de desarrollo y desplegar el sistema de manera autónoma, contribuyendo a la transparencia y accesibilidad del trabajo realizado. En la siguiente sección se detalla la estrategia de documentación adoptada para el prototipo desarrollado.

6.3. Documentación

Como parte integral del proceso de ingeniería de software, la documentación busca asegurar la mantenibilidad del código, facilitar la colaboración futura y consolidar el conocimiento técnico del proyecto. En este trabajo, el prototipo se documentó en tres niveles: la documentación del código fuente, la interfaz de la API y la configuración del proyecto.

En el primer nivel, se incluyeron comentarios directamente en el código fuente de cada repositorio, tanto en los contratos inteligentes, como en la API y el *frontend*. Esto permite que el código sea autoexplicativo y más fácil de comprender para otros desarrolladores o para futuros trabajos de mantenimiento.

En el segundo nivel, se utilizó la especificación de OpenAPI⁵ para describir la interfaz de la API del *backend*, detallando todos los *endpoints*, parámetros y formatos de solicitudes y respuestas. A partir de este estándar, se utilizó una librería para generar un sitio web interactivo que presenta esta documentación de manera accesible. Exponer esta documentación facilita que la API sea interoperable y pueda ser consumida por cualquier otra aplicación cliente, por ejemplo, en el caso de que se desarrollem nuevas interfaces de usuario o aplicaciones móviles que se conecten a la misma API.

En la Figura 6.1 se puede observar una captura de pantalla del sitio web con la documentación del *endpoint* de creación de lotes de botellas. Este sitio web fue generado automáticamente a partir de la especificación OpenAPI implementada en el Código 5 y contiene información sobre el *endpoint* como el método HTTP, la URL, los parámetros esperados, los códigos de respuesta y ejemplos de solicitudes. Esta documentación interactiva sirve como referencia para desarrolladores que deseen integrar o extender la funcionalidad de la API, proporcionando una guía clara y confiable sobre cómo interactuar con el sistema.

Finalmente, en el tercer nivel, cada repositorio cuenta con un archivo *README* que actúa como una guía de referencia rápida para la configuración y operación del sistema. Estos archivos detallan los requisitos técnicos, la estructura del proyecto y los comandos para ejecutar pruebas y desplegar el sistema. A su vez, también se incluyó una explicación de la arquitectura de cada repositorio y una serie de enlaces de utilidad que pueden ser de ayuda para los desarrolladores

⁵ Estándar OpenAPI: <https://swagger.io/specification/>

Código 5: Descripción del endpoint para crear un lote de botellas mediante el estándar OpenAPI

```

1 /**
2  * POST /producer/batch
3  * @summary Crea un nuevo lote de botellas base
4  * @tags 2. Productor Primario - Operaciones para productores de botellas
5  * @param {CreateBaseBottlesBatchRequest} request.body.required - Datos del lote
6  * @security BearerAuth
7  * @return {CreationResponse} 201 - Lote creado exitosamente
8  * @return {ErrorResponse400} 400 - Datos de lote inválidos
9  * @return {ErrorResponse401} 401 - No autorizado
10 * @example request - Ejemplo de lote
11 *
12 *   "quantity": 100,
13 *   "bottleType": {
14 *     "weight": 500,
15 *     "color": "Verde",
16 *     "thickness": 2,
17 *     "shapeType": "Cuello alto",
18 *     "originLocation": "Argentina",
19 *     "extraInfo": "Vidrio reciclado",
20 *     "composition": [{"name": "Calcín", "amount": 100, "measureUnit": "%" }]
21 *   },
22 *   "createdAt": "2025-05-22T10:00:00Z"
23 * }
24 */
25 async function CreateBaseBottlesBatch(req: Request, res: Response) {
26   /* Código del router... */
27 }

```

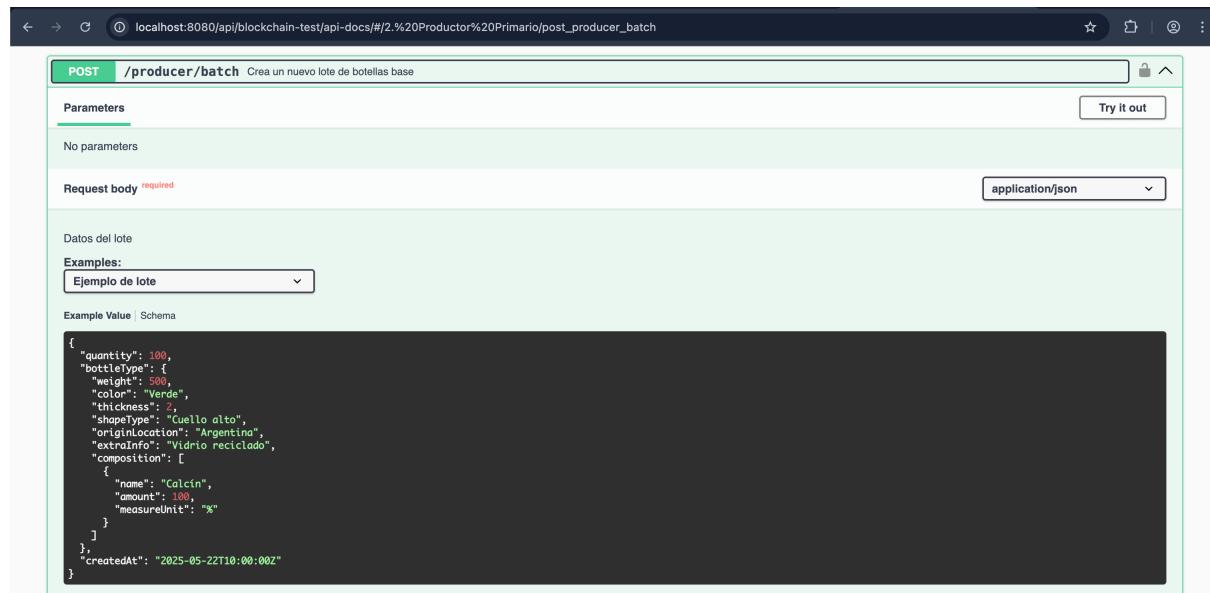


Figura 6.1: Documentación interactiva del endpoint para crear un lote de botellas generada automáticamente a partir de la especificación OpenAPI

que comiencen a interactuar con el código del proyecto.

Concluido el proceso de implementación y documentación del código, el prototipo del sistema de trazabilidad del vidrio se considera listo para la fase de validación. La construcción de cada componente, desde los contratos inteligentes hasta la interfaz de usuario, ha sido exhaustivamente verificada con pruebas unitarias, sentando las bases para una evaluación más rigurosa. El siguiente capítulo abordará en detalle el proceso de verificación del sistema, detallando la metodología de pruebas unitarias, de integración, de sistema y de aceptación del usuario para asegurar la calidad y el correcto funcionamiento del prototipo en su conjunto.



PRUEBAS

El presente capítulo aborda el proceso de pruebas del prototipo, un componente central del modelo en V de ingeniería de software que rige el desarrollo de este trabajo. Este proceso de validación, que abarca la totalidad de la segunda mitad del modelo, tiene como objetivo principal verificar que el prototipo se alinee con los requerimientos y el diseño definidos en las fases previas. El proceso de pruebas se estructura en un ciclo progresivo, donde la granularidad de la validación disminuye a medida que se avanza en las etapas, comenzando por las unidades de código más pequeñas y atómicas (pruebas unitarias), avanzando con pruebas de integración entre los módulos del sistema, hasta alcanzar la validación del sistema en su totalidad (pruebas de sistema y de aceptación). La naturaleza de estas pruebas varía entre automatizada y manual. Las pruebas automatizadas, si bien requieren una inversión inicial, se ejecutan de manera instantánea y repetible, lo cual resulta ideal para verificar comportamientos de forma constante. Por su parte, las pruebas manuales, aunque resultan más lentas de ejecutar, permiten una validación completa de los flujos de usuario y la experiencia general del sistema.

A continuación, se detallan las cuatro etapas de pruebas integrales realizadas en el proyecto:

- **Pruebas unitarias:** se enfocan en la validación del código a nivel de componente.
- **Pruebas de integración:** verifican la interacción entre los módulos del sistema.
- **Pruebas de sistema:** validan el cumplimiento de los requerimientos funcionales y no funcionales del prototipo en su totalidad.
- **Pruebas de aceptación con usuarios:** validan que el prototipo cumpla con las expectativas y necesidades del usuario final.

La Tabla 7.1 presenta una comparación de estas etapas, destacando sus características y su alcance, como una guía visual para comprender la metodología de prueba aplicada.

Tabla 7.1: Comparación de las etapas de prueba del prototipo de trazabilidad de vidrio

Etapa de Prueba	Frec. Ejecución	Tipo	Complejidad	Alcance
Pruebas unitarias	Continua (por cada cambio)	Automatizada	Baja	Componentes
Pruebas de integración	Antes de cada despliegue	Automatizada	Media	Interacción de componentes
Pruebas de sistema	Al finalizar la implementación	Manual	Media-Alta	Requerimientos funcionales
Pruebas de aceptación	Al finalizar la implementación	Manual	Alta	Experiencia del usuario

A lo largo de este capítulo se detallará la gestión de las incidencias halladas durante las pruebas. Los errores detectados en cada etapa de prueba fueron registrados y se les dio seguimiento en la herramienta Jira, asegurando que cada problema se resolviera antes de avanzar a la siguiente fase. A continuación, se describen en detalle cada una de las etapas de prueba mencionadas, proporcionando una visión completa del proceso de validación del prototipo. Adicionalmente, el Apéndice E contiene los detalles de la ejecución de cada prueba, los resultados obtenidos y la gestión de las incidencias documentadas.

7.1. Pruebas unitarias

Las pruebas unitarias constituyen la base de la pirámide de pruebas y se corresponden directamente con la fase de codificación del modelo en V. Su objetivo es validar la unidad más pequeña de código de forma aislada del resto del sistema, por ejemplo, un método de un contrato inteligente, un *endpoint* de *backend* o un componente reutilizable del *frontend*. La naturaleza atómica de estas pruebas permite verificar que cada componente individual se comporte de acuerdo con las especificaciones de diseño antes de ser integrado con otras partes del prototipo. Para este proyecto, se implementaron pruebas unitarias automatizadas para permitir una verificación continua de la integridad del código a lo largo de todo el proceso de implementación y luego de cada modificación de código posterior.

El desarrollo de cada módulo del prototipo se realizó de forma conjunta con la escritura de sus pruebas unitarias. Se utilizó el *framework* Jest¹ en las tres capas del proyecto (contratos inteligentes, API y *frontend*), aunque con configuraciones específicas para cada entorno. Por ejemplo, en los contratos inteligentes, las pruebas unitarias se orientaron a verificar que la lógica de negocio se ejecute correctamente y que el estado de los contratos cambie como se espera. En el *backend*, se enfocaron en validar la lógica de negocios. En el *frontend*, se validó el comportamiento de los componentes, su estado interno y la interacción con la API. En el

¹ <https://jestjs.io/>

Código 6: Código fuente de prueba unitaria para solicitud inválida en el endpoint de creación de lote en la API backend

```

1 describe('POST /producer/batch', () => {
2   it('should not create batch if invalid body', async () => {
3     const invalidBatch = { quantity: '100' }; // Missing required fields
4
5     const res = await request(app)
6       .post(`${BASE_PATH}/producer/batch`)
7       .set('Authorization', `Bearer userWithId-userRole-PRODUCER`)
8       .send(invalidBatch)
9       .expect(400);
10
11     expect(res.body.status).toBe(400);
12     expect(res.body.data).toBe(null);
13   });
14 });

```

Código 6 se muestra un ejemplo de la prueba unitaria implementada para el caso de solicitud inválida en el *endpoint* de creación de un lote de vidrio en la API *backend*, mientras que en la Figura 7.1 se puede observar la salida de la ejecución de todas las pruebas unitarias sobre ese *endpoint*, donde se imprimen los títulos de los casos de prueba ejecutados y un contador indicando que todas las pruebas se ejecutaron exitosamente.

```

PASS tests/unit/producer.test.ts
Producer API
  POST /producer/batch
    ✓ should create a new base bottles batch (23 ms)
    ✓ should not create batch if invalid body (5 ms)
    ✓ should not create batch if user is not authenticated (2 ms)
    ✓ should handle create error when user doesn't exist (1 ms)
    ✓ should handle create error when blockchain fails (3 ms)
    ✓ should handle create error when ownership repository fails (2 ms)

Test Suites: 7 skipped, 1 passed, 1 of 8 total
Tests:       311 skipped, 6 passed, 317 total
Snapshots:  0 total
Time:        2.473 s, estimated 3 s
Ran all test suites matching /tests\unit/i with tests matching "POST /producer/batch".

```

Figura 7.1: Salida de la ejecución de las pruebas unitarias del endpoint de creación de lote en la API backend

Un indicador representativo de la calidad de las pruebas unitarias es la cobertura de código (conocida comúnmente como *coverage*), que mide el porcentaje de código fuente ejecutado por las pruebas. Este valor, si bien no garantiza la ausencia de errores, es una herramienta útil para evaluar la robustez del código. Es importante destacar que, frecuentemente, una cobertura de código del 100 % puede no ser alcanzable o justificable, ya que existen bloques de código que no se pueden ejecutar en pruebas unitarias (debido a limitaciones del *framework* de pruebas), fragmentos de código que son difíciles de probar de forma aislada, o incluso casos de uso que se consideran de bajo riesgo de errores o casos donde se toma la decisión de limitar el esfuerzo invertido en escribir pruebas unitarias por escasez de tiempo o recursos. Debido a estos desafíos, el objetivo en cada proyecto es alcanzar un umbral de cobertura que mitigue el riesgo

de fallos en el funcionamiento del sistema, sin incurrir en un esfuerzo desproporcionado en la escritura de pruebas unitarias. Los requerimientos de cobertura mínima para este proyecto se definieron en función de la criticidad de cada módulo del sistema:

- **Contratos inteligentes:** se requirió una cobertura mínima del 100 %. Dado que los contratos son la base del sistema de trazabilidad y no pueden modificarse una vez desplegados, resulta fundamental garantizar que todos los caminos de ejecución del código estén cubiertos para minimizar el riesgo de errores una vez desplegados. Este es un estándar de la industria para contratos inteligentes.
- **API backend:** se requirió una cobertura mínima del 80 %. Dado que la API representa el componente central del sistema como responsable de la comunicación con la *blockchain* y la base de datos, resulta relevante garantizar la calidad del código para minimizar el riesgo de errores en el manejo de datos. Por este motivo, se ha elegido este umbral de cobertura, que es un estándar habitual en la industria para aplicaciones de propósito general.
- **Aplicación frontend:** se estableció una cobertura mínima del 60 %. Dado que el *frontend* del prototipo tiene fines demostrativos y no es una parte crítica del sistema, este umbral se consideró suficiente para garantizar el correcto funcionamiento de la interfaz de usuario sin requerir un esfuerzo excesivo en la escritura de pruebas.

El proceso de pruebas unitarias permitió identificar y corregir algunos errores de manera temprana. Los errores detectados en esta etapa fueron resueltos de manera inmediata, ya que las pruebas unitarias son cercanas a la implementación y permiten una rápida retroalimentación sobre el estado del código. En la Tabla 7.2 se muestra un resumen del objetivo de cobertura de cada módulo del sistema, el umbral alcanzado y la cantidad de pruebas unitarias implementadas, mientras que en el Apéndice E se puede consultar en mayor detalle el listado de casos de prueba de cada módulo.

Tabla 7.2: Resumen de las pruebas unitarias implementadas en cada módulo del sistema

Módulo	Coverage Objetivo	Coverage Alcanzado	Cant. de Pruebas
Contratos inteligentes	100 %	100 %	98
API backend	80 %	90.84 %	317
Aplicación frontend	60 %	65.27 %	286

La ejecución automatizada de las pruebas unitarias proporciona una capa de seguridad que facilita la revalidación del comportamiento del sistema, lo cual es relevante en un prototipo que podría expandirse en un futuro. La validación constante de la base de código antes de cualquier despliegue favorece la estabilidad y la calidad del sistema a largo plazo. Una vez que se ha verificado el funcionamiento de cada componente del sistema de forma individual, es posible

continuar con la siguiente etapa de validación, donde se verificará la correcta interacción entre los módulos del sistema.

7.2. Pruebas de integración

Las pruebas de integración se sitúan en la siguiente capa de la pirámide de pruebas. Su objetivo principal consiste en verificar que los distintos componentes y módulos del sistema interactúen correctamente y de forma coherente con sus responsabilidades. A diferencia de las pruebas unitarias, que validan el funcionamiento aislado de una unidad de código, las pruebas de integración evalúan el flujo de datos y las interacciones entre componentes para asegurar que sus interfaces y responsabilidades estén debidamente sincronizadas. Esta etapa de prueba se corresponde con la fase de diseño de componentes del modelo en V, donde se definen las interacciones entre los módulos del sistema.

Para este proyecto, las pruebas de integración se diseñaron para ser automatizadas y se enfocaron en los puntos de interacción más críticos del sistema. El entorno de prueba se configuró para simular un escenario lo más cercano posible a un entorno real, pero aislado, sin hacer uso de funcionalidades simuladas. Por ejemplo, mientras que durante las pruebas unitarias se simularon los datos que debería retornar la *blockchain* para probar la funcionalidad de la API, durante las pruebas de integración se utilizaron datos reales obtenidos de un entorno virtual de la red *blockchain*. De esta manera, se pudo validar que la comunicación entre los módulos funciona correctamente en un contexto más realista.

```
Test Suites: 5 passed, 5 total
Tests:     18 passed, 18 total
Snapshots: 0 total
Time:      119.97 s
Ran all test suites matching /tests\\integration/i.
```

Figura 7.2: Salida de la ejecución de las pruebas de integración del sistema

El proceso de pruebas de integración no detectó fallos en la interacción entre los módulos. Se realizaron pruebas de integración para 18 casos de uso del sistema, los cuales se han documentado en el Apéndice E. En la Figura 7.2 se puede observar la salida de la ejecución de las pruebas de integración, indicando que todas las pruebas se ejecutaron exitosamente. La ejecución de estas pruebas proporciona una capa de verificación adicional del sistema y se recomienda ejecutarlas de forma rutinaria antes de cada despliegue en un entorno productivo para mitigar los riesgos asociados a los cambios en el código que puedan afectar la interacción entre los módulos del sistema. Una vez que se ha validado la interacción de los componentes, la siguiente etapa consiste en verificar que el sistema en su totalidad cumpla con los requerimientos funcionales y no funcionales definidos al comenzar el proyecto.

7.3. Pruebas de sistema

Las pruebas de sistema constituyen la siguiente fase en el ciclo de validación y tienen como propósito verificar que el prototipo, en su conjunto, cumple con los requerimientos funcionales y no funcionales definidos al comenzar el proyecto. Esta etapa se corresponde con el diseño de arquitectura en el modelo en V, y su objetivo es evaluar el comportamiento del sistema de manera integral para asegurar que la arquitectura implementada efectivamente logra cumplir los requerimientos establecidos.

A diferencia de las pruebas unitarias y de integración, que fueron automatizadas, las pruebas de sistema se ejecutaron de manera manual. En primer lugar, se documentó de forma detallada cada caso de prueba a ejecutarse, incluyendo un título, requerimientos asociados, los pasos a seguir desde la perspectiva de un usuario, los datos de entrada (en caso de requerirlos) y los resultados esperados. Posteriormente, se ejecutó de forma manual cada caso de prueba, siguiendo los pasos definidos y registrando los resultados obtenidos. La metodología consistió en interactuar con la interfaz de usuario (*frontend*) del sistema y seguir los pasos detallados en el caso. En estas pruebas, se utilizó el sistema completo desplegado en el entorno de pruebas, sin emplear datos simulados ni probar funcionalidades de forma aislada. Esto permitió una validación precisa del comportamiento del prototipo, desde la perspectiva de un usuario, en un entorno que replicaba la realidad. En la Figura 7.3 se muestra un ejemplo de un caso de prueba documentado para la creación exitosa de un lote de vidrio, incluyendo información como un identificador, nombre, requerimientos asociados, precondiciones, pasos de ejecución, resultado esperado y resultado obtenido luego de la ejecución, entre otros datos. Adicionalmente, en el Apéndice E se encuentran documentados todos los casos de prueba ejecutados durante esta etapa.

ID: PROD-001	Caso de prueba: Crear un lote con datos válidos	
Historia de usuario: RF-006 - Cargar lotes de producción		
Pantallas: /productor	Prioridad: Alta	Precondiciones: Usuario Productor Primario autenticado
Pasos de ejecución:		
1. Ir a Lotes producidos; 2. Tab o clic en botón 'Crear lote'; 3. Tab al campo 'Cantidad de envases'; clic e ingresar 100; 4. Tab al campo 'Peso por envase'; clic e ingresar 500; 5. Tab al desplegable 'Color del envase'; clic y seleccionar 'verde'; 6. Tab al campo 'Espesor'; clic e ingresar 2; 7. Tab al campo 'Tipo/forma del envase'; clic e ingresar 'cuello largo'; 8. Tab al campo 'Nombre del material'; clic e ingresar 'Vidrio'; 9. Tab al campo 'Cantidad de material'; clic e ingresar 100; 10. Tab al campo 'Unidad de medida'; clic e ingresar '%'; 11. Tab al campo 'Fecha de producción'; clic y seleccionar 27-04-2025; 12. Tab al campo 'Localización de producción'; clic e ingresar 'Argentina'; 13. Tab al botón 'Crear'; presionar Enter o hacer clic		
Datos de prueba: quantity=100; bottleType(weight=500; color=verde; thickness=2; shapeType='cuello largo'; originLocation=Argentina; composition=[Vidrio:100%]); createdAt=27-04-2025	Resultado esperado: Se cierra modal; En listado aparece nuevo lote con ID asignado, cantidad 100, color verde y fecha 27/04/2025; Respuesta API POST /productor/batch → 201 "Created" con información del lote	
Resultado obtenido: Resultado esperado	Estado: Exitoso	Comentarios: -

Figura 7.3: Caso de prueba documentado para la creación exitosa de un lote de vidrio

A su vez, también se incluyeron pruebas de requerimientos no funcionales en esta etapa, como son la escalabilidad, el rendimiento y la seguridad del sistema. A pesar de las limitaciones de recursos del entorno de pruebas (debido a que se utilizaron plataformas gratuitas), se llevaron a cabo pruebas controladas para validar que el sistema puede cumplir con estos requerimientos en un entorno real. Los detalles y resultados de estas pruebas se documentaron junto con los casos de prueba funcionales en el Apéndice E.

El proceso de pruebas de sistema incluyó un total de 55 casos de prueba y resultó en la identificación de 7 incidencias, las cuales fueron registradas y rastreadas en la herramienta Jira hasta su resolución. Entre estas incidencias, se encontraron fallas en la validación de formularios, mensajes de error poco claros al subir formularios con datos inválidos e inconsistencias en los datos al reciclar materiales como productor. Estas incidencias afectaban negativamente la experiencia de usuario, pero no se consideró que fueran críticas, debido a que en ningún caso bloqueaban el uso de la plataforma o el acceso a otras funcionalidades. Cada incidencia fue corregida y validada mediante la reejecución de los casos de prueba afectados.

La ejecución de pruebas de sistema permitió identificar y corregir errores, pero también sirve como una base para futuras pruebas de regresión del sistema, ayudando a prevenir que cualquier cambio o nueva funcionalidad no afecte negativamente el comportamiento actual. Luego de esta etapa, el prototipo ya está listo para ser presentado a un grupo reducido de usuarios, quienes serán responsables de evaluar su funcionalidad durante las pruebas de aceptación.

7.4. Pruebas de aceptación con usuarios

La fase final de validación del prototipo corresponde a las Pruebas de Aceptación con Usuarios (UAT, *User Acceptance Testing*). Esta etapa se enfoca en verificar que el sistema no solo funcione correctamente a nivel técnico, sino que también satisfaga las necesidades y expectativas del usuario final, tal como se definieron en la etapa de modelado de requerimientos del modelo en V. En esta instancia, un grupo reducido de usuarios es convocado para utilizar el sistema en un entorno controlado, de forma similar a cómo utilizaría el sistema en su rutina habitual.

Dada la naturaleza de este trabajo como proyecto académico, la obtención de usuarios reales de la industria del vidrio para la ejecución de estas pruebas presentó una limitación. No obstante, se llevó a cabo un experimento controlado con un grupo de usuarios voluntarios del entorno académico. El experimento se realizó en un laboratorio de computación de la Facultad de Ingeniería de la Universidad Nacional de Cuyo, donde cada usuario voluntario contó con una computadora de escritorio individual con acceso a Internet, desde donde pudo acceder al *frontend* del prototipo desplegado en el entorno de pruebas para explorar la plataforma e interactuar con ella como lo haría un usuario real. La metodología de la prueba combinó una fase guiada y una fase libre, creando un entorno mixto, donde los participantes recibieron una guía para ejecutar una serie de casos de prueba predefinidos, seguidos de un período de exploración libre del sistema. Este enfoque permitió validar tanto los flujos de trabajo específicos

como la usabilidad general del prototipo. Los participantes evaluaron tanto los requerimientos funcionales (por ejemplo, la capacidad de registrar un lote de vidrio) como los no funcionales (como la facilidad de uso y la transparencia de la información) registrando en una planilla la descripción del caso de prueba ejecutado y el resultado obtenido.

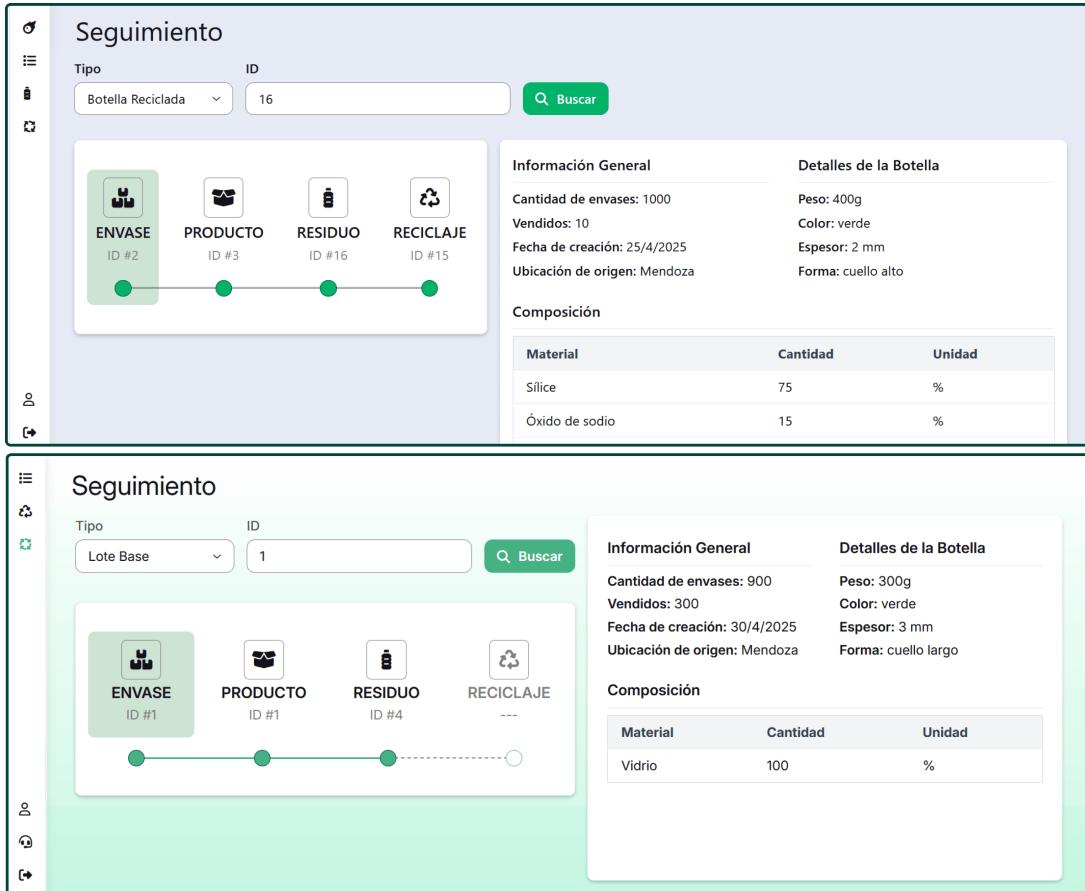


Figura 7.4: Rediseño de la interfaz de usuario de la pantalla de trazabilidad basado en la retroalimentación recibida

La ejecución de las pruebas generó una retroalimentación detallada que se utilizó para identificar y resolver fallos, así como para mejorar la experiencia de usuario (UX, *User Experience*) y el diseño de la interfaz (UI, *User Interface*). Los participantes en su conjunto registraron un total de 63 casos de prueba ejecutados, incluyendo acciones como crear un lote de envases, asociar un código comercial a un lote de producto, reciclar envases y realizar consultas sobre de seguimiento de los mismos. Durante las pruebas ejecutadas, los participantes reportaron 16 casos donde el sistema que no funcionaba como esperaban, incluyendo inconsistencias en ciertos datos, mensajes de error poco claros e inesperados al enviar formularios, falta de validación en formularios y problemas de usabilidad en la funcionalidad de seguimiento. De estas incidencias, se consideró prioritario resolver las que implicaban mensajes de error inesperados o inconsistencias en los datos, ya que afectaban directamente la funcionalidad del sistema. Otras incidencias relacionadas con la experiencia de usuario y el diseño de la interfaz se priorizaron para su resolución en función del impacto percibido por los usuarios y la viabilidad técnica de implementar las mejoras. A partir del relevamiento de los usuarios voluntarios, se registraron en Jira un total de 9 incidencias y 3 sugerencias recibidas para su posterior seguimiento y re-



Figura 7.5: Usuarios voluntarios recibiendo las instrucciones del experimento



Figura 7.6: Usuarios voluntarios interactuando con el prototipo durante prueba guiada

solución, teniendo en cuenta que algunos de los comportamientos reportados se repitieron en más de un caso.

Como resultado directo de las pruebas de aceptación, se realizaron mejoras en los flujos de navegación, la presentación de los datos de trazabilidad y el diseño visual de la aplicación. En la Figura 7.4 se puede observar el rediseño de la interfaz de usuario de la pantalla de trazabilidad basado en la retroalimentación recibida, que incluyó modificar la disposición y proporción de los elementos en pantalla, cambiar el color del fondo para aumentar el contraste y estandarizar

el ancho de cada paso del ciclo de vida del material. Complementariamente, en el Apéndice E se encuentran documentados de forma detallada los casos de prueba ejecutados durante el experimento, el listado de participantes y su retroalimentación. En las Figuras 7.5 y 7.6 se incluyen fotografías tomadas durante la ejecución del experimento, donde se puede observar a los usuarios voluntarios recibiendo las instrucciones del experimento y utilizando el prototipo en las computadoras del laboratorio.

Al finalizar la etapa de pruebas con usuarios, se llevó a cabo una revisión de los resultados obtenidos. En el Apéndice A se puede acceder a un video demostrativo de uso del sistema finalizado luego de su implementación y validación. El riguroso proceso de pruebas llevado a cabo, desde las unidades más pequeñas de código hasta la validación con usuarios voluntarios, demostró que el prototipo desarrollado es funcional y está alineado con los requerimientos originales del proyecto. Esta etapa de validación confirma la viabilidad del sistema de trazabilidad del vidrio basado en *blockchain*, estableciendo un precedente sólido para su potencial escalabilidad y aplicación en un entorno productivo. La finalización de esta fase marca la conclusión del ciclo de vida del desarrollo del software de este proyecto académico, cuyos principales hallazgos, lecciones aprendidas y oportunidades de mejora se resumen en el siguiente capítulo.

8

CONCLUSIONES

En este capítulo de conclusiones, tras haber validado técnicamente el prototipo mediante pruebas rigurosas, se presentan las conclusiones generales del trabajo realizado. Adicionalmente, se reflexiona sobre el proceso de desarrollo del prototipo de trazabilidad de envases de vidrio, analizando los resultados obtenidos, los desafíos superados y las perspectivas futuras que emanan de este trabajo. Se busca ofrecer una visión completa que no solo se limite a lo técnico, sino que también abarque la experiencia metodológica y el potencial de impacto real de la solución.

8.1. Conclusiones del trabajo

En este trabajo se desarrolló un prototipo tecnológico que implementa un sistema de trazabilidad para envases de vidrio en la industria vitivinícola, utilizando tecnología *blockchain*. El objetivo principal fue diseñar e implementar una solución que permita rastrear el ciclo de vida de los envases desde su producción hasta su reciclaje, con el fin de garantizar el cumplimiento normativo, mejorar la eficiencia operativa, fomentar la sostenibilidad y aumentar la confianza entre todos los actores involucrados en el proceso. El resultado final del trabajo fue un prototipo tecnológico funcional con una interfaz de usuario cuidada y una experiencia de usuario validada a través de un riguroso proceso de pruebas. A partir de los resultados obtenidos, se concluye que los objetivos planteados fueron cumplidos satisfactoriamente y que este prototipo representa una prueba de concepto que demuestra la viabilidad del uso de tecnología *blockchain* para impulsar la transparencia y la sostenibilidad en la industria vitivinícola de la región.

Al reflexionar sobre el proceso de ejecución de este trabajo y los resultados obtenidos, se consi-

dera que la elección del modelo en V para guiar el desarrollo del prototipo fue idónea. Esta metodología, que se asocia habitualmente a proyectos de gran escala que demandan alta calidad, demostró ser altamente eficaz en este contexto. La inmutabilidad de la tecnología *blockchain* (y de los contratos inteligentes) exige un enfoque que minimice la aparición de errores en las etapas finales del ciclo de vida del software. En este sentido, las fases de definición de requerimientos y diseño del software previo a su implementación, así como la ejecución de pruebas unitarias desde la implementación, propuestas por el modelo en V fueron valiosas, permitiendo detectar y corregir inconsistencias de diseño e integración antes de la implementación, lo que facilitó un desarrollo tanto ágil como robusto.

A pesar de la rigidez de la metodología, el proceso de ejecución se adaptó a las circunstancias. Aunque la planificación inicial se desvió 4 semanas debido a contratiempos, como los viajes o las cargas académicas, no impidieron el avance continuo del proyecto. De hecho, el viaje de investigación a Europa motivado por una iniciativa profesional, aunque desvió el cronograma, enriqueció de forma significativa el proyecto, proporcionando una perspectiva global sobre la economía circular y la cultura del reciclaje. Esta experiencia personal reforzó la convicción sobre la aplicabilidad del trabajo, destacando que la flexibilidad y la resiliencia son también componentes valiosos en la gestión de proyectos académicos, particularmente en entornos de investigación aplicada con restricciones reales de tiempo y recursos. Luego de la retrospectiva, se concluye que este trabajo cumple con los objetivos planteados y se valora positivamente el resultado final, la ejecución de la metodología y los aprendizajes adquiridos.

8.2. Reflexiones finales

El desarrollo de este trabajo no estuvo exento de desafíos, muchos de los cuales fueron tan importantes como el propio desarrollo del código. El primer gran reto fue ir más allá de la dimensión técnica para comprender las motivaciones, necesidades y limitaciones de los distintos actores involucrados en la cadena de valor. Plantear una solución que pudiera atender las particularidades de cada uno, desde el productor hasta el reciclador, requirió una labor de análisis y conceptualización que sentó las bases para el éxito del prototipo.

Desde el punto de vista técnico, la implementación de tecnología *blockchain* sin experiencia previa representó una curva de aprendizaje considerable. A pesar de su complejidad, se constató que *blockchain* es una tecnología altamente pertinente para resolver problemas asociados a transparencia y descentralización. Los aprendizajes adquiridos a lo largo de este proceso de desarrollo son un activo valioso, y la experiencia con *blockchain* abre la puerta a futuras investigaciones y aplicaciones relacionadas.

A nivel metodológico, la adaptación de un proceso pensado para un equipo a un trabajo individual fue otro desafío. El uso de herramientas de gestión de proyectos como Jira fue una estrategia eficaz para mantener el orden, la visibilidad del progreso y la trazabilidad de las tareas, indicando que la disciplina metodológica contribuye al éxito incluso en proyectos uni-

personales. Por último, la validación del sistema con usuarios en un contexto académico fue un reto que se abordó con creatividad, recurriendo a usuarios voluntarios para simular un entorno de pruebas realista y obtener una retroalimentación valiosa que permitió refinar la interfaz y la experiencia de usuario.

8.3. Perspectivas futuras

Con una perspectiva a futuro, se estima que este trabajo de grado representa una prueba de concepto con un gran potencial de escalabilidad y expansión. En primer lugar, la arquitectura del sistema puede extenderse para incluir la trazabilidad de otros materiales, como el plástico y el aluminio. Además, el prototipo puede servir como el núcleo que potencie una familia de aplicaciones independientes, desarrolladas a la medida de cada actor de la cadena, o que sirvan como incentivo a los ciudadanos, tal como se observó en múltiples proyectos revisados en el estado del arte. A su vez, la apertura del sistema es una perspectiva relevante a futuro, ya que la trazabilidad podría iniciarse en cualquier punto de la cadena de valor, para reducir la barrera de entrada y facilitar la adopción del sistema.

Desde una perspectiva técnica, las mejoras futuras podrían incluir la integración con sensores IoT en las líneas de producción para automatizar la carga de información, minimizando el error humano. A nivel de impacto, la implementación de esta solución podría tener una influencia positiva en la cultura de la sostenibilidad en la región. Como se observó en las investigaciones de campo en Europa, la confianza generada por la trazabilidad y la transparencia puede ser una herramienta para generar conciencia ciudadana. Al proveer a los consumidores de información sobre el ciclo de vida de los productos, se puede impulsar un cambio de comportamiento que beneficie a la economía y ecología locales.

Apéndices



CONTENIDO EXTERNO

A continuación se presentan una serie de recursos adicionales relacionados con el proyecto accesibles en línea.

A.1. Demostración

Como parte de este trabajo se grabó un video demostrativo que resume los aspectos más relevantes del prototipo desarrollado y presenta el funcionamiento de la plataforma desde la perspectiva del usuario. Este video se encuentra disponible en el siguiente enlace: <https://github.com/RocioCM/computer-science-thesis/blob/main/docs/system-demo.mov>

Complementariamente, se documentó el estado final del sistema mediante una serie de capturas de pantalla de la plataforma, comprendiendo las principales funcionalidades de cada pantalla del sistema. Estas capturas pueden consultarse en el siguiente enlace: <https://github.com/RocioCM/computer-science-thesis/blob/main/docs/screenshots>

A su vez, se encuentra disponible en línea la demostración del prototipo, desplegada en un entorno de pruebas. Esta demostración permite a los usuarios explorar las funcionalidades del sistema en un entorno controlado. La demostración está disponible en el siguiente enlace: <https://computer-science-thesis.vercel.app/app>

A.2. Código fuente

El código fuente completo del prototipo se encuentra alojado en un repositorio público de GitHub. El repositorio puede ser consultado en el siguiente enlace: <https://github.com/RocioCM/>

computer-science-thesis/tree/main/code

A.3. Documentación técnica

La documentación técnica del sistema se encuentra disponible en el repositorio de GitHub mencionado anteriormente. Esta documentación comprende los siguientes aspectos:

- Descripción de la arquitectura del sistema y estructura del código.
- Detalles sobre la implementación de los contratos inteligentes.
- Guía de uso de la API (OpenAPI).
- Instrucciones de instalación y despliegue.
- Instrucciones para ejecutar las pruebas automatizadas.
- Instrucciones para configurar el entorno de desarrollo.

A.4. Gestión del proyecto

El reporte de la planificación y ejecución de las tareas del proyecto, incluyendo la gestión de incidencias y el seguimiento del progreso, se encuentra disponible en la herramienta Jira. Se puede consultar un listado resumido de las incidencias registradas en el siguiente enlace: <https://github.com/RocioCM/computer-science-thesis/blob/main/docs/Estado-Jira-Export.pdf>

A su vez, en el siguiente enlace se puede consultar el listado completo de cada una de las tareas e incidencias registradas, incluyendo su descripción completa y metadatos: <https://github.com/RocioCM/computer-science-thesis/blob/main/docs/Tareas-Jira-Export.pdf>



ENTREVISTA A VERALLIA

Para obtener una comprensión profunda de la situación del reciclaje de vidrio y los desafíos que enfrentan los actores de la cadena de suministro en la región, se realizó una entrevista con Lucía J., representante del Área de Medio Ambiente de Verallia, la principal empresa productora de envases de vidrio en la provincia de Mendoza. La entrevista se realizó de forma telefónica, fue semiestructurada y tuvo una duración aproximada de 30 minutos.

La conversación se centró en conocer los procesos actuales de producción y reciclaje de vidrio en la empresa, las iniciativas sostenibles que están implementando y las oportunidades para mejorar la trazabilidad y la gestión de residuos en su cadena de suministro. Para guiar la conversación, se preparó una lista de preguntas que abarcaban temas como el proceso que realizan, el origen y la clasificación del vidrio reciclado, el rol de sus clientes y proveedores, las reglamentaciones locales y el potencial de un sistema de trazabilidad como el propuesto en este trabajo. Durante la conversación, se permitió una fluidez natural y se profundizó en temas relevantes a medida que la entrevista avanzaba. A continuación, se presenta la transcripción literal de la entrevista, seguida de un análisis y reflexión sobre los puntos más relevantes de la misma.

B.1. Transcripción de la entrevista

Entrevistadora: Primero te pongo un poco en contexto sobre lo que estoy haciendo y después te pido que me cuentes lo que puedas. Como te adelanté, estudio computación. Para mi trabajo final estoy planteando un sistema para fomentar el reciclaje, especialmente de vidrio, porque encontré que hay muchos sistemas orientados al plástico o residuos electrónicos, pero no tanto al vidrio. Por eso elegí este material para mi trabajo. Por ejemplo, trabajé en una aplicación

con la municipalidad de Godoy Cruz, llamada Greeny Points, que incentiva a los usuarios a reciclar. Las personas dejan botellas de plástico o latas en una máquina inteligente, reciben un QR que escanean con el celular y suman puntos, que luego pueden canjear por beneficios como carga en la tarjeta Sube o entradas al cine. Es un sistema que ya probamos y funciona muy bien. Podría ir por algo similar, pero primero necesito entender cómo funciona todo el ecosistema del vidrio. Contame un poco cómo es el proceso que hacen ustedes.

Lucía: Sí, el sistema de reciclado de vidrio. Hoy funciona así: por un lado, hay empresas pequeñas o PyMEs¹ que recolectan vidrio de la calle, junto con recolectores informales. En algunos municipios hay recolectores en blanco. El problema principal en Mendoza, y en Argentina en general, es que no hay una cultura de reciclaje.

Entrevistadora: ¿Qué significa esto?

Lucía: Por ejemplo, en algunos municipios te retiran los residuos reciclables ciertos días. Yo vivo en Capital y sé que los martes y jueves hay que sacar papel, cartón, vidrio, todo separado. Pero luego, en el camión, lo mezclan y en la planta recicladora lo separan. En otros países, la gente separa incluso por tipo de vidrio (blanco, verde), porque el sistema lo fomenta. Acá no hay una política general en Mendoza; cada municipio lo hace a su criterio. Tuvimos una reunión con el Ministerio de Medio Ambiente y el director de la fábrica, y hablamos de esto. El problema es que quienes recolectan vidrio informalmente y nos lo venden no tienen la documentación legal habilitada. Como empresa multinacional, necesitamos documentos de AFIP y otros requisitos legales para comprarles, y si no los tienen, no podemos hacerlo. Ellos terminan vendiendo el vidrio en otro lado. Nosotros compramos a quienes sí tienen la documentación, generalmente PyMEs o empresas chicas. Ese vidrio se limpia porque viene con tapitas, etiquetas, y no todo tipo de vidrio se puede reciclar. Por ejemplo, no podemos reciclar el vidrio marrón de botellas de cerveza, solo blanco y verde. Hay categorías según el diámetro y tipo de envase. Compramos el vidrio, lo almacenamos y lo enviamos a dos empresas que lo limpian y clasifican manualmente, lo que encarece el proceso. Estas empresas siguen ciertos procesos de calidad. Desde abril tenemos una planta de limpieza propia para intentar reciclar más y tener vidrio limpio. El vidrio limpio se mezcla en el horno con la materia prima y se funde para fabricar nuevos envases.

A nivel de proveedores, la empresa tiene el programa “Vidrio, una acción transparente”, que lleva adelante Alejandra Merín, responsable de Responsabilidad Social Empresarial. Este programa realiza acciones externas, como campañas en supermercados y barrios privados, donde hay campanas para que la gente lleve sus botellas. Actualmente tenemos alrededor de 40 campanas ubicadas en diferentes puntos de Capital y Guaymallén. Nosotros las retiramos y las procesamos. Se hacen campañas como el Día del Niño, donde si llevás botellas podés canjearlas por entradas al cine, o sorteos de canastas de mercadería en el Día del Medio Ambiente. Son formas de incentivar el reciclaje, pero el volumen que se recicla por estas campañas es mínimo comparado con lo que necesitamos. Es una forma de fomentar el reciclaje porque no hay

¹ PyMEs: Pequeñas y Medianas Empresas

políticas provinciales o municipales que acompañen estas acciones. Esa sería la problemática, que no hay políticas que acompañen las iniciativas de reciclaje.

Entrevistadora: Entonces, ¿la mayoría del vidrio que reciben para reciclar proviene de recicladores y solo una mínima parte de las campañas?

Lucía: Exacto, la mayoría llega de pequeñas empresas. También recibimos algo de bodegas, que traen descartes, pero es muy poco. El problema es que comprar vidrio reciclado es caro, más que la materia prima virgen, por la limpieza manual. Por eso tuvimos que bajar los objetivos de reciclaje este año. Si los recolectores informales pudieran vendernos directamente, sería más barato, pero como no tienen la documentación, hay intermediarios y el precio sube. Además, el vidrio tiene que clasificarse bien y a veces se descarta por alto contenido de plomo, porque las botellas de algunas empresas de la competencia que no exportan tienen más plomo. Entonces eso limita la cantidad de vidrio que podemos reciclar porque nosotros exportamos y tenemos que cumplir ciertas normas internacionales, también las bodegas, que son nuestros clientes.

Entrevistadora: ¿Ustedes se encargan de clasificar el vidrio o ya viene clasificado?

Lucía: La clasificación se hace en nuestra planta de limpieza y en las dos empresas externas que limpian el vidrio.

Entrevistadora: ¿Cómo lo clasifican? ¿En qué categorías?

Lucía: Principalmente por color, aunque hay otros parámetros de calidad que desconozco. Separan tapitas, etiquetas y basura, pero detalles específicos los maneja el encargado de la planta.

Entrevistadora: ¿Saben cuánto del vidrio reciclado proviene de sus propios envases y cuánto de otras empresas?

Lucía: No, porque llega vidrio de cualquier lado. Hay competencia en la región y no siempre sabemos el origen exacto de cada botella. Hay otras empresas productoras de envases en Buenos Aires, Santa Fe y San Juan de las que también llegan envases para reciclar, entonces no sabemos de dónde viene cada botella.

Entrevistadora: ¿Considerás que podría ayudar a aliviar el proceso de clasificación implementar una máquina que preclasifique por color o características?

Lucía: Sí, podría ser útil. Si averiguo más sobre la tecnología de clasificación, te voy a avisar.

Entrevistadora: A pesar de haber disminuido la meta anual de reciclaje debido a los costos, ¿Sus proveedores pudieron cumplir con la cantidad de vidrio que necesitaban para reciclar?

Lucía: Sí, hay oferta suficiente de vidrio para reciclar.

Entrevistadora: ¿La restricción de no comprar a recicladores informales es por política de la empresa o legislación local?

Lucía: Es por ambas razones: necesitamos papeles legales y cumplir requisitos impositivos, tanto por la empresa como por la legislación local.

Entrevistadora: Pienso en una aplicación para centralizar la venta de vidrio reciclado mediante una sola empresa y reducir intermediarios y así reducir el precio de adquisición del material. ¿Creés que sería posible?

Lucía: No creo, este proveedor centralizado al mismo precio no lo vendería, aunque la aplicación podría servir para los recicladores informales. Se podría usar una aplicación para que la gente lleve vidrio a puntos verdes, como los que ya hay en plazas de Ciudad y Godoy Cruz. Eso podría aumentar la cantidad de vidrio recolectado. Tenemos muchas campanas distribuidas y la gente puede llevar sus botellas ahí. Para mí, el ciudadano tiene que empezar a tener ese compromiso de separar y llevar el vidrio.

Entrevistadora: También pienso que las bodegas podrían hacer campañas para que los consumidores devuelvan sus botellas a la misma bodega. ¿Creés que podría servir?

Lucía: Sí, sería útil y vendría clasificado. Se podría hacer un convenio para que las bodegas devuelvan botellas propias.

Entrevistadora: ¿Las botellas tienen algún tipo de código para identificarlas?

Lucía: Sí, tienen un cuadradito con una V en la base, antes era un trébol. Algunas dicen "eco" y son más livianas. También tienen un número de molde y puntitos que identifican el modelo.

Entrevistadora: ¿Se podría usar esa información para clasificar automáticamente?

Lucía: Sí, las máquinas podrían identificar esos códigos y ayudar en la clasificación, aunque la limpieza manual siempre será necesaria, que sigue siendo un costo alto.

Entrevistadora: ¿Creés que se podría mejorar el reciclaje asociando lo que venden con lo que reciclan?

Lucía: Sí, podría ayudar a saber cuánto vuelve y mejorar la calidad del reciclado. También podría servir para pagar mejor por el vidrio propio o para hacer los convenios con las bodegas.

Entrevistadora: Por último, ¿la empresa estaría dispuesta a participar en algún experimento o proyecto piloto en el futuro?

Lucía: Sí, todo se puede plantear. Habría que consultarla a la hora de la gestión, pero sí, todo lo que sume se puede plantear y gestionar.

Entrevistadora: Perfecto, muchas gracias por tu tiempo y toda la información, sirve mucho para mi trabajo.

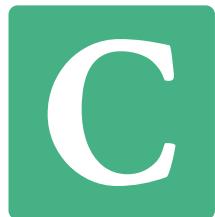
Lucía: De nada, cualquier otra duda me escribís. ¡Suerte!

B.2. Análisis y reflexión

La entrevista realizada proporcionó información valiosa sobre la situación actual del reciclaje de vidrio en Mendoza y los desafíos que enfrenta Verallia en su cadena de suministro. A lo largo de la conversación, se identificaron varias oportunidades de mejora para implementar un sistema que beneficie a todos los actores involucrados. A continuación, se destacan los puntos más relevantes tratados durante la entrevista:

- **Falta de cultura de reciclaje:** La ausencia de una cultura sólida de reciclaje en Argentina, y específicamente en Mendoza, es un obstáculo significativo. La separación de residuos se limita a iniciativas municipales específicas, y la gente no tiene el hábito de clasificar por tipo de vidrio (blanco, verde, etc.). La falta de políticas municipales coherentes y la dependencia de recolectores informales dificultan la recolección eficiente de vidrio reciclable.
- **Clasificación y limpieza del vidrio:** El vidrio que llega a la planta de reciclaje de Verallia no está clasificado y requiere una limpieza manual, lo que encarece significativamente el proceso. La empresa no puede reciclar cualquier tipo de vidrio (por ejemplo, el vidrio marrón de algunas botellas de cerveza) y se ha visto obligada a bajar sus objetivos de reciclaje por los altos costos operativos, ya que el vidrio reciclado termina representando un costo mayor al vidrio virgen. También hay un problema con el descarte de vidrio que contiene altos niveles de plomo, lo que limita la exportación de envases reciclados. La implementación de tecnologías que faciliten la preclasificación del vidrio podría aliviar la carga de trabajo en las plantas de limpieza y reducir costos.
- **Restricciones legales:** Gran parte del vidrio que llega a Verallia para reciclaje proviene de recolectores informales o pequeñas PyMEs. Lucía menciona que, si bien hay oferta, el proceso está encarecido por la falta de formalidad de estos recolectores, que no cuentan con la documentación legal necesaria para vender directamente a una empresa multinacional como Verallia. Esto obliga a la empresa a comprar a través de intermediarios, lo que aumenta los costos. Un sistema que centralice a los recicladores y garantice la legalidad podría mejorar la eficiencia y reducir costos, al minimizar la dependencia de intermediarios.
- **Iniciativas de concientización:** Las campañas internas y externas para fomentar el reciclaje son valiosas para lograr una mayor concientización ciudadana, pero su impacto en volumen de material reciclado es limitado. Un enfoque más amplio que involucre a clientes y consumidores finales podría aumentar la cantidad de vidrio reciclado.
- **Trazabilidad y tecnología:** La posibilidad de implementar un sistema de trazabilidad utilizando códigos o marcas en las botellas presenta una oportunidad para mejorar la gestión del reciclaje. Esto podría facilitar la identificación y clasificación del vidrio, beneficiando tanto a Verallia como a los recicladores.
- **Colaboración con clientes:** Involucrar a los clientes, como bodegas, en programas de devolución y reciclaje podría aumentar la cantidad de vidrio recuperado y mejorar la

calidad del material reciclado. Lucía reconoce que un sistema de trazabilidad que asocie lo que Verallia vende a las bodegas con lo que después vuelve para reciclar podría ser muy útil. También sugiere que la empresa podría estar dispuesta a participar en un experimento o proyecto piloto si se presenta de forma estructurada.



VIAJE DE INVESTIGACIÓN

Durante la ejecución de este proyecto, el proceso de investigación fue complementado por una experiencia de campo que, aunque ajena a la planificación inicial, resultó ser enriquecedora para el trabajo. En noviembre de 2024, debido a un proyecto laboral independiente de este trabajo, se realizó un viaje de investigación a Europa con el objetivo de estudiar de primera mano los sistemas de reciclaje y los modelos de economía circular implementados en diversas ciudades. Esta oportunidad permitió obtener una perspectiva global que, si bien generó un desvío en el cronograma, aportó un conocimiento valioso y una visión práctica de los desafíos abordados en este trabajo.

La experiencia, de dos semanas de duración, incluyó visitas a centros de reciclaje, interacciones con organizaciones dedicadas a la sostenibilidad y el estudio de redes de comercio circular en tres ciudades: Madrid (España), Ámsterdam (Países Bajos) y Berlín (Alemania). Cada una ofreció un panorama distinto de los esfuerzos por promover la economía circular. La primera ciudad visitada fue Madrid, donde se observó la mecánica de la recolección diferenciada y las categorías de residuos en origen. A pesar de que la sociedad española manifiesta un creciente interés por la sostenibilidad, se constató que la implementación de sistemas de reciclaje y economía circular se encuentra en una etapa de desarrollo inicial, en comparación con las ciudades visitadas posteriormente. Se analizó el rol de Ecoembes¹, una organización sin fines de lucro encargada de la gestión de los envases domésticos, cuyo modelo de financiamiento a través de las empresas envasadoras permite costear el proceso de reciclaje. A pesar de su importancia, existen controversias sobre las cifras de reciclaje que reportan y destino final del plástico que gestionan, un debate que pone de manifiesto la complejidad de la medición y regulación en estos sistemas².

¹ <https://www.ecoembes.com/es>

² Fuente: <https://es.greenpeace.org/es/sala-de-prensa/comunicados/ecoembes-miente/>

El viaje continuó en Ámsterdam, una ciudad reconocida por su alto nivel de conciencia ambiental y sus esfuerzos por fomentar hábitos de vida sostenibles. Al aterrizar en el aeropuerto de la ciudad se pudo ver un gran parque de energía eólica desde las alturas, una muestra inicial del compromiso de la ciudad con las energías limpias. Se observó el uso extendido de bicicletas y un sistema de transporte público mayoritariamente eléctrico que contribuyen a reducir la huella de carbono de sus habitantes. En la Figura C.1.A, se puede observar un estacionamiento de bicicletas repleto en la ciudad de Ámsterdam, un escenario frecuente en toda la ciudad. En lo que respecta al reciclaje de envases, Ámsterdam implementa un DRS, en el que los consumidores pagan un depósito por botellas y latas, el cual es reembolsado al devolver los envases en Máquinas Expendedoras Inversas (RVM, *Reverse Vending Machine*) ubicadas en los supermercados. Este sistema ha demostrado ser efectivo para aumentar las tasas de reciclaje de envases. Sin embargo, se identificó un problema de higiene pública, donde los contenedores de basura en la vía pública son vandalizados por recolectores informales que buscan los envases desecharados para obtener el depósito al devolverlos. En la Figura C.1.B, se puede observar una fotografía capturada en la vía pública de un contenedor de residuos vandalizado en este contexto. A pesar de esta situación, el sistema ha logrado fomentar el hábito de reciclaje en los ciudadanos. En esta ciudad se visitó un centro verde, donde se constató una alta tasa de recuperación y reciclaje que no se limita a los envases, sino que se extiende a otros residuos como muebles, ropa y madera, entre otros. La separación de residuos es una práctica normal en la rutina de los ciudadanos, que acuden a estos centros para entregar sus residuos y elementos en desuso. Sin embargo, en el servicio de recolección diferenciada semanal, los residentes reclaman que la frecuencia es insuficiente, lo que lleva a que los contenedores se desborden con frecuencia. Por otro lado, al indagar en el destino final de los materiales reciclables, se informó que la mayoría son procesados en el país, pero se desconoce el destino final de aquellos que son exportados fuera de Europa. Además, aunque se encontraron numerosas tiendas de productos sostenibles, se admitió que la trazabilidad de los materiales reciclados es limitada, lo que sugiere que la separación de residuos y la venta de productos reciclados son flujos independientes y sin conexión directa.

La última parada del viaje fue Berlín, donde se encontró la cultura de sostenibilidad más avanzada entre las ciudades visitadas. La experiencia comenzó con una reunión con la organización Circular Berlin³, una iniciativa que promueve la economía circular en la región a través de la colaboración entre múltiples actores. Su modelo opera como un punto de encuentro que conecta a emprendimientos, empresas, investigadores, ciudadanos y el gobierno local. El objetivo central de la organización es crear una red de actores para compartir conocimientos, debatir ideas e impulsar proyectos de economía circular, con el fin de catalizar un cambio a nivel comunitario y generar un impacto positivo a nivel cultural, económico y ecológico. En este encuentro, un miembro de la organización presentó su misión y destacó proyectos de circularidad que se estaban desarrollando en Berlín, lo que sirvió como una guía estratégica para organizar las visitas a los diferentes puntos de interés durante la estadía. Este encuentro fue un punto de inflexión en el viaje, proporcionando una hoja de ruta para visitar proyectos específicos y com-

³ <https://circular.berlin/>

prender la visión integral de la economía circular en la región, mientras que se exploró cómo la colaboración entre diversos actores puede ser un motor de cambio.

En la capital alemana, se observó un sistema DRS, denominado Pfand [17], que posee similitudes y diferencias con el de Ámsterdam. En Berlín, los consumidores también pagan un depósito por las botellas y latas, el cual es reembolsado al devolver los envases en máquinas RVM ubicadas en supermercados. En la Figura C.2.C se puede observar una fotografía de una máquina RVM ubicada en el exterior de un supermercado en la ciudad de Berlín. A su vez, en esta ciudad se realizó el experimento de devolver envases en una máquina RVM, lo que permitió observar de primera mano la experiencia del usuario del sistema Pfand. La Figura C.2.A muestra una fotografía de la pantalla de una máquina RVM en la que se indica un caso de uso inválido en idioma alemán, donde se explica al usuario que la máquina no acepta envases que contienen líquido dentro, mientras que el mensaje de la fotografía C.2.B indica que el envase no es aceptado por estar dañado de modo que la máquina no puede reconocerlo, y la Figura C.2.C muestra un mensaje que indica que el envase no es aceptado porque no pertenece al sistema Pfand y debe desecharse en un contenedor común por tratarse de un envase importado de otro país o de una marca no registrada en el sistema. Estas situaciones evidencian las limitaciones del sistema actual, donde ciertos envases no son aceptados por las máquinas, lo que puede generar frustración en los usuarios y reducir la efectividad del sistema DRS.

En el caso de Berlín, a diferencia de Ámsterdam, la conciencia de los ciudadanos ha evolucionado de manera tal que, al desechar envases en la vía pública, prefieren dejarlos junto a los cestos de basura para facilitar que las personas sin hogar los recuperen y obtengan el depósito para adquirir alimentos diariamente con estos fondos. Esta práctica demuestra cómo una iniciativa de reciclaje puede también promover un comportamiento social positivo, evitando los problemas de higiene que se observaron en los Países Bajos. Se visitó un centro verde, similar al de Ámsterdam, donde se reciben y clasifican los residuos. El gran flujo de ciudadanos que acuden a estos centros demuestra el alto compromiso de la población con la reducción y la separación de residuos. A su vez, los trabajadores indicaron que esta es una práctica habitual, ya que los ciudadanos pueden ser multados por dejar grandes volúmenes de residuos en la vía pública. Se identificó, además, una gran presencia de tiendas de segunda mano y comercios circulares que promueven la reutilización y la reducción de residuos. Incluso se visitó una tienda de productos a granel, donde los clientes llevan sus propios envases reutilizables para comprar alimentos y productos de limpieza. A partir de lo observado, se pudo constatar que la economía circular en Berlín está profundamente integrada en la rutina diaria de sus habitantes.

En retrospectiva, el viaje de investigación a Europa resultó ser una experiencia valiosa. Cada ciudad visitada proporcionó una visión única de los desafíos y éxitos de los sistemas de reciclaje y economía circular. Se puede inferir que el sistema DRS es un método que ha demostrado ser efectivo para fomentar el reciclaje de envases, logrando tasas de recuperación superiores al 90 %, un nivel que no se ha alcanzado con otros enfoques [35]. A su vez, se pudo constatar que la conciencia ciudadana es un factor que influye en el éxito de cualquier sistema de reciclaje. Los modelos de colaboración y fomento de proyectos, como los observados en Berlín, son un

vehículo eficaz para impulsar la economía circular a nivel regional. El conocimiento obtenido en este viaje apoya la hipótesis de que una solución de trazabilidad del vidrio puede contribuir no solo a la transparencia de la cadena de valor, sino también a generar conciencia y un impacto positivo en la cultura local.



Figura C.1: Imágenes capturadas durante el viaje de investigación a Europa

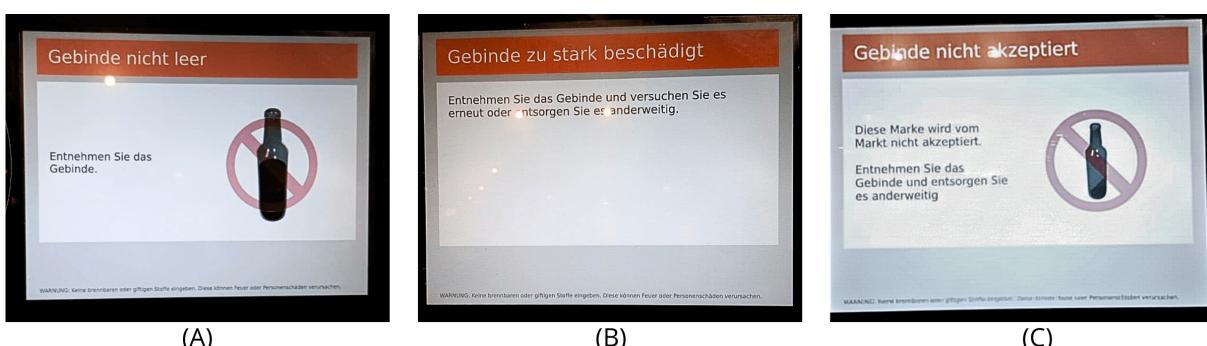


Figura C.2: Mensajes asociados a casos de uso inválidos del sistema de depósito (DRS) en Alemania



FLUJOS DE USUARIO

En este trabajo, se definieron y documentaron los flujos de usuario para cada tipo de usuario identificado en el sistema durante la etapa de diseño de solución. Estos flujos de usuario permiten comprender cómo interactúan los diferentes actores con la plataforma con el objetivo de garantizar una experiencia de usuario intuitiva y eficiente antes de la ejecución de las pruebas de aceptación. A continuación, se describen los flujos de usuario para cada rol identificado: productor primario, productor secundario, consumidor y reciclador. Estos flujos fueron construidos a partir de los requerimientos funcionales definidos en la etapa de modelado y representan las principales interacciones de los distintos actores del sistema con la plataforma.

Inicialmente, todos los usuarios deben registrarse en la plataforma proporcionando información básica como correo electrónico y contraseña (Figura D.1). Al momento del registro, se debe elegir un rol para el usuario y, una vez creado, el usuario puede iniciar sesión para acceder a las funcionalidades correspondientes a su rol (Figura D.2). Dentro de la plataforma, cada usuario puede acceder a la información de su perfil, pudiendo agregar el nombre de su empresa, su nombre y número de contacto (Figura D.3). Adicionalmente, también disponen de la opción para cerrar sesión en todo momento desde el menú lateral de navegación.

Para los productores primarios (Figura D.4), el flujo de uso incluye la posibilidad de registrar nuevos lotes de producción de envases, así como consultar el estado de los lotes existentes y consultar su trazabilidad. A su vez, estos usuarios pueden ver el inventario de material reciclable adquirido por la empresa para su reutilización en la producción.

En el caso de los productores secundarios (Figura D.5), el flujo de uso se centra en la gestión de los envases adquiridos. Estos usuarios pueden asignar sus envases a códigos de productos específicos, posibilitando así su trazabilidad posterior. A su vez, pueden consultar el estado de los envases adquiridos y su trazabilidad. Por último, pueden registrar sus ventas y controlar

el stock de envases disponibles.

Los consumidores (Figura D.6), por su parte, pueden consultar información del origen de cualquier envase a partir de su código de producto. Pueden acceder a información detallada sobre los productos que han adquirido, incluyendo detalles sobre su trazabilidad y el ciclo de vida de los envases. Además, pueden reportar la entrega de envases a recicladores, pudiendo luego consultar el estado de cada uno de los envases reportados.

Finalmente, los recicladores pueden consultar el inventario de envases entregados por los consumidores (Figura D.7), para clasificarlos y asignarlos a lotes de reciclaje según su composición. Estos usuarios pueden crear lotes de reciclaje y registrar la venta de material reciclado a productores primarios (Figura D.8), cerrando así el ciclo de trazabilidad. A su vez, pueden consultar el origen de un envase recibido para reciclarlo correctamente.

A su vez, todos los usuarios pueden acceder a la funcionalidad de Seguimiento (Figura D.9), donde pueden consultar el estado de cualquier envase registrado en la plataforma a partir de su código de producto o identificador. Esta funcionalidad permite a todos los actores del sistema verificar la trazabilidad y el ciclo de vida de los envases, promoviendo la transparencia y la confianza en el sistema.

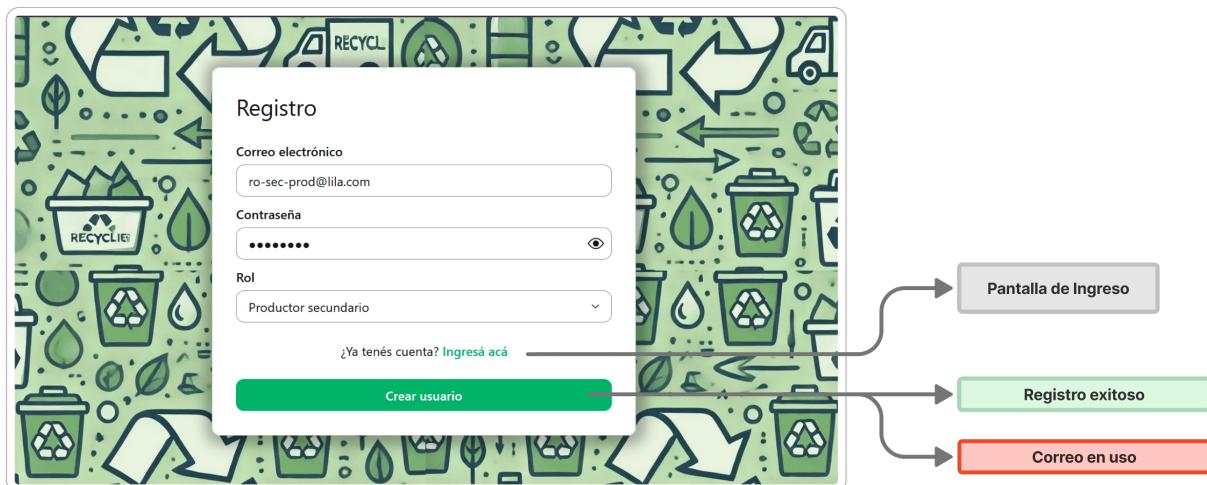


Figura D.1: Flujo de registro de usuarios

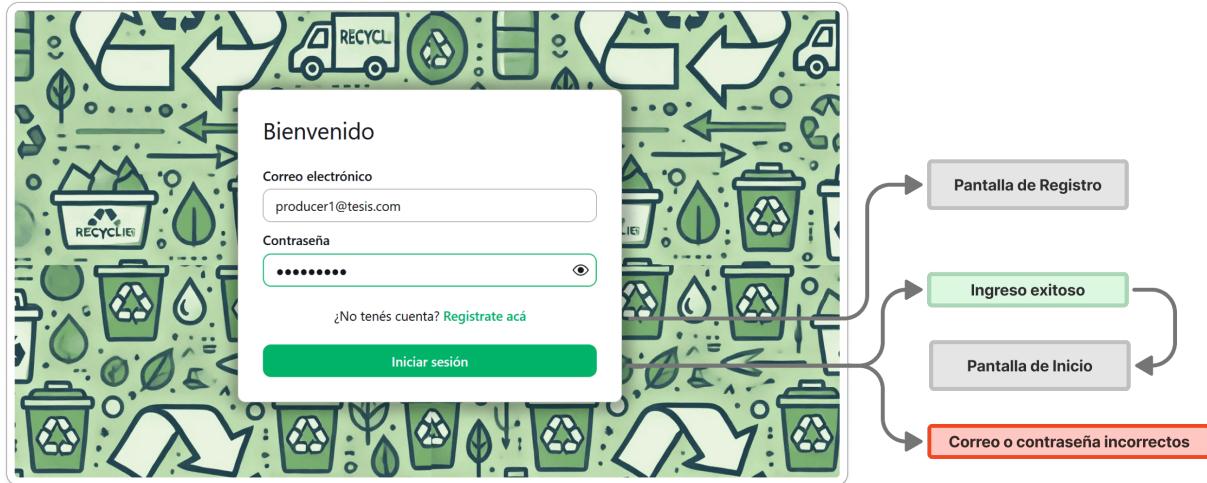


Figura D.2: Flujo de autenticación de usuarios

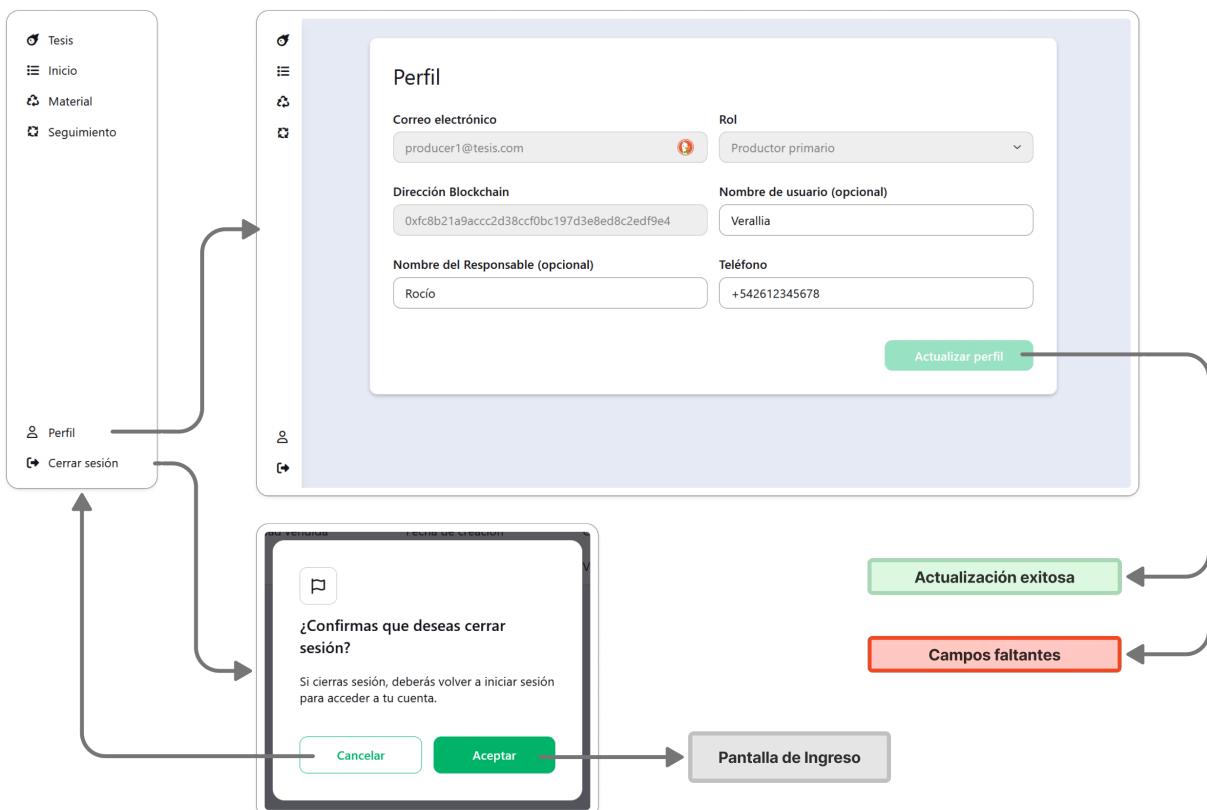


Figura D.3: Flujo de gestión de usuarios

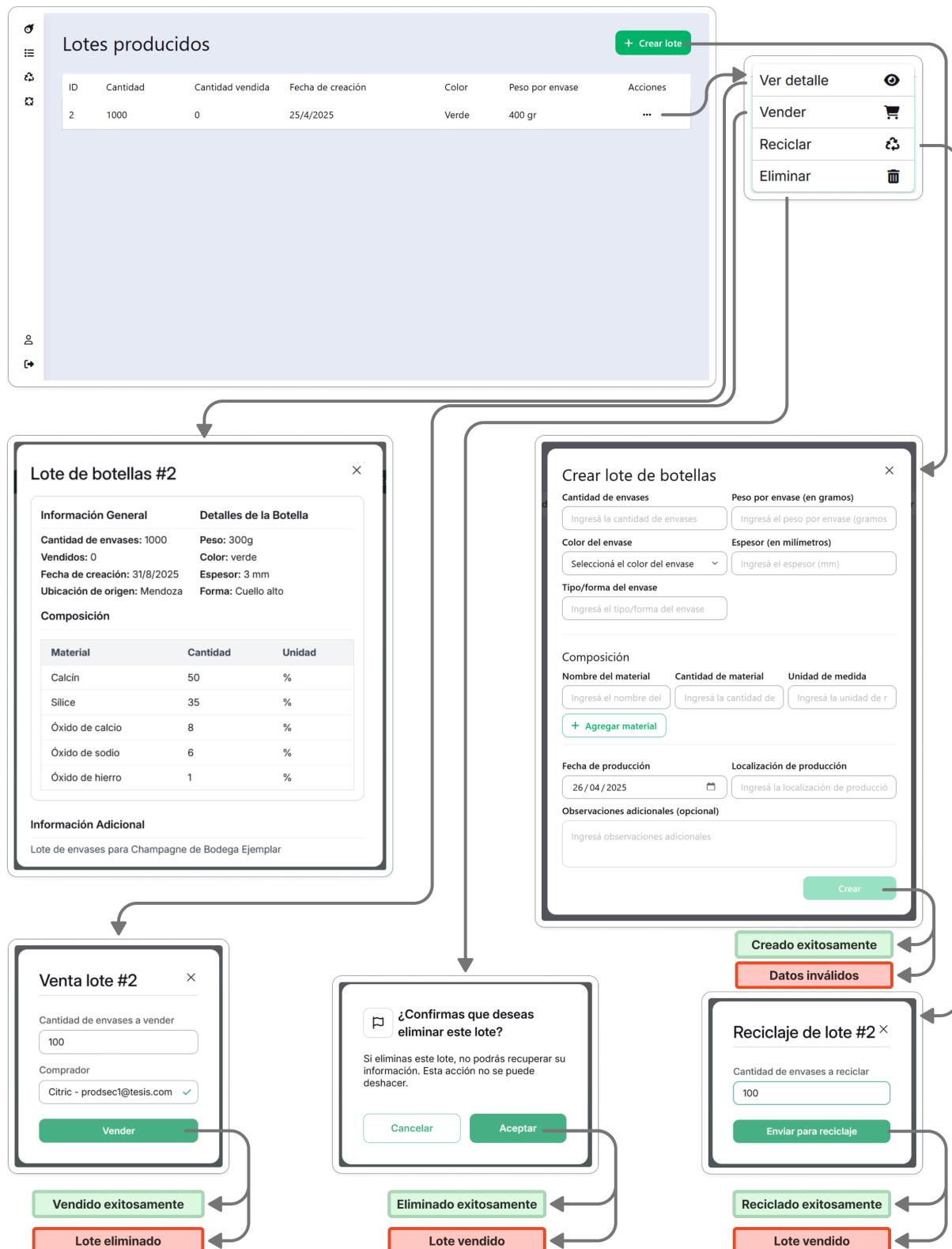


Figura D.4: Flujo de usuario de productor primario

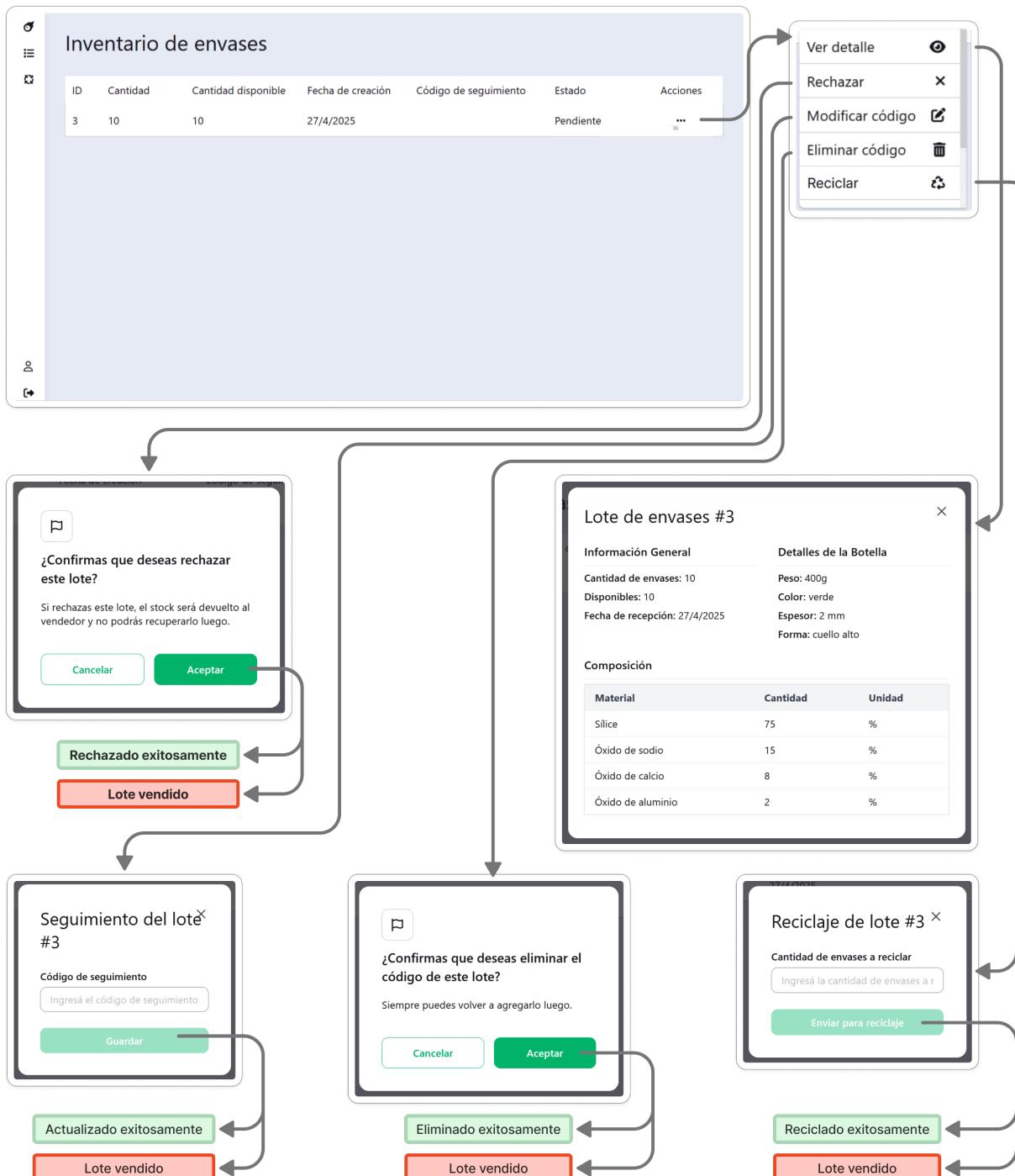


Figura D.5: Flujo de usuario de productor secundario

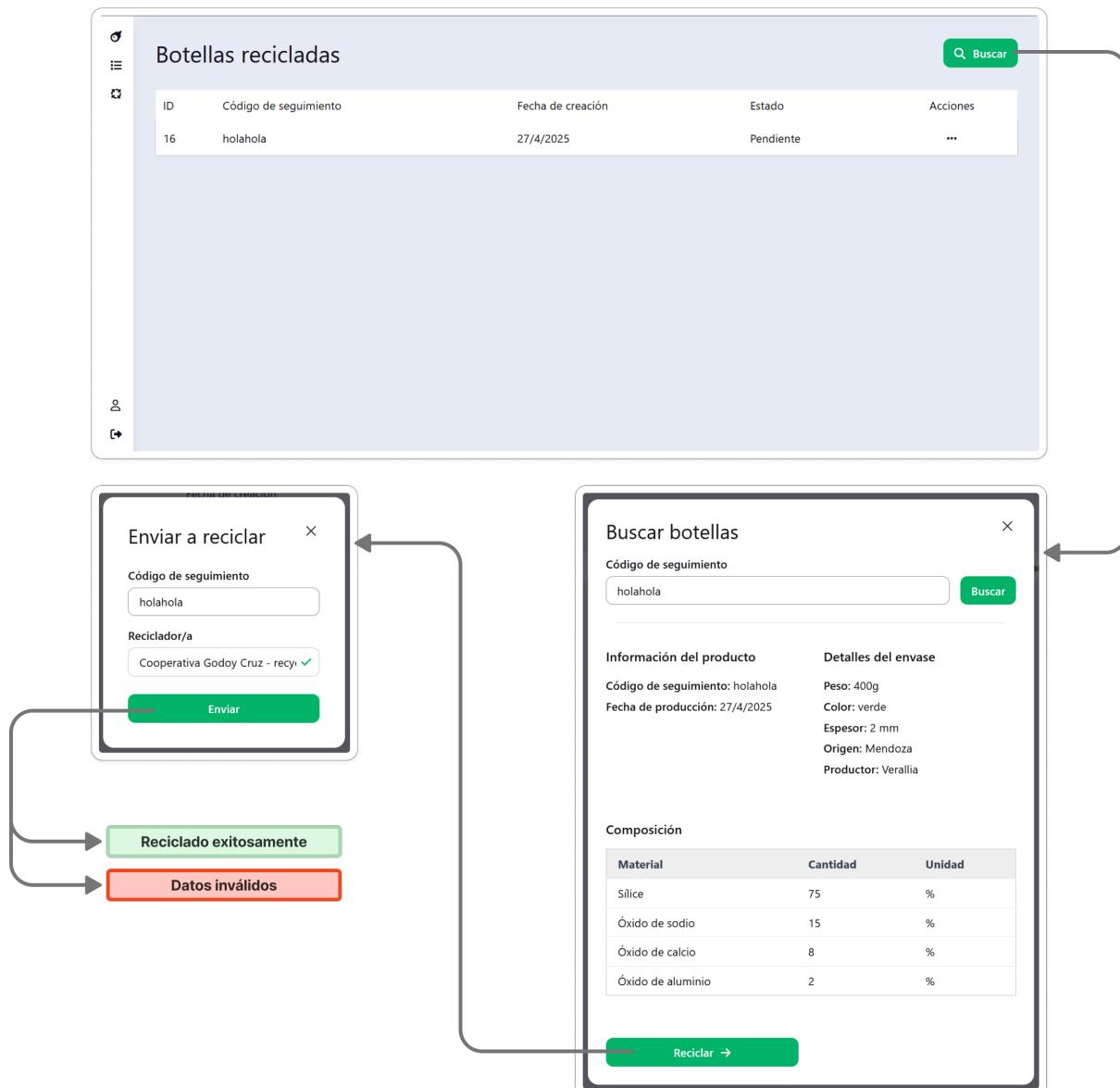


Figura D.6: Flujo de usuario de consumidor

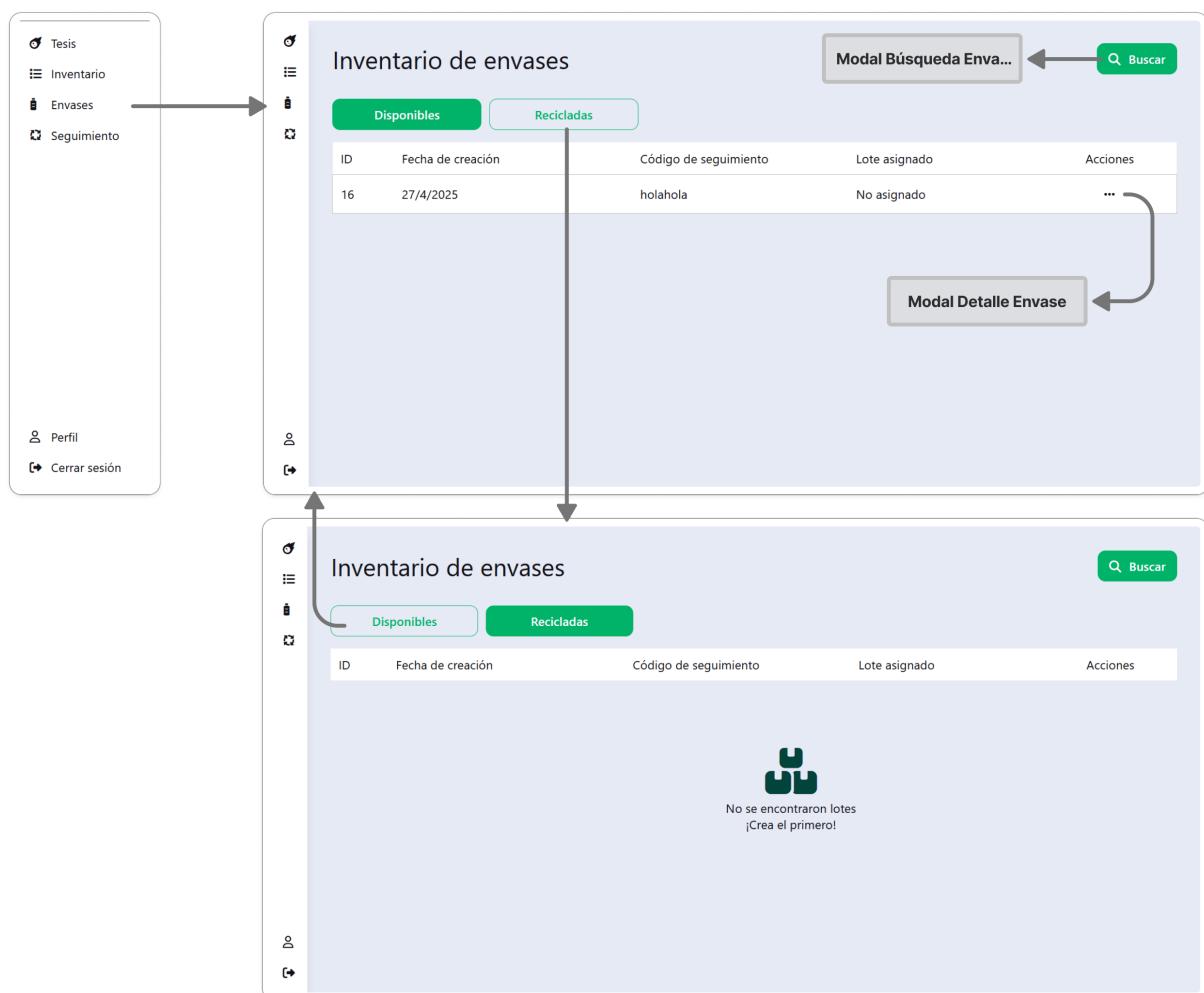


Figura D.7: Flujo de usuario de reciclador para gestión de envases

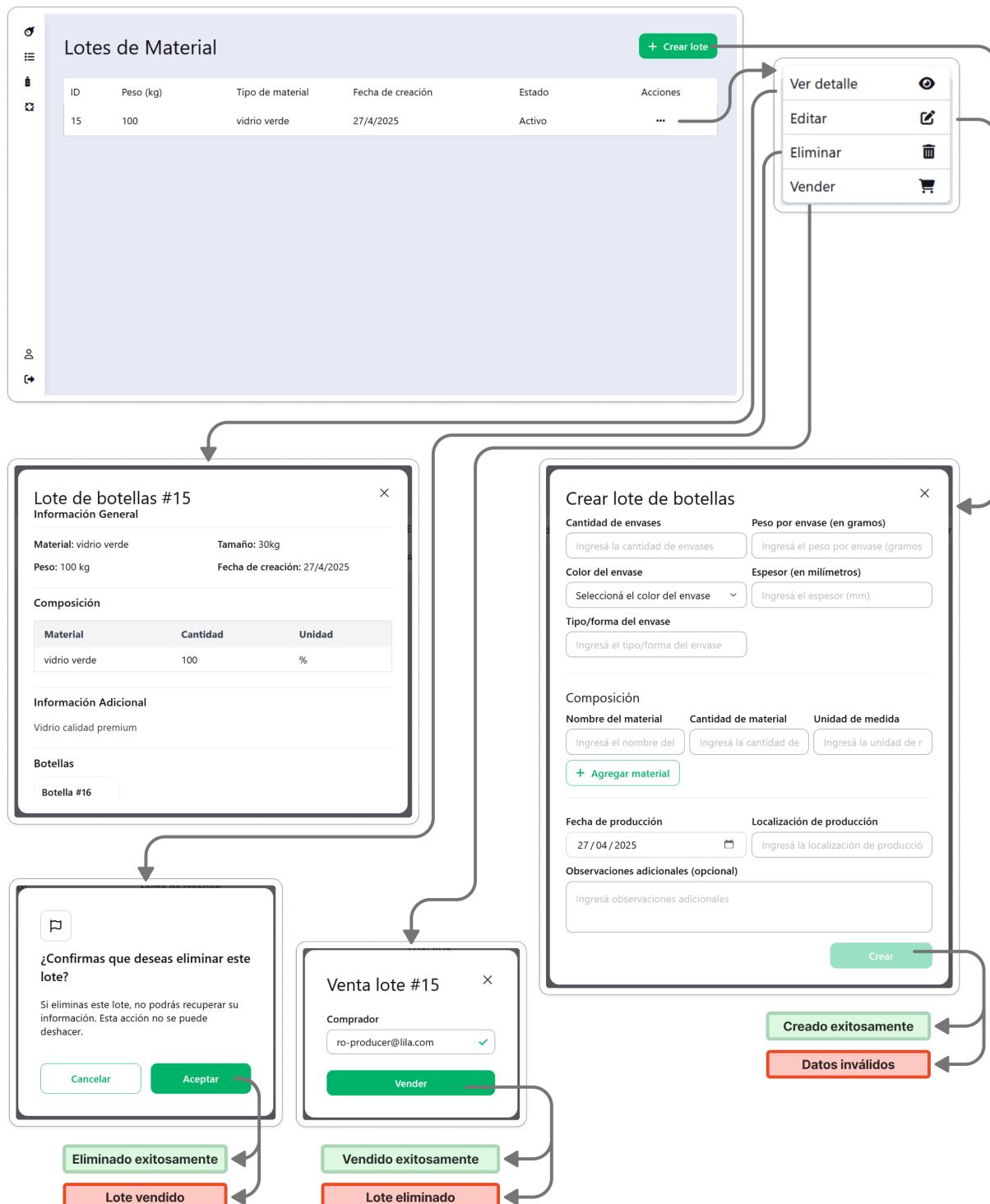


Figura D.8: Flujo de usuario de reciclador para gestión de lotes de reciclaje

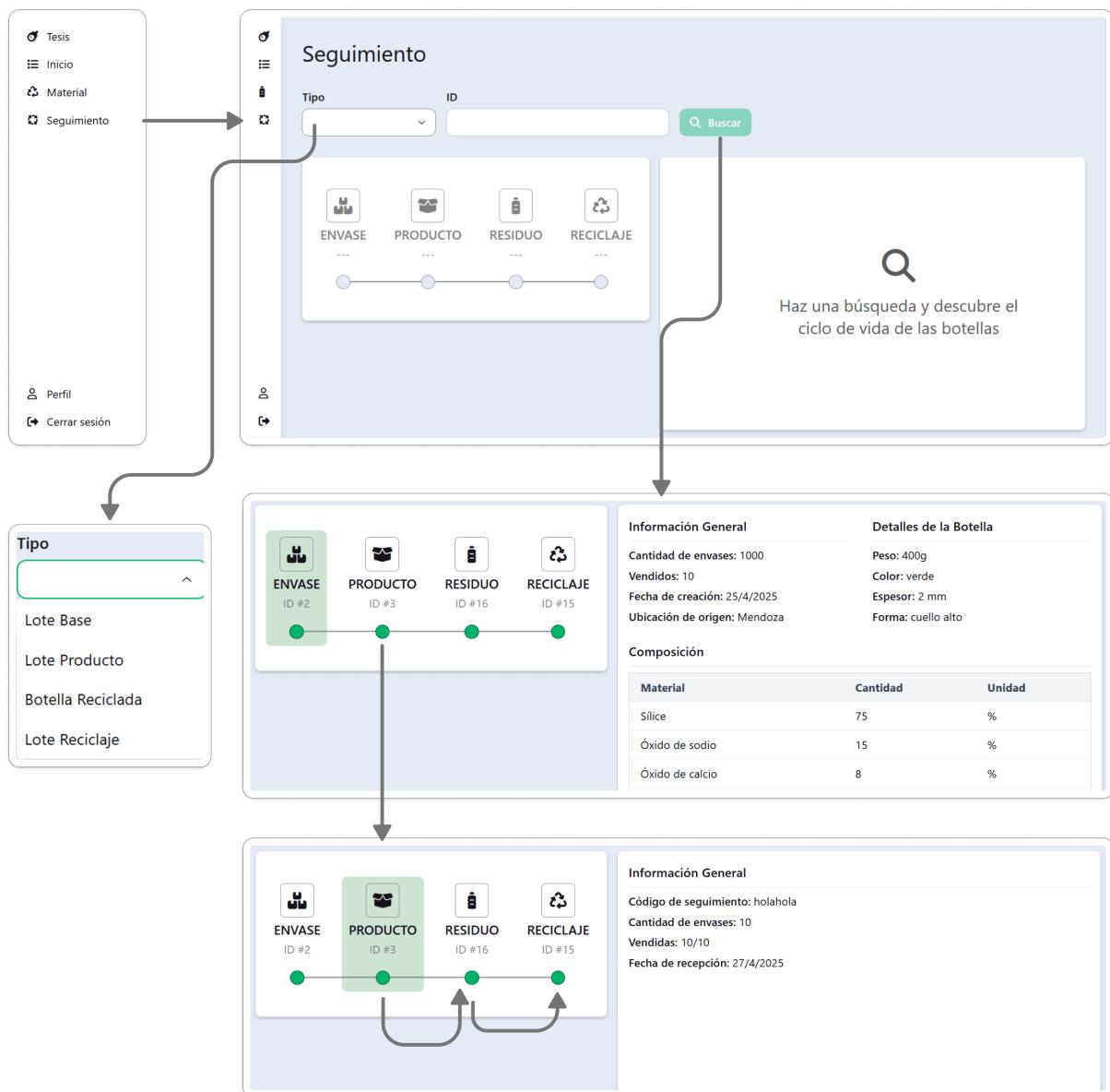


Figura D.9: Flujo de seguimiento de envases



CASOS DE PRUEBA

En este trabajo, cuyo desarrollo siguió el modelo en V, se realizaron secuencialmente múltiples instancias de pruebas del prototipo tecnológico: pruebas unitarias, pruebas de integración, pruebas de sistema y pruebas de aceptación. Cada etapa incluyó la definición y ejecución de casos de prueba, ya sean manuales o automatizados, con el objetivo de validar que el sistema cumpliera con los requerimientos funcionales y no funcionales establecidos en la fase de modelado de requerimientos. En este apéndice se detallan los casos de uso de cada etapa de pruebas, los resultados obtenidos y las incidencias detectadas y resueltas durante el proceso.

E.1. Pruebas unitarias

En la fase de pruebas unitarias, se implementaron y ejecutaron pruebas unitarias automatizadas para cada componente del sistema. Estas pruebas tuvieron como objetivo validar la funcionalidad de los contratos inteligentes, la API *backend* y la interfaz *frontend*. Cada módulo de pruebas fue diseñado para cubrir casos de uso positivos y negativos, asegurando que cada unidad de código funcionara correctamente de manera aislada. En la Tabla E.1 se presenta un resumen de las pruebas unitarias realizadas sobre los contratos inteligentes, en la Tabla E.2 las pruebas ejecutadas sobre la API (*backend*), y en la Tabla E.3 las pruebas realizadas sobre la interfaz (*frontend*). En total, se desarrollaron 98 pruebas unitarias automatizadas para los contratos inteligentes, 317 para el *backend* y 286 para el *frontend*, todas las cuales finalizaron con resultados satisfactorios sin fallos reportados. El listado completo de pruebas unitarias puede obtenerse a partir del código fuente (Sección A.2) siguiendo las instrucciones que se encuentran en el repositorio.

Tabla E.1: Resumen de pruebas unitarias realizadas sobre los contratos inteligentes

#	Contrato	Tests	Descripción	Resultado
1	RecycledMaterialContract	44	Gestión de botellas de residuo, lotes de material reciclado, operaciones CRUD, reciclaje, venta y control de autorización	OK
2	BaseBottlesBatchContract	20	Gestión de lotes de botellas base, operaciones CRUD, reciclaje, venta y control de autorización	OK
3	ProductBottlesBatchContract	32	Gestión de lotes de botellas de producto, códigos de seguimiento, reciclaje, venta, rechazo y control de autorización	OK

Tabla E.2: Resumen de pruebas unitarias realizadas sobre la API backend

#	Módulo	Tests	Descripción	Resultado
1	Tracking API	33	Endpoints para seguimiento público de productos, lotes base, lotes de productos, botellas de residuo y lotes de reciclaje	OK
2	Consumer API	41	Gestión de botellas de residuo, consulta de origen de productos, gestión de recicladores y operaciones de consumidor	OK
3	Secondary Producer API	44	Gestión de lotes de productos secundarios, códigos de seguimiento, reciclaje y venta de productos	OK
4	Producer API	54	Gestión de lotes de botellas base, operaciones CRUD, venta, reciclaje y consulta de compradores	OK
5	Helpers	36	Utilidades auxiliares: autenticación Firebase, interacción blockchain, variables de entorno, logging, middleware y validaciones	OK
6	App Setup	2	Configuración y manejo de excepciones de la aplicación	OK
7	Auth API	12	Registro de usuarios, autenticación, gestión de perfiles y filtrado por roles	OK
8	Recycler API	95	Gestión completa de reciclaje: seguimiento de botellas, creación y gestión de lotes, asignación de botellas y ventas	OK

Tabla E.3: Resumen de pruebas unitarias realizadas sobre la interfaz frontend

#	Módulo	Tests	Descripción	Resultado
1	Auth Module	8	Páginas de login y registro, validaciones de formularios y manejo de errores de autenticación	OK
2	Tracking Module	25	Página de seguimiento público, búsqueda por código, timeline y pestañas de diferentes tipos de lotes	OK
3	Consumer Module	4	Página del consumidor, modales de búsqueda y reciclaje de botellas	OK
4	Primary Producer Module	6	Inventario de productor primario, modales de gestión de lotes, venta y reciclaje	OK
5	Secondary Producer Module	5	Gestión de lotes de productos secundarios, modales de formulario, detalle, venta y reciclaje	OK
6	Recycler Module	11	Inventario de reciclador, gestión de botellas de residuo, asignación de lotes y modales asociados	OK
7	Profile Module	4	Página de perfil de usuario, validaciones y actualización de datos	OK
8	Common Components	102	Componentes reutilizables: botones, inputs, modales, tablas, iconos, tarjetas y elementos de UI	OK
9	Auth Libraries	48	Servicios de autenticación Firebase, gestión de estado, sesiones y hooks de autenticación	OK
10	Common Hooks	12	Hooks personalizados para manejo de estado, efectos y utilidades de componentes	OK
11	Common Utils	17	Utilidades de formateo, logging, construcción de URLs y servicios de request	OK

E.2. Pruebas de integración

En la fase de pruebas de integración, se verificó la interacción entre los diferentes módulos del sistema para asegurar que funcionaran correctamente en conjunto. Esto incluyó pruebas de la API y su comunicación con la base de datos y la *blockchain*.

La particularidad de las pruebas de integración es que requieren ser ejecutadas sobre el sistema completo (API, base de datos y *blockchain*); sin embargo, no requieren la interfaz de usuario. Esto se debe a que las pruebas de integración se centran en validar la correcta interacción entre los módulos del *backend* y la persistencia de datos, sin involucrar la capa de presentación. A su vez, todos los módulos mencionados deben probarse sin hacer uso de simulaciones ni *mocks*, para garantizar que las interacciones sean reales y reflejen el comportamiento esperado en un entorno de producción. Por este motivo, se configuró un *script* para las pruebas de integración que se encarga de ejecutar una instancia local de la *blockchain* con los contratos desplegados, también levanta una base de datos local y la API, permitiendo así ejecutar las pruebas de integración en un entorno controlado que simula el entorno productivo, pero en local, sin otras

dependencias externas.

En esta etapa, el enfoque no fue la exhaustividad, sino la representatividad de los flujos críticos del sistema. Por este motivo, se seleccionaron casos de prueba que cubren los flujos más críticos y relevantes del sistema, asegurando que las interacciones entre los módulos funcionen como se espera en situaciones reales. En la Tabla E.4 se presenta un listado de los casos de prueba de integración ejecutados.

Tabla E.4: Listado de pruebas de integración realizadas sobre el sistema

#	Módulo	Descripción	Resultado
1	Authentication	Registrar un nuevo usuario	OK
2	Authentication	Obtener usuario autenticado después del registro	OK
3	Authentication	Actualizar usuario autenticado después del registro	OK
4	Consumer	Obtener origen del producto por código de seguimiento	OK
5	Consumer	Crear una botella de residuo	OK
6	Consumer	Obtener botellas de residuo paginadas del usuario	OK
7	Primary Producer	Crear un lote de botellas base	OK
8	Primary Producer	Consultar lote por ID	OK
9	Primary Producer	Listar lotes del usuario con paginación	OK
10	Primary Producer	Vender lote de botellas base	OK
11	Primary Producer	Reciclar lote de botellas base	OK
12	Primary Producer	Listar lotes reciclados del usuario	OK
13	Secondary Producer	Asignar código de seguimiento a lote de producto	OK
14	Secondary Producer	Reciclar lote de producto	OK
15	Secondary Producer	Vender lote de producto (requiere código de seguimiento)	OK
16	Secondary Producer	Obtener lotes de producto paginados	OK
17	Recycler	Crear un lote de reciclaje	OK
18	Recycler	Vender un lote de reciclaje	OK

E.3. Pruebas de sistema

En el caso de las pruebas de sistema, estas buscan validar que el sistema completo funcione como se espera, cumpliendo con los requerimientos funcionales y no funcionales establecidos. En esta etapa se realizaron pruebas manuales y automatizadas sobre un entorno de pruebas en la nube similar al de producción.

En primera instancia, se llevaron a cabo pruebas manuales que incluyeron la verificación de flujos de usuario, así como la validación de la integración de la información entre los diferentes módulos del sistema. Estas pruebas se documentaron detalladamente para asegurar una correcta trazabilidad y facilitar la identificación de posibles problemas. Cada caso de prueba se

diseño previo a su ejecución, detallando el objetivo de la prueba, los requerimientos puestos a prueba, el módulo del sistema involucrado, los pasos a seguir y los resultados esperados. En la Tabla E.5 se presenta un resumen de los casos de prueba de sistema ejecutados y sus resultados, incluyendo las incidencias relevadas. Mientras que en la Tabla E.6 se detallan las incidencias relevadas durante la ejecución de las pruebas de sistema, indicando su estado actual (resueltas o desestimadas), identificador de la incidencia en Jira y una breve descripción de cada una.

Debido a la extensión de los casos de prueba no se incluyen en este documento, pero la lista completa se encuentra disponible en el siguiente enlace: https://github.com/RocioCM/computer-science-thesis/blob/main/docs/Pruebas-de-Sistema_Test-Cases.pdf

Además de las pruebas manuales, se implementaron pruebas automatizadas para validar los requerimientos no funcionales del sistema. En este caso, se implementaron pruebas de carga y de seguridad. Las pruebas de seguridad buscan asegurar que se cumplen los requerimientos de autenticación y autorización de usuarios especificados en las historias de usuario. Mientras que las pruebas de carga buscan evaluar el rendimiento del sistema bajo condiciones de alta demanda. Las pruebas de carga se realizaron en un entorno local utilizando la herramienta K6¹, poniendo a prueba 3 *endpoints* representativos de la API y representan únicamente un resultado orientativo sobre cómo puede comportarse el sistema en un entorno de producción real, ya que el rendimiento dependerá de múltiples factores, como la infraestructura subyacente y la configuración del entorno. En la Tabla E.7 se presentan los resultados de las pruebas de seguridad realizadas, mientras que en la Tabla E.8 se presentan los resultados de las pruebas de carga.

Tabla E.5: Resumen de pruebas de sistema realizadas por módulo

#	Módulo	Tests	Descripción	Resultado	Incidencias
1	Auth Module	9	Registro, login, manejo de sesión, autorización, perfil de usuario (visualización y edición)	8/9 OK	1
2	Producers Module	12	Creación, edición y eliminación de lotes; validaciones de campos; consultas de historial y materiales; ventas y reciclaje; restricciones de operaciones sobre lotes vendidos	9/12 OK	2
3	Secondary Producers Module	13	Asociación, edición y eliminación de códigos; consultas de inventario; rechazo, reciclaje y venta de lotes; validación de restricciones en acciones sobre lotes vendidos	10/13 OK	2

Continúa en la siguiente página

¹ Load testing tool for engineering teams: <https://k6.io/>

#	Módulo	Tests	Descripción	Resultado	Incidencias
4	Consumers Module	5	Consulta de trazabilidad, registro de envases reciclables, seguimiento de envases y validación de búsquedas según estado de comercialización	3/5 OK	2
5	Recyclers Module	7	Consulta de envases, creación, edición y eliminación de lotes reciclados; ventas de material reciclado; bloqueo de acciones sobre lotes vendidos	6/7 OK	1
6	Tracking Module	9	Búsqueda y trazabilidad de lotes base, productos, botellas recicladas y lotes de reciclaje; validación de habilitación de campos y mensajes de error	8/9 OK	1

Tabla E.6: Lista de errores hallados e incidencias relevadas en pruebas de sistema

#	Módulo	Descripción	ID Jira	Estado
1	Auth Module	Tras registrar usuario exitoso, los inputs quedan en estado de error en lugar de reiniciarse	SCRUM-43	Resuelto
2	Auth Module	Al acceder a ruta no permitida, la pantalla queda en blanco en lugar de mostrar mensaje de autorización denegada	SCRUM-42	Resuelto
3	Producers Module	Materiales reciclados no se mostraban en el listado del mismo productor	SCRUM-36	Resuelto
4	Producers Module	Error al intentar reciclar más unidades que las disponibles: se muestra mensaje genérico y la API devuelve 500 en lugar de 400	SCRUM-37	Resuelto
5	Producers Module	Error al intentar vender más unidades que las disponibles: mensaje genérico y API devuelve 500 en lugar de 400	SCRUM-37	Resuelto
6	Secondary Producers Module	Material reciclado no aparecía en dashboard del productor primario de origen tras reciclar lote	SCRUM-36	Resuelto
7	Secondary Producers Module	Error al reciclar más unidades que disponibles: mensaje genérico y API 500 en lugar de mensaje claro	SCRUM-37	Resuelto
8	Secondary Producers Module	Error al vender más unidades que disponibles: mensaje genérico y API 500 en lugar de mensaje claro	SCRUM-37	Resuelto
9	Consumers Module	Al dar seguimiento a envase, el modal no se abre (acción inactiva)	SCRUM-38	Resuelto

Continúa en la siguiente página

#	Módulo	Descripción	ID Jira	Estado
10	Consumers Module	Consumidor puede buscar códigos de botellas sin comercializar y se habilita acción de reciclaje (comportamiento no esperado)	SCRUM-39	Resuelto
11	Recyclers Module	Timeout al crear lote de material reciclado, API 500 aunque el lote se crea	SCRUM-44	Resuelto
12	Tracking Module	Error al buscar lote de reciclaje inexistente: se muestran placeholders vacíos en vez de mensaje de error adecuado	SCRUM-40	Resuelto

Tabla E.7: Casos de prueba de seguridad ejecutados sobre el sistema

#	Tipo	Descripción	Resultado
1	Autenticación	Verifica que el acceso a un endpoint protegido sin credenciales retorne 401 Unauthorized	OK
2	Autenticación	Verifica que el acceso a un endpoint protegido con token inválido retorne 401 Unauthorized	OK
3	Autenticación	Verifica que el acceso a un endpoint protegido con token expirado retorne 401 Unauthorized	OK
4	Autorización	Verifica que el acceso a un endpoint protegido con rol no autorizado retorne 403 Forbidden	OK

Tabla E.8: Resumen de pruebas de carga realizadas sobre el sistema

#	Max. Conexiones	Solicitudes	Fallos	Duración Promedio
1	50	2173	0.00 %	1.45s
2	100	5184	0.00 %	5.35s
3	200	4332	0.00 %	13.12s
4	1000	27171	95.71 %	8.16s

E.4. Pruebas de aceptación

En la instancia de pruebas de aceptación, se realizó un conjunto de pruebas manuales para validar el comportamiento del prototipo desde la perspectiva del usuario final. Estas pruebas se llevaron a cabo en un entorno controlado, donde los participantes interactuaron con el sistema y proporcionaron retroalimentación sobre su experiencia. Al comienzo de la sesión, se asignó un rol a cada participante y se le proporcionó una guía básica sobre el uso del sistema, asegurando que comprendieran las funcionalidades principales y los objetivos de las pruebas.

En la Tabla E.9 se presenta el listado de los participantes, junto con la cantidad de pruebas realizadas e incidencias generadas a partir de su interacción con el sistema. En la Tabla E.10 se presenta un resumen de las incidencias detectadas durante las pruebas.

El listado completo de casos de prueba ejecutados por cada participante se puede consultar en el siguiente enlace: https://github.com/RocioCM/computer-science-thesis/blob/main/docs/Pruebas-de-Aceptacion_Test-Cases.pdf

Tabla E.9: Listado de participantes en pruebas de aceptación de usuario

#	Nombre	Rol	Pruebas ejecutadas	Incidencias
1	Francisco	Productor Primario	4	0
2	Luciano	Productor Primario	7	1
3	Facundo	Productor Secundario	1	1
4	Yeumen	Productor Secundario	9	1
5	Lucas	Productor Secundario	7	0
6	Micaela	Consumidor	15	3
7	Agustín	Consumidor	6	2
8	Andrés	Reciclador	11	3
9	Lucía	Reciclador	3	0

Tabla E.10: Lista de errores hallados e incidencias relevadas en pruebas de aceptación

#	Rol	Descripción	ID Jira	Estado
1	Productor Primario	Inconsistencia en stock al rechazar lote vendido	SCRUM-52	Resuelto
2	Productor Secundario	Overflow en input number permite venta con valores inválidos	SCRUM-51	Resuelto
3	Productor Secundario	Campo de correo no reconocido correctamente en pantalla de venta	SCRUM-46	Resuelto
4	Consumidor	Error al guardar nombre de usuario con caracteres especiales/emoji	SCRUM-47	Resuelto
5	Consumidor	Al llegar al fin de la lista no se muestran las opciones disponibles	SCRUM-48	Resuelto
6	Consumidor	Errores al eliminar botellas recicladas o no recicladas (mensajes poco claros)	SCRUM-49	Resuelto
7	Consumidor	Problemas de navegación y visualización en seguimiento de envases	SCRUM-46	Resuelto
8	Reciclador	Estado pegado al cambiar rápidamente entre pestañas de disponibles/reciclados	SCRUM-50	Resuelto

Continúa en la siguiente página

#	Rol	Descripción	ID Jira	Estado
9	Reciclador	Modal de detalle de envases no funciona correctamente	SCRUM-45	Resuelto
10	Reciclador	Tabla de envases no se refresca correctamente al cambiar de pestaña o cargar envases nuevos	SCRUM-34	Resuelto

BIBLIOGRAFÍA

- [1] Hamzah Alaidaros, Mazni Omar y Rohaida Romli, «The state of the art of agile kanban method: challenges and opportunities», *Independent Journal of Management & Production*, vol. 12, n.º 8, pp. 2535-2550, 2021. <https://doi.org/10.14807/ijmp.v12i8.1482>.
- [2] Eiman Alnuaimi y col., «Blockchain-based system for tracking and rewarding recyclable plastic waste», *Peer-to-Peer Networking and Applications*, vol. 16, n.º 1, pp. 328-346, 2023. <https://doi.org/10.1007/s12083-022-01413-5>.
- [3] Elli Androulaki y col., «Hyperledger fabric: a distributed operating system for permissioned blockchains», en *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1-15. <https://doi.org/10.1145/3190508.3190538>.
- [4] J. I. Arroyo y col., *Clima. El gato y la caja*, 2022, ISBN 9789874863812. <https://elgatoylacaja.com/libros/clima> (visitado 20-09-2025).
- [5] Gavina Baralla, Andrea Pinna, Roberto Tonelli y Michele Marchesi, «Waste management: A comprehensive state of the art about the rise of blockchain technology», *Computers in Industry*, vol. 145, p. 103 812, 2023. <https://doi.org/10.1016/j.compind.2022.103812>.
- [6] Alejandro Bartolomeo y Gustavo Machin Urbay, «Introducción a la tecnología blockchain: su impacto en las Ciencias Económicas», *Ponencia presentada en Jornadas de Ciencias Económicas. Buenos Aires*, vol. 7, n.º 8, 2020. <https://bdigital.uncu.edu.ar/fichas.php?idobjeto=15304>.
- [7] Ahmad Bathaei, Bahador Bahramimianrood y Siti Rahmah Awang, «Blockchain-enabled circular economy: Rethinking waste management in global manufacturing», *Transformations and Sustainability*, vol. 1, n.º 2, pp. 118-131, 2025. <https://doi.org/10.63775/1xwx6s13>.
- [8] Kesaven Bhubalan y col., «Leveraging blockchain concepts as watermarks of plastics for sustainable waste management in progressing circular economy», *Environmental Research*, vol. 213, p. 113 631, 2022, ISSN 0013-9351. <https://doi.org/10.1016/j.envres.2022.113631>.
- [9] Katarzyna Bułkowska, Magdalena Zielińska y Maciej Bułkowski, «Implementation of Blockchain Technology in Waste Management», *Energies*, vol. 16, n.º 23, p. 7742, 2023. <https://doi.org/10.3390/en16237742>.

- [10] Vitalik Buterin, «Ethereum white paper», 2014. https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf (visitado 20-09-2025).
- [11] CEPAL, NU, «Economía circular en América Latina y el Caribe: oportunidad para una recuperación transformadora», 2021. <https://hdl.handle.net/11362/47309>.
- [12] Emilio Cerdá y Aygun Khalilova, «Economía circular», *Economía industrial*, vol. 401, n.º 3, pp. 11-20, 2016, ISSN 0422-2784. <https://dialnet.unirioja.es/servlet/articulo?codigo=5771932> (visitado 20-09-2025).
- [13] Circularise. (2025). Integrations, <https://www.circularise.com/feature/integrations> (visitado 20-09-2025).
- [14] (2025). Circulor, <https://www.circulor.com/about-us> (visitado 20-09-2025).
- [15] Colmena, 2024. <https://www.colmenaproject.io/en#services> (visitado 20-09-2025).
- [16] Clayson Cosme Da Costa Pimenta, «La Economía Circular como eje de desarrollo de los países latinoamericanos», *Revista Economía y Política*, n.º 35, pp. 1-18, 2022. <https://doi.org/10.25097/rep.n35.2022.01>.
- [17] Arianna De Bellis, «Sustainability of the German Deposit system: Case study in Berlin», Tesis doctoral, Politecnico di Torino, 2020. <http://webthesis.biblio.polito.it/id/eprint/13778>.
- [18] Francisco Javier Díaz, Mónica Diana Tugnarelli y Mauro F Fornaroli, «Protocolos de consenso», en *XXIV Workshop de Investigadores en Ciencias de la Computación (WICC 2022, Mendoza)*, 2022. <https://sedici.unlp.edu.ar/handle/10915/144937>.
- [19] Alberto Díez Arias y col., «Web 3.0 y Blockchain en la Educación Secundaria», 2023. <https://uvadoc.uva.es/handle/10324/63058>.
- [20] Leonor Dormido, Isabel Garrido, Pilar L'Hotellerie-Fallois y Javier Santillán Fraile, «El cambio climático y la sostenibilidad del crecimiento: iniciativas internacionales y políticas europeas», *Documentos Ocasionales/Banco de España*, 2213, 2022. <https://repositorio.bde.es/handle/123456789/22528>.
- [21] Ellen MacArthur Foundation. (2014). What is a circular economy?, <https://www.ellenmacarthurfoundation.org/topics/circular-economy-introduction/overview> (visitado 20-09-2025).
- [22] Carlos Gómez Gil, «Objetivos de Desarrollo Sostenible (ODS): una revisión crítica», *Papeces de relaciones ecosociales y cambio global*, n.º 140, pp. 107-118, 2018, ISSN 1888-0576. <https://dialnet.unirioja.es/servlet/articulo?codigo=6312616> (visitado 20-09-2025).
- [23] Greenly Points, 2024. <https://maipu.gob.ar/maipu-estrena-su-innovador-punto-de-ecocanje-en-el-edificio-municipal/> (visitado 20-09-2025).
- [24] Ali Gunawan, Richard Richard, Sinclair Claus Chang y Shilvi Shilvi, «A review of data security of blockchain applications in social media», en *AIP Conference Proceedings*, AIP Publishing, vol. 3026, 2024. <https://doi.org/10.1063/5.0199772>.

- [25] Charles Hoskinson, «Why we are building Cardano», IOHK, 2017. <https://why.cardano.org/en/introduction/motivation/> (visitado 20-09-2025).
- [26] Innovar Sustentabilidad, *El vidrio ya tiene su planta de reciclaje en la Argentina*, Descripción del programa de reciclaje de vidrio a domicilio en San Isidro con la organización Reciqlo, 2022. <https://www.innovar-sustentabilidad.com/el-vidrio-ya-tiene-su-planta-de-reciclaje-en-la-argentina/> (visitado 20-09-2025).
- [27] Jovan Kalajdjeski, Mayank Raikwar, Nino Arsov, Goran Velinov y Danilo Gligoroski, «Databases fit for blockchain technology: A complete overview», *Blockchain: Research and Applications*, vol. 4, n.º 1, p. 100 116, 2023. <https://doi.org/10.1016/j.bcra.2022.100116>.
- [28] Robert C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, 1st. USA: Prentice Hall Press, 2017, ISBN 0134494164.
- [29] Jesus R Melendez, Jorge Luis Delgado, Víctor Chero y John Franco Rodríguez, «Economía Circular: Una Revisión desde los Modelos de Negocios y la Responsabilidad Social Empresarial», *Revista Venezolana de Gerencia: RVG*, vol. 26, n.º 6, pp. 560-573, 2021. <https://doi.org/10.52080/rvgluz.26.e6.34>.
- [30] Ministerio de Ambiente y Desarrollo Sostenible, «Estrategia Nacional de Consumo y Producción Sostenibles», 2021. https://www.argentina.gob.ar/sites/default/files/encps_1.pdf (visitado 20-09-2025).
- [31] Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO). (2023). España Circular 2030: Estrategia Española de Economía Circular, <https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/economia-circular/estrategia.html> (visitado 20-09-2025).
- [32] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Original Bitcoin white-paper, octubre de 2008. <https://bitcoin.org/bitcoin.pdf>.
- [33] ONU. (2015). Objetivos de Desarrollo Sostenible, <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/> (visitado 20-09-2025).
- [34] Josep Lluís Pelegrí, «Informe IPCC: Certezas e incertidumbres sobre el cambio climático», 2021. <http://hdl.handle.net/10261/261038>.
- [35] Caterina Picuno, Spyridoula Gerassimidou, Weimu You, Olwenn Martin y Eleni Iacovidou, «The potential of Deposit Refund Systems in closing the plastic beverage bottle loop: A review», *Resources, Conservation and Recycling*, vol. 212, p. 107 962, 2025. <https://doi.org/10.1016/j.resconrec.2024.107962>.
- [36] H.-O. Pörtner y col., eds., *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2022, In Press. <https://doi.org/10.1017/9781009325844>.

- [37] Roger S Pressman, *Ingeniería de software: Un enfoque práctico*. 2010, vol. 7, ISBN 9786071503145.
- [38] M Abeedur Rahman, Kaushik Chowdhury, Lutfun Nahar Chowdhury, Shuvo Kumar Mallik y Noushin Akhter Nova, «The Future is Traceable: Integrating Blockchain, Waste Management, and the Circular Economy for Sustainability», vol. 5, pp. 229-248, septiembre de 2025. <https://doi.org/10.63332/joph.v5i9.3310>.
- [39] Reciclos, 2024. <https://www.ecoembes.com/es/el-proceso-del-reciclaje-de-envases/reciclos> (visitado 20-09-2025).
- [40] Abderahman Rejeb y col., «The role of blockchain technology in the transition toward the circular economy: Findings from a systematic literature review», *Resources, Conservation & Recycling Advances*, vol. 17, p. 200126, 2023. <https://doi.org/10.1016/j.rcradv.2022.200126>.
- [41] Michael JW Rennock, Alan Cohn y Jared R Butcher, «Blockchain technology and regulatory investigations», *Practical Law Litigation*, vol. 1, pp. 35-44, 2018.
- [42] Óscar Rodríguez, Guillermo Rudas Lleras, Erika Nieves, Julián Roa y María Paula Rivera, «Modelamiento de los efectos macroeconómicos de la transición a la economía circular en América Latina: Los casos de Chile, Colombia, México y el Perú», 2023. <https://hdl.handle.net/11362/48751>.
- [43] R Sandhiya y Seeram Ramakrishna, «Investigating the applicability of blockchain technology and ontology in plastics recycling by the adoption of ZERO plastic model», *Materials Circular Economy*, vol. 2, pp. 1-12, 2020. <https://doi.org/10.1007/s42824-020-00013-z>.
- [44] Reuben Schuitemaker y Xun Xu, «Product traceability in manufacturing: A technical review», *Procedia CIRP*, vol. 93, pp. 700-705, 2020, 53rd CIRP Conference on Manufacturing Systems 2020, ISSN 2212-8271. <https://doi.org/10.1016/j.procir.2020.04.078>.
- [45] Abdel-Aziz Ahmad Sharabati y Elias Radi Jreisat, «Blockchain technology implementation in supply chain management: a literature review», *Sustainability*, vol. 16, n.º 7, p. 2823, 2024. <https://doi.org/10.3390/su16072823>.
- [46] Signeblock, *Soluciones de valor para la transformación digital de las empresas*, Sitio web, 2024. <https://www.signeblock.com/trazabilidad-blockchain-so-5-es> (visitado 20-09-2025).
- [47] Farhana Akter Sunny y col., «A systematic review of blockchain applications», *Ieee Access*, vol. 10, pp. 59 155-59 177, 2022. <https://doi.org/10.1109/ACCESS.2022.3179690>.
- [48] Hamed Taherdoost, «Smart contracts in blockchain technology: A critical review», *Information*, vol. 14, n.º 2, p. 117, 2023. <https://doi.org/10.3390/info14020117>.
- [49] Juan Esteban Torres Castro y Paula Valentina Marín Agudelo, «Las tendencias en el uso del blockchain en el área de la cadena de suministro», 2022. <https://hdl.handle.net/20.500.12495/10594>.
- [50] Gautami Tripathi, Mohd Abdul Ahad y Gabriella Casalino, «A comprehensive review of blockchain technology: Underlying principles and historical background with future

- challenges», *Decision Analytics Journal*, vol. 9, p. 100344, 2023. <https://doi.org/10.1016/j.dajour.2023.100344>.
- [51] Karthik Kumar Vaigandla, RadhaKrishna Karne, Mounika Siluveru y Madhavi Kesojiu, «Review on blockchain technology: architecture, characteristics, benefits, algorithms, challenges and applications», *Mesopotamian Journal of CyberSecurity*, vol. 2023, pp. 73-84, 2023. <https://doi.org/10.58496/MJCS/2023/012>.
- [52] Verallia, «Reimagining reuse for the circular economy of glass: Stakeholder Perspectives Series», 2022. https://www.verallia.com/re-use/en/publication/contents/templates/VERALLIA_WHITE-BOOK_EN.pdf (visitado 20-09-2025).
- [53] Verallia Argentina, «Vidrio, una acción transparente», 2025. https://ar.verallia.com/s/vidrio-une-accion-transparente?language=es_AR (visitado 20-09-2025).
- [54] Verallia Iberia, «Proceso de fabricación del vidrio», 2025. <https://es.verallia.com/s/proceso-de-fabricacion-del-vidrio?language=es> (visitado 20-09-2025).
- [55] Varun Verma, «An overview of Blockchain Technology: Past & Future», *Eduzone: International Peer Reviewed/Refereed Multidisciplinary Journal*, vol. 12, n.º 1, pp. 100-104, 2023. <https://www.eduzonejournal.com/index.php/eiprmj/article/view/262>.
- [56] Elton Kee Sheng Wong, Huong Yong Ting y Abdulwahab Funsho Atanda, «Enhancing supply chain traceability through blockchain and IoT integration: A comprehensive review», *Green Intelligent Systems and Applications*, vol. 4, n.º 1, pp. 11-28, 2024. <https://doi.org/10.53623/gisa.v4i1.355>.

GLOSARIO

API

Application Programming Interface (API) es un conjunto de protocolos, herramientas y definiciones que permiten que diferentes aplicaciones de software se comuniquen entre sí de manera estructurada. Las APIs definen métodos de solicitud, formatos de datos, convenciones y reglas que deben seguir las aplicaciones para intercambiar información. En sistemas de trazabilidad *blockchain*, las APIs actúan como intermediarios que permiten a las aplicaciones web interactuar con los contratos inteligentes y consultar datos de la cadena de bloques. (p. 62–64, 71)

API REST

API REST (*Representational State Transfer*) es un modelo arquitectónico para diseñar servicios web que utiliza los métodos estándar de HTTP (GET, POST, PUT, DELETE) para realizar operaciones sobre recursos identificados por URLs. Las APIs REST no mantienen estado entre solicitudes y utilizan una interfaz uniforme, lo que las hace escalables y fáciles de implementar. Para una aplicación descentralizada, las REST APIs proporcionan *endpoints* que permiten a las aplicaciones *frontend* consultar y actualizar información tanto de bases de datos tradicionales como de contratos inteligentes. (p. 63–65, 71)

backend

El *backend* es la parte de una aplicación de software que se ejecuta en el servidor y maneja la lógica de negocio, el procesamiento de datos, la autenticación, las conexiones a bases de datos y las comunicaciones con servicios externos. No es directamente visible para los usuarios finales, pero proporciona la funcionalidad que soporta el *frontend*. En un sistema de trazabilidad *blockchain*, el *backend* gestiona las interacciones con los contratos inteligentes, procesa las transacciones *blockchain* y administra las bases de datos complementarias. (p. 56–58, 62, 65, 70, 71, 79, 80, 84, 86, 123)

base de datos

Una base de datos es un sistema organizado para almacenar, gestionar y recuperar información de manera estructurada. Permite a las aplicaciones almacenar datos persistentes, realizar consultas complejas y mantener la integridad de la información a través de reglas y restricciones. En sistemas de trazabilidad, las bases de datos permiten almacenar metadatos, información de sesiones de usuario y datos sobre el estado de los productos. (p. 22, 56, 58, 66, 69, 70, 76, 86, 125)

blockchain	<p><i>Blockchain</i> es una estructura de datos distribuida que mantiene un registro inmutable de transacciones o eventos mediante técnicas criptográficas que protegen contra la manipulación. La información se organiza en transacciones que son validadas y agrupadas en bloques. Cada bloque, junto con un puntero al bloque anterior, forma una cadena de transacciones interconectadas. Una vez que una transacción se ha añadido a la cadena, no puede ser modificada ni eliminada porque requeriría cambiar todos los bloques posteriores en la cadena, lo cual es computacionalmente impracticable debido a la distribución y la seguridad criptográfica de la red <i>blockchain</i> [41]. (p. 5, 7, 56, 86, 125)</p>
cadena de suministro	<p>La cadena de suministro constituye el entramado logístico, operativo y estratégico que permite el flujo de materiales, información y recursos desde la extracción de materias primas hasta la llegada de un producto al consumidor final. Involucra múltiples etapas y actores como proveedores, fabricantes, distribuidores, comerciantes y, en modelos circulares, gestores de residuos y reguladores. Su objetivo es garantizar que los bienes y servicios se produzcan y entreguen de manera eficiente, segura y rentable [42]. (p. 16, 17, 20, 26, 29, 59, 60, 62, 103)</p>
Clean Architecture	<p><i>Clean Architecture</i> es un patrón de diseño de software propuesto por Robert C. Martin que organiza el código en capas concéntricas, donde las dependencias apuntan hacia el centro. Este enfoque busca crear sistemas independientes de <i>frameworks</i>, bases de datos, interfaces de usuario y elementos externos, facilitando el <i>testing</i>, mantenimiento y evolución del software. Las capas internas contienen la lógica de negocio y son independientes de las capas externas que manejan aspectos técnicos como bases de datos o interfaces web. (p. 70)</p>
cliente-servidor	<p>El modelo <i>Cliente-Servidor</i> es una arquitectura de red en la que las tareas se distribuyen entre proveedores de recursos o servicios, llamados servidores, y solicitantes de servicios, llamados clientes. Los clientes inician solicitudes de servicios y los servidores responden a estas solicitudes, gestionando recursos como bases de datos, archivos o aplicaciones. Este modelo permite la centralización de recursos y facilita la escalabilidad y mantenimiento del sistema. Este modelo es ampliamente utilizado en aplicaciones web y móviles, donde el cliente (navegador o aplicación) interactúa con el servidor para acceder a datos y funcionalidades. (p. 6)</p>
contrato inteligente	<p>Un contrato inteligente es un programa autoejecutado que implementa lógica de negocio directamente en una <i>blockchain</i>, ejecutándose automáticamente cuando se cumplen condiciones predefinidas sin necesidad de intermediarios. Estos contratos son inmutables una vez desplegados y proporcionan transparencia y confianza en las transacciones ejecutadas [10]. (p. 7, 13, 16, 17, 22, 27, 34, 57, 59, 61, 64, 67, 75–77, 79, 123)</p>

coverage	<p><i>Coverage</i> o cobertura de código es una métrica de testing que mide el porcentaje del código fuente que es ejecutado durante la ejecución de pruebas automatizadas. Una alta cobertura indica que la mayoría del código ha sido probado, lo que incrementa la confianza en la calidad del software y ayuda a identificar partes del código que no han sido testeadas. En el desarrollo de contratos inteligentes para sistemas de trazabilidad, el <i>coverage</i> permite asegurar que todas las funciones críticas han sido probadas antes del despliegue en <i>blockchain</i>, aunque no garantiza la ausencia de errores. (p. 85, 86)</p>
criptomoneda	<p>Una <i>criptomoneda</i> es una moneda digital o virtual que utiliza criptografía para asegurar la integridad de las transacciones, controlar la emisión de nuevos fondos y verificar la transferencia de activos. Las criptomonedas operan de manera descentralizada en redes <i>blockchain</i>, lo que las hace independientes de autoridades centrales como bancos o gobiernos. Ejemplos populares de criptomonedas incluyen Bitcoin y Ethereum. (p. 7, 10, 15, 16, 28, 59, 60)</p>
CSS	<p><i>Cascading Style Sheets</i> (CSS) es un lenguaje de definición de estilos para páginas web, utilizado para describir la presentación visual de documentos HTML y XML, incluyendo colores, fuentes, espaciado, disposición y animaciones. CSS permite separar el contenido de la presentación, facilitando el mantenimiento y la consistencia visual en aplicaciones web. (p. 65)</p>
dApps	<p>Una aplicación descentralizada (<i>dApp</i>) es una aplicación de software que se ejecuta en una red <i>blockchain</i> en lugar de servidores centralizados. Las <i>dApps</i> combinan contratos inteligentes en el <i>backend</i> con interfaces de usuario tradicionales en el <i>frontend</i>, permitiendo a los usuarios interactuar con servicios descentralizados de manera transparente. (p. 14, 59)</p>
DER	<p>Un Diagrama Entidad-Relación (DER) es una herramienta de modelado conceptual que representa gráficamente las entidades de un sistema de información, sus atributos y las relaciones entre ellas. Los DER son fundamentales en el diseño de bases de datos, ya que ayudan a visualizar la estructura lógica de los datos antes de su implementación real. En sistemas de trazabilidad, los DER modelan entidades como productos, lotes, ubicaciones y actores de la cadena de suministro, así como sus interrelaciones. (p. 69)</p>
DRS	<p><i>Deposit Return Scheme</i> (DRS) o Sistema de Retorno de Depósito es un mecanismo económico donde los consumidores pagan un pequeño depósito al comprar productos con envases retornables, el cual es reembolsado cuando devuelven el envase vacío para su reciclaje o reutilización. Los DRS han demostrado ser altamente efectivos para aumentar las tasas de reciclaje de envases, especialmente botellas y latas de bebidas. (p. 28, 110–112)</p>

economía circular

La economía circular es un modelo económico regenerativo que busca minimizar el desperdicio y maximizar el aprovechamiento de recursos mediante la reutilización, reparación, renovación y reciclaje de materiales y productos existentes. A diferencia del modelo lineal tradicional de “extraer-producir-desechar”, la economía circular mantiene los productos y materiales en uso durante el mayor tiempo posible, extrayendo el máximo valor de ellos antes de su recuperación y regeneración. Este enfoque busca reducir la presión sobre los recursos naturales y minimizar el impacto ambiental de la producción y el consumo [12]. (p. 16–18, 20, 22, 25, 27, 41, 47, 109)

endpoint

Un *endpoint* es una URL específica dentro de una API que representa un recurso o una funcionalidad particular a la que se puede acceder mediante solicitudes HTTP. Cada *endpoint* define un punto de entrada para interactuar con el sistema, permitiendo operaciones como la obtención, creación, actualización o eliminación de datos. En un sistema de trazabilidad *blockchain*, los *endpoints* de una API REST pueden permitir a las aplicaciones *frontend* consultar información de productos, registrar nuevas transacciones y comunicarse con contratos inteligentes en la *blockchain*. (p. 63, 71, 76, 80, 81, 84, 85, 124, 129)

frontend

El *frontend* es la parte de una aplicación de software con la que los usuarios interactúan directamente, incluyendo la interfaz de usuario, elementos visuales, navegación y experiencia de usuario. Se ejecuta en el lado del cliente (navegador web o aplicación móvil) y se comunica con el *backend* para obtener y enviar datos. En sistemas de trazabilidad *blockchain*, el *frontend* permite a los usuarios consultar información de productos, visualizar el historial de trazabilidad y realizar operaciones como registro de nuevos lotes o transferencias. (p. 56, 57, 64, 72, 76, 80, 84, 86, 89, 123, 125)

hash

Un *hash* es una función criptográfica que transforma datos de entrada de cualquier tamaño en una cadena de caracteres de longitud fija, actuando como una “huella digital” única de los datos originales. En *blockchain*, los *hashes* son fundamentales para garantizar la integridad de los datos, ya que se utilizan para enlazar bloques secuencialmente. Los *hashes* tienen la propiedad de que cualquier modificación en el contenido original resulta en un *hash* completamente diferente, lo que hace que sea computacionalmente impracticable alterar datos sin detección. (p. 7)

HTTP

Hypertext Transfer Protocol (HTTP) es el protocolo de comunicación fundamental de la Internet que define cómo se formatean y transmiten los mensajes entre clientes y servidores web. HTTP establece las reglas para el intercambio de recursos web como páginas HTML, imágenes, videos y datos de aplicaciones. En sistemas de trazabilidad *blockchain*, HTTP es comúnmente utilizado para las comunicaciones entre interfaces web y APIs que interactúan con los contratos inteligentes y bases de datos. (p. 63)

ingeniería de software	La ingeniería de software es una disciplina que aplica principios de ingeniería al diseño, desarrollo, operación y mantenimiento de software. El objetivo es producir software de alta calidad que satisfaga las necesidades del usuario, sea mantible y escalable. Esta disciplina se enfoca en gestionar la complejidad del desarrollo de sistemas de software a gran escala, integrando la programación con metodologías, herramientas y buenas prácticas. (p. 15, 32, 33, 67, 80, 83)
IoT	<i>Internet of Things</i> (IoT) es un paradigma tecnológico que permite la interconexión de dispositivos físicos cotidianos a través de Internet, dotándolos de capacidades de comunicación, monitoreo y control remoto. En el contexto de la trazabilidad de cadenas de suministro, los dispositivos IoT como sensores RFID, códigos QR y sensores ambientales recopilan y transmiten datos en tiempo real sobre la ubicación, estado y condiciones de los productos a lo largo de su ciclo de vida, facilitando la construcción de sistemas de trazabilidad integrales. (p. 3, 16, 22, 23, 27, 28, 56, 60, 62, 71, 95)
JSON	<i>JavaScript Object Notation</i> (JSON) es un formato ligero de intercambio de datos que es fácil de leer y escribir para humanos y simple de interpretar y generar para las computadoras. JSON se basa en la notación de objetos de JavaScript pero es independiente del lenguaje de programación. En APIs REST, JSON se utiliza para estructurar los datos enviados y recibidos entre clientes y servidores. (p. 63)
mecanismo de consenso	Un mecanismo de consenso es un protocolo utilizado en redes distribuidas, como una <i>blockchain</i> , que permite a los nodos de la red llegar a un acuerdo sobre el estado actual del sistema o sobre qué transacciones son válidas y deben ser agregadas al registro o cadena de bloques. El objetivo principal de un mecanismo de consenso es asegurar que todos los participantes de la red lleguen a un consenso o acuerdo sobre la verdad de los datos, incluso cuando algunos participantes puedan ser deshonestos o intenten manipular la red, sin depender de un servidor o nodo central. Un mecanismo de consenso eficaz debe ser seguro, resistente a la censura, tolerante a fallas y verificable en tiempo real [18]. (p. 9–11, 59, 61)
MVC	<i>Model-View-Controller</i> (MVC) es un patrón de arquitectura de software que separa la lógica de una aplicación en tres componentes interconectados: el Modelo (datos y lógica de negocio), la Vista (interfaz de usuario) y el Controlador (intermediario que gestiona la entrada del usuario y coordina el modelo y la vista). Esta separación permite un desarrollo más organizado, facilita el mantenimiento del código y posibilita que diferentes equipos trabajen simultáneamente en los distintos componentes. (p. 71)

nodo	Un <i>nodo</i> es un dispositivo conectado dentro de una red distribuida, como una <i>blockchain</i> , que participa en la validación, almacenamiento y transmisión de datos. Los nodos pueden tener diferentes roles, como nodos completos que mantienen una copia completa de la <i>blockchain</i> y validan todas las transacciones, o nodos ligeros que dependen de nodos completos para ciertas funciones. En una red <i>blockchain</i> , los nodos trabajan conjuntamente para asegurar la integridad y seguridad del sistema mediante la verificación de transacciones y la participación en mecanismos de consenso. (p. 9, 10, 12, 14, 16)
POO	La Programación Orientada a Objetos (POO) es un paradigma de programación que organiza el software en torno a objetos que contienen tanto datos (atributos) como comportamiento (métodos). Los principios fundamentales de la POO incluyen encapsulamiento, herencia, polimorfismo y abstracción, que facilitan la reutilización de código, mantenimiento y escalabilidad. (p. 67)
QR	<i>Quick Response</i> (QR) es un tipo de código de barras bidimensional que puede almacenar información en una matriz de puntos y puede ser leído rápidamente por dispositivos móviles equipados con cámaras. Los códigos QR pueden contener varios tipos de datos como URLs, texto, números de teléfono o identificadores únicos. En sistemas de trazabilidad, los códigos QR se utilizan como identificadores físicos únicos que vinculan productos físicos con sus registros digitales, permitiendo a los usuarios acceder información del producto mediante el escaneo del código. (p. 23, 28, 29)
red descentralizada	Una red descentralizada es una arquitectura de comunicación donde no existe un punto central de control o falla, sino que el poder y la responsabilidad se distribuyen entre múltiples nodos participantes. A diferencia de los modelos centralizados donde un servidor central coordina las operaciones, en una red descentralizada cada nodo puede operar de forma independiente mientras mantiene sincronización con el resto de la red. <i>Blockchain</i> es un ejemplo paradigmático de red descentralizada, donde múltiples participantes mantienen copias de la cadena de bloques y validan transacciones colectivamente. (p. 10, 11)
RFID	<i>Radio Frequency Identification</i> (RFID) es una tecnología de identificación por radiofrecuencia que utiliza campos electromagnéticos para identificar y rastrear automáticamente etiquetas adheridas a objetos. Las etiquetas RFID contienen información almacenada electrónicamente que puede ser leída a corta distancia sin necesidad de contacto directo. En sistemas de trazabilidad de cadenas de suministro, RFID permite el seguimiento automatizado de productos a lo largo de su ciclo de vida, facilitando la implementación de sistemas de monitoreo integral en tiempo real. (p. 1, 22, 23, 60)

software	El software es un conjunto de instrucciones, datos o programas utilizados para operar computadoras y ejecutar tareas específicas. Incluye aplicaciones, sistemas operativos, utilidades y herramientas de desarrollo que permiten a los usuarios interactuar con el hardware y realizar diversas funciones. (p. 9, 32, 33, 38, 56, 75)
sostenibilidad	La sostenibilidad es la capacidad de satisfacer las necesidades del presente sin comprometer la capacidad de las futuras generaciones para satisfacer sus propias necesidades. En el contexto empresarial e industrial, implica la implementación de prácticas que equilibren el crecimiento económico, la protección ambiental y el bienestar social. (p. 5, 19, 21, 22, 27, 44, 61, 109)
token	Un <i>token</i> , en el contexto de <i>blockchain</i> , es una unidad de valor que representa un activo digital y está asociado a una plataforma <i>blockchain</i> en particular. Se suele utilizar como sinónimos de criptomonedas. Los tokens se pueden intercambiar, transferir, almacenar y utilizar en todas las aplicaciones construidas sobre una <i>blockchain</i> . Los tokens pueden ser fungibles o no fungibles, dependiendo de si son intercambiables entre sí o son únicos. (p. 60)
trazabilidad	La trazabilidad es la capacidad de identificar, registrar y seguir el historial, la aplicación o la ubicación de un producto, proceso o servicio a través de etapas identificadas de producción, procesamiento y distribución. En el contexto de economía circular, la trazabilidad permite monitorear el ciclo de vida completo de los productos desde su origen hasta su reutilización o reciclaje, proporcionando transparencia sobre el origen, procesamiento, transporte y destino final de los materiales, facilitando la implementación de prácticas sostenibles y la verificación de compromisos ambientales. (p. 5, 15, 16, 20, 21, 27, 42, 47, 49, 58, 62, 66, 78, 91, 103, 110, 113, 126)
URL	Una URL (<i>Uniform Resource Locator</i>) es una dirección web que especifica la ubicación de un recurso en Internet y el protocolo utilizado para acceder a él. Las URLs incluyen componentes como el protocolo (HTTP/HTTPS), nombre del dominio, puerto, ruta (<i>endpoint</i>) y parámetros de consulta. En aplicaciones web, las URL se utilizan para acceder al <i>frontend</i> desde el cliente y para realizar solicitudes de recursos a APIs. (p. 80)

SIGLAS

NPoS Nominated Proof of Stake. (*p. 60*)

ODS Objetivos de Desarrollo Sostenible. (*p. 18, 19*)

PoA Proof of Authority. (*p. 11, 60*)

PoS Proof of Stake. (*p. 10, 11, 13, 59–61*)

PoW Proof of Work. (*p. 10, 11, 13, 59–61*)

REST Representational State Transfer. (*p. 63*)

RVM Reverse Vending Machine. (*p. 110, 111*)

UAT User Acceptance Testing. (*p. 89*)

UI User Interface. (*p. 90*)

UML Unified Modeling Language. (*p. 67, 68*)

UX User Experience. (*p. 90*)

