ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO Y SIMULACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS Y TRACKING DEL PROCESO PRODUCTIVO DEL CAFÉ

SISTEMA DE TRACKING BASADO EN BLOCKCHAIN DEL PROCESO PRODUCTIVO DEL CAFÉ

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y AUTOMATIZACIÓN

NATHALIA ISABEL BARREIROS FLORES

nathalia.barreiros@epn.edu.ec

DIRECTOR: ING. GEOVANNY DANILO CHÁVEZ GARCÍA, PhD.

danilo.chavez@epn.edu.ec

DMQ, octubre 2022

CERTIFICACIONES

Yo, Nathalia Isabel Barreiros Flores declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

NATHALIA ISABEL BARREIROS FLORES

Certifico que el presente trabajo de integración curricular fue desarrollado por Nathalia Isabel Barreiros Flores, bajo mi supervisión.

ING. GEOVANNY DANILO CHÁVEZ GARCÍA, PhD.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Nathalia Isabel Barreiros Flores

Ing. Geovanny Danilo Chávez García, PhD.

DEDICATORIA

A mis padres Yolanda y Rodrigo, por su amor incondicional y por ser el motor y soporte durante cada etapa de mi vida. A mis hermanos Ángel y José, por ser mi motivación y mi guía.

A todas las personas soñadoras, curiosas y disruptivas que de alguna forma inspiraron este trabajo.

AGRADECIMIENTO

A mis padres y hermanos, por ayudarme a alcanzar mis sueños y por todo su apoyo. A mi madre Yolanda, por ser mi amiga incondicional, mi motivación y mi ejemplo de fortaleza, gracias por todo el cariño, la comprensión y los consejos que me has brindado. A mi padre Rodrigo, por ser mi guía y ejemplo de perseverancia. A mi hermano Ángel, por estar pendiente de mí, por su tiempo y apoyo. A mi hermano José, por ser esa persona especial que me enseñó a maravillarme por tantas cosas, a ver el mundo de formas únicas y con infinitas posibilidades. Gracias a los cuatro, por las risas, las experiencias y los consejos, pero sobre todo por estar conmigo en los momentos difíciles y por creer en mí. Los amo.

Al Dr. Danilo Chávez, por su disposición, motivación, paciencia, confianza y guía durante la realización y culminación del presente proyecto.

Al Ing. Ricardo Villaroel, por el interés, la colaboración, el conocimiento y el tiempo prestado en el desarrollo de este proyecto.

A la Escuela Politécnica Nacional y a sus docentes, que me han permitido formarme profesionalmente.

A todos mis amig@s, por acompañarme y compartir conmigo a lo largo de mi vida universitaria, especialmente a Alex, Darwin y Jeff. Finalmente, quiero agradecer a todas las personas que de alguna manera me han aportado y permitido culminar satisfactoriamente esta etapa.

Nathalia

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN X
ABSTRACT XI
1 INTRODUCCIÓN 1
1.1 OBJETIVO GENERAL 2
1.2 OBJETIVOS ESPECÍFICOS 2
1.3 ALCANCE 2
1.4 MARCO TEÓRICO 3
1.4.1 CADENAS DE SUMINISTRO 3
1.4.1.1 Cadena de Suministro de Café 4
1.4.2 DESAFÍOS DE LAS CADENAS DE SUMINISTRO 5
1.4.3 TRAZABILIDAD DE LA CADENA DE SUMINISTRO 6
1.4.3.1 Requisitos de un Sistema de Trazabilidad 6
1.4.3.2 Tecnologías para la Transmisión del Identificador de Producto en Sistemas de Trazabilidad 8
1.4.4 TECNOLOGÍA BLOCKCHAIN 9
1.4.4.1 Proceso de Transacción en Blockchain 9
1.4.4.2 Proceso de Validación en Blockchain 10
1.4.4.3 Contratos Inteligentes 11
1.4.4.4 Blockchain en la Trazabilidad de Cadenas de Suministro 12
2 METODOLOGÍA 12
2.1 REQUERIMIENTOS PARA EL SISTEMA DE TRACKING UTILIZANDO LA TECNOLOGÍA BLOCKCHAIN 13
2.2 DEFINICIÓN DE LAS VARIABLES A RASTREAR 15

2.3 ARQUITECTURA DEL SISTEMA DE TRACKING 16
2.4 IMPLEMENTACIÓN DE LOS CONTRATOS INTELIGENTES 17
2.4.1 CADENA DE BLOQUES DE ETHEREUM 18
2.4.2 CONFIGURACIÓN DE LA CADENA DE BLOQUES TESTNET DE RINKEBY EN LA BILLETERA WEB DE METAMASK 21
2.4.3 CONFIGURACIÓN DEL ENTORNO DE DESARROLLO PARA LOS CONTRATOS INTELIGENTES CON TRUFFLE 21
2.4.4 ESTRUCTURA Y DISEÑO DE LOS CONTRATOS INTELIGENTES 22
2.4.5 CONFIGURACIÓN DE LA BLOCKCHAIN LOCAL BASADA EN ETHEREUM CON GANACHE 28
2.4.6 DEFINICIÓN DEL ARCHIVO DE CONFIGURACIÓN DE TRUFFLE 29
2.4.7 CONFIGURACIÓN DEL SCRIPT DE MIGRACIÓN 31
2.4.8 PRUEBAS DE LOS CONTRATOS INTELIGENTES CON TRUFFLE Y GANACHE 31
2.4.9 DESPLIEGUE DE LOS CONTRATOS INTELIGENTES EN LAS CADENAS DE BLOQUES 32
2.4.10 VERIFICACIÓN DE LOS CONTRATOS INTELIGENTES 32
2.5 APLICACIÓN WEB DEL SISTEMA DE TRACKING 33
2.5.1 CARACTERÍSTICAS DE IMPLEMENTACIÓN DE LA APLICACIÓN WEB 33
2.5.1.1 Conexión de la Dapp con la Blockchain a través de MetaMask 33
2.5.1.2 Escritura y Lectura de Datos en los Contratos Inteligentes 35
2.5.1.3 Escucha de Eventos 35
2.5.1.4 IPFS para el Almacenamiento de Imágenes 36
2.5.1.5 Código QR para Transmisión del Identificador de Lote 36
2.5.1.6 Google Maps API para los Datos de Ubicación 37
2.5.2 MODELO DE LA APLICACIÓN WEB Y FLUJOS DE PROCESO DE USO 37
2.5.3 DESPLIEGUE DE LA APLICACIÓN EN VERCEL 40
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 40
3.1 RESULTADOS 40

3.1.1 CONTRATOS INTELIGENTES VERIFICADOS DESPLEGADOS EN LA BLOCKCHAIN TESTNET RINKEBY 40
3.1.2 PRUEBAS REALIZADAS DEL SISTEMA DE TRACKING CON LA APLICACIÓN WEB 41
3.1.2.1 Inicio de Sesión en la Aplicación de Trazabilidad CoffeeTrack 41
3.1.2.2 Prueba del Registro de Granja y Creación de Nuevo ID de Lote 42
3.1.2.3 Prueba del Registro de Cosecha 44
3.1.2.4 Prueba del Registro de Procesado 46
3.1.2.5 Prueba del Registro de Catación 48
3.1.2.6 Prueba del Registro de Venta del Lote de Café 49
3.1.2.7 Prueba del Registro de Bodegaje 50
3.1.2.8 Prueba del Registro de Transporte a Empacador 50
3.1.2.9 Prueba del Registro de Empacado 51
3.1.2.10 Prueba del Registro de Transporte a Retailer 51
3.1.2.11 Prueba del Registro de Comercialización en Retailer 51
3.1.2.12 Prueba del Registro de Usuarios 52
3.1.2.13 Prueba de la Actualización de Perfil 53
3.1.2.14 Prueba de la Consulta de Trazabilidad de una Funda de Café de Especialidad a través de su Código QR 53
3.2 CONCLUSIONES 55
3.3 RECOMENDACIONES 57
4 REFERENCIAS BIBLIOGRÁFICAS 58
5 ANEXOS 61
ANEXO I 64
ANEXO II 66
ANEXO III 67
ANEXO IV 69
ANEXO V 71
ANEXO VI 73

ANEXO VII....74
ANEXO VIII....75
ANEXO IX....76
ANEXO X....77
ANEXO XI....78
ANEXO XII....79
ANEXO XIII....80
ANEXO XIV....81
ANEXO XV....82
ANEXO XVI....83
ANEXO XVII....84
ANEXO XVIII....85
ANEXO XIX....86
ANEXO XX....91
ANEXO XXI....92
ANEXO XXII....95
ANEXO XXIII....98
ANEXO XXIV....99
ANEXO XXV....100
ANEXO XXVI....102
ANEXO XXVII....104
ANEXO XXVIII....113
ANEXO XXIX....117
ANEXO XXX....118
ANEXO XXXI....119
ANEXO XXXII....120
ANEXO XXXIII....122
ANEXO XXXIV....123
ANEXO XXXV....124

ANEXO XXXVI.....126
ANEXO XXXVII.....128
ANEXO XXXVIII.....129
ANEXO XXXIX.....132
ANEXO XL.....136
ANEXO XLI.....175

RESUMEN

En el presente proyecto de titulación se detalla la aplicación de la tecnología blockchain como alternativa para el registro y almacenamiento de información de manera descentralizada, para el diseño y simulación de un sistema de trazabilidad de la cadena de suministro agrícola del café de especialidad. Se consideró una cadena de suministro con un flujo lineal, teniendo al lote de café como unidad trazable (TRU). De manera específica, se desarrollaron los contratos inteligentes que permiten ejecutar la lógica de funcionamiento del sistema de trazabilidad para llevar a cabo el registro de información en la blockchain, de las diferentes etapas de intervención en la cadena de valor del lote de café y de los usuarios modificadores, así como la lectura de los datos para ser mostrados a los usuarios. Los contratos inteligentes fueron codificados en el lenguaje de Solidity y desplegados en la testnet de Rinkeby. Para llevar a cabo la interacción de los usuarios con los contratos inteligentes alojados en la blockchain, se creó una aplicación web denominada CoffeeTrack. Esta permite que los usuarios modificadores inicien sesión a través de su cuenta de billetera digital MetaMask y accedan a su panel de control para realizar el registro de información de un lote de café específico. Por otro lado, la aplicación web también presenta una sección dirigida para los consumidores, la cual permite escanear el código QR presente en la funda de su café para conocer su procedencia y datos de trazabilidad a lo largo de su cadena de valor.

PALABRAS CLAVE: Blockchain, contratos inteligentes, ecosistema de Ethereum, testnet, red peer-to-peer, sistemas descentralizados, Dapps, trazabilidad, cadena de suministro del café, billetera digital, Ethers.js, IPFS, proveedor de nodos.

ABSTRACT

In this capstone project, the application of blockchain technology is presented as an alternative for the registration and storage of information in a decentralized fashion, for the design and simulation of a traceability system for the agricultural supply chain of specialty coffee. A supply chain with a linear flow was considered, with the coffee batch as the traceable resource unit (TRU). Smart contracts register information about the different intervention stages throughout the value chain of a specific coffee batch, as well as information about users who execute transformation processes. Smart contracts also allow the reading of the registered data in the system to be shown to users. The smart contracts were coded in Solidity language and deployed on a Rinkeby testnet. User interaction with smart contracts hosted on the blockchain was based on a web application called CoffeeTrack. This allows involved users to log in through their MetaMask digital wallet account and access their dashboard to register information for a specific coffee batch. The web application also provides a section aimed at consumers, which allows them to scan the QR code on the cover of their coffee to find out its origin and traceability data throughout the value chain.

KEYWORDS: Blockchain, smart contracts, Ethereum ecosystem, testnet, peer-to-peer network, decentralized systems, Dapps, traceability, coffee supply chain, digital wallet, Ethers.js, IPFS, nodes provider.

1 INTRODUCCIÓN

Actualmente las cadenas de suministro agrícola se han vuelto más complejas, añadiendo un mayor número de participantes desde el manejo de la materia prima hasta llegar al producto final, lo que ha llevado a la necesidad de dar seguimiento a los productos a lo largo de su cadena de valor. Otra razón que incentiva la adopción de sistemas de trazabilidad o seguimiento de productos en las cadenas de suministro agrícola es la creciente tendencia por parte de los consumidores de conocer el origen e idoneidad de sus productos, este incluso es un factor decisivo al momento de llevar a cabo una compra para algunos consumidores. En la línea de la cadena de suministro agrícola del café, se estima que más de cincuenta países en desarrollo tanto en América Latina, África y Asia producen café, siendo la fuente de ingresos para alrededor de 25 millones de familias en el mundo [1]. En específico en Ecuador, existen alrededor de 46 mil productores ecuatorianos que dependen de la comercialización de café [2]. El nicho de mercado para el café especial de alta calidad es más lucrativo, sin embargo, el consumo de este tipo de café depende altamente de la apropiación del producto por parte de los consumidores, por lo que conocer la procedencia en este tipo de café es un requisito para su adopción.

Por otro lado, los sistemas de trazabilidad actuales empleados se basan en una arquitectura centralizada y en algunos casos, gestionada por terceros. Esto tiene como resultado que sean sistemas susceptibles a ataques de un solo nodo y de manipulación de datos [3]. De esta manera, una infraestructura centralizada presenta problemas en cuanto a transparencia, posible bloqueo de datos y dificultad en la disponibilidad y auditabilidad de los datos. Blockchain al ser una tecnología descentralizada y distribuida para almacenar datos de manera inmutable, se muestra como una solución para los problemas que presentan los sistemas de trazabilidad tradicionales. Blockchain también permite el acceso a los datos de manera inmediata, transparente y compartida, creando confianza en la fuente de información por parte de los consumidores como de los usuarios modificadores de una cadena de suministro agrícola.

El proyecto presentado a continuación, demuestra el uso de la tecnología blockchain para simular un sistema de trazabilidad para la cadena de suministro agrícola del café de especialidad. Para este fin, se lleva a cabo la programación de los contratos inteligentes que son alojados en la blockchain testnet de Rinkeby, los cuales contiene toda la lógica de funcionamiento del sistema de trazabilidad cuya información ingresada está enfocada en garantizar la confianza por parte del consumidor en su café de especialidad. Además, para permitir que los usuarios modificadores del café ingresen los datos que serán almacenados

en la blockchain, se crea una aplicación web, la cual, a través de la billetera digital MetaMask instalada como extensión en el navegador, permite a los usuarios interactuar con la blockchain. La aplicación web también permite que los usuarios consumidores escaneen el código QR presente en la funda de su café para consultar la trazabilidad del producto.

1.1 OBJETIVO GENERAL

Diseñar y simular un sistema de tracking basado en blockchain dentro del proceso productivo del café de especialidad en Ecuador.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar una revisión bibliográfica referente a las cadenas de suministro, a la trazabilidad en cadenas de suministro y sobre la tecnología blockchain.
2. Identificar los actores y variables a ser rastreados dentro de la cadena de suministro del café de especialidad, desde el punto de vista de generar confianza en el consumidor por sobre su producto.
3. Desarrollar la lógica de funcionamiento del sistema de seguimiento a través de contratos inteligentes.
4. Desplegar los contratos inteligentes en una blockchain de prueba o testnet basada en el ecosistema de la blockchain de Ethereum.
5. Desarrollar la aplicación web para interactuar con el sistema desarrollado.
6. Realizar las pruebas necesarias que permitan verificar el correcto funcionamiento de todo el sistema de tracking (contratos inteligentes y aplicación web).

1.3 ALCANCE

Para abordar el alcance del componente se tendrán en cuenta las siguientes fases:

- ***Fase de revisión bibliográfica:*** Se estudia el funcionamiento de la tecnología blockchain, así como las características de los sistemas de trazabilidad y los desafíos que presentan las cadenas de suministro, con el fin de comprender cómo se enlaza la tecnología blockchain con los sistemas de seguimiento o tracking. Así también, se analiza la cadena de suministro para el café de especialidad en Ecuador.

- ***Fase de diseño:*** Se identifican los actores y variables relevantes a la cadena de suministro del café de especialidad, teniendo como finalidad generar confianza en el producto a ser adquirido por un consumidor. Se define la lógica de funcionamiento que llevará el sistema de seguimiento y las características de implementación de la aplicación web que permitirán interactuar con el sistema de seguimiento.

- ***Fase de implementación:*** Se implementa la lógica de funcionamiento para el sistema de seguimiento en contratos inteligentes utilizando Solidity como lenguaje de programación. Se despliegan los contratos inteligentes en una blockchain de prueba o testnet fundamentada en el ecosistema de la blockchain de Ethereum. Se desarrolla la aplicación web utilizando React como librería de desarrollo y Ethers.js como librería para conectarse con el ecosistema de la cadena de bloques de Ethereum.
- ***Fase de pruebas:*** Se realizan pruebas de los contratos inteligentes antes de ser desplegados en la blockchain. Se comprueba el funcionamiento integral del sistema empleando la aplicación web desarrollada.

Con las fases antes definidas, se establece que los productos que se entregarán del componente corresponden a una aplicación web para interactuar con el sistema desarrollado por medio de una blockchain y los contratos inteligentes correspondientes al sistema de tracking para la cadena de suministro del café de especialidad desplegados en una blockchain.

1.4 MARCO TEÓRICO

En esta sección se detallan los conceptos teóricos sobre las cadenas de suministro, la trazabilidad en cadenas de suministro y la tecnología blockchain como alternativa para trazabilidad de productos, además de otra información relevante para el entendimiento del presente trabajo realizado.

1.4.1 CADENAS DE SUMINISTRO

Se puede definir a una cadena de suministro como un conjunto de participantes que se encuentran interconectados con el objetivo de agregar valor a un flujo de insumos que han sido transformados desde su fuente de origen hasta los productos o servicios finales que llegan a los consumidores [4]. Típicamente una cadena de suministro puede incluir como actores a los proveedores de materias primas, los productores, procesadores, transportistas de mercancías, intermediarios y en general, a cualquier participante que de alguna manera añada valor a un producto o servicio hasta llegar a las manos de los usuarios finales. Un flujo genérico de una cadena de suministro se muestra a continuación.

Figura 1.1. Flujo genérico de una cadena de suministro

El movimiento de los insumos a través de una cadena de suministro lleva consigo información relevante al producto en el estado en el que se encuentre, dicha información es importante para una gestión eficiente de la cadena de suministro. Por otro lado, el manejo adecuado de una cadena de suministro es un factor relevante para mejorar la industria, así como para fidelizar a los consumidores [5].

1.4.1.1 Cadena de Suministro de Café

El café es parte de la identidad del Ecuador y debido a la diversidad de ecosistemas en el país, la producción de café se extiende por todo el territorio ecuatoriano, pasando por la tradicional provincia de Loja, las Islas Galápagos, zonas del Carchi y Santo Domingo, etc. Se estima que cerca de 46 mil productores ecuatorianos dependen del café, cultivo al que destinan alrededor de 96.312 hectáreas [2].

Por otro lado, la cadena de suministro de café puede ser muy simple o más compleja, de igual manera el número de participantes puede variar ampliamente. Para efectos del desarrollo del presente proyecto se hará uso del modelo de comercio directo, cuyo producto es el café de especialidad. El flujo de la cadena de suministro y sus actores o participantes para este caso, se describen en la Tabla 1.1.

Tabla 1.1. Participantes de la cadena de suministro del café de especialidad

Actor	Detalle del Proceso
Agricultor	- Se encarga de cultivar el cafeto y de recolectar las cerezas del café cuando llegue la temporada de cosecha. - A continuación, lleva sus granos a un procesador.
Procesador	- Recibe el café del agricultor. - Realiza el procesamiento del grano, lo que consiste en: lavar, secar, trillar, clasificar, tostar y moler los granos de café. - A continuación, devuelve el grano al agricultor
Catador	- Se encarga de calificar el grano del agricultor en función de los parámetros de la granja de cultivo, así como del procesado realizado. La catación es requerida ya que se trata de café de especialidad.
Agricultor	- Recibe el grano procesado y la puntuación emitida por el catador. - A continuación, vende el grano a una bodega.
Bodega	- Compra el grano del agricultor y lo almacena bajo condiciones adecuadas hasta conseguir un comprador, que en este caso correspondería a una empresa empacadora. - A continuación, envía el grano a una empresa empacadora.
Transporte hacia empacador	- Se encarga del correcto transporte del grano desde la bodega hasta la empresa empacadora.
Empacador	- Recibe el grano de la bodega por medio del transportista. - Realiza el empacado del grano en función del requerimiento del retailer. - Envía el grano empacado al retailer.
Transporte hacia retailer	- Se encarga del correcto transporte del grano desde el empacador hasta el retailer.

Retailer	- Recibe el grano del empacador por medio del transportista. - El retailer consiste en la bodega y en los puntos de venta final. - Existe un transporte desde la bodega hacia los puntos de venta final. - En los puntos de venta final es donde el consumidor de café de especialidad compra su producto.

Cabe mencionar que el transporte se encuentra presente a lo largo de toda la cadena de suministro de café. Sin embargo, para fines del presente trabajo se consideró los puntos de transporte más significativos durante la cadena debido a que en muchas ocasiones durante los primeros procesos, el transporte es llevado a cabo por el mismo participante de la red, ya se el agricultor o el procesador.

1.4.2 DESAFÍOS DE LAS CADENAS DE SUMINISTRO

La gestión de las cadenas de suministro es una tarea compleja debido a la interconexión de la información y las partes, presentando así los desafíos siguientes.

Tabla 1.2. Desafíos que presentan las cadenas de suministro

Desafío	Explicación
Transparencia	- Existen productos cuya procedencia representa una característica esencial para determinar la decisión de compra de un consumidor. - Podemos pensar en el sector pesquero, el cual presenta un alto porcentaje de actividades no reguladas e ilegales [6], donde los consumidores que defienden la pesca sostenible requieren conocer el origen de su producto. - A esto se le suma la dificultad para distinguir entre diferentes alternativas para un mismo producto, en donde el hecho de no determinar dónde se fabrica u obtiene un producto, conlleva a la introducción de productos ilegales y dañinos en el mercado, también puede tener consecuencias perjudiciales sobre la salud de los consumidores y además degrada la industria [6].
Trazabilidad	- Desarrollar una cadena de suministro eficiente implica poder localizar los productos en un momento específico. - Rastrear la procedencia y trayectoria de los productos es un tema crítico porque permite encontrar la fuente de productos con fallas o dañinos [7]. - Aunque las partes de una cadena de suministro están interconectadas, no siempre existe un flujo bidireccional y transparente de información entre estas, poniendo en tela de duda la seguridad alimentaria y pudiendo generar pérdidas económicas cuando existe cuestionamientos sobre la procedencia y el estado de ciertos productos [7].
Confianza	- El factor de la confianza juega un papel protagonista en las cadenas de suministro, pudiendo causar pérdidas económicas y bajo rendimiento como resultados de la falta de confianza [8]. - La confianza en cualquier industria agiliza y flexibiliza los procesos. - En una cadena de suministro todas las partes deberían poder confiar entre sí, desde el fabricante con su proveedor hasta el

	consumidor depositando su confianza en su minorista y por ende en toda la cadena de suministro de su producto.
Descentralización y almacenamiento de datos	- La mayoría de las cadenas de suministro presentan una gestión centralizada, lo que conlleva a un proceso de toma de decisiones, flujo de relaciones de producción e intercambio y ganancias centralizadas. - Las arquitecturas centralizadas en cuanto al manejo y almacenamiento de datos son susceptibles a ataques de un solo nodo, manipulación de datos y divulgación de información [3], [9].

1.4.3 TRAZABILIDAD DE LA CADENA DE SUMINISTRO

La trazabilidad puede definirse como la capacidad de acceder a información relacionada con el ciclo de vida de un producto a lo largo de una red de suministro, mediante el uso de identificadores registrados [10]. Ya que la trazabilidad requiere del registro de todas las transformaciones que se realice a un artículo en la cadena, esto significa que la trazabilidad permite rastrear un producto hacia atrás o hacia adelante de su ciclo de vida.

Cuando se habla de trazabilidad es común hacer alusión a los términos de tracking y tracing. Por un lado, el tracking o seguimiento hace referencia al proceso informativo en el cual un producto es seguido a través de la cadena de suministro, manteniendo registros o actualizaciones en cada etapa de modificación. El tracing o rastreo se define como la capacidad de reconstruir el histórico de un artículo, y se requiere identificar el origen de este [11]. Sin embargo, los términos de tracking, tracing y trazabilidad en la mayoría de los casos se usan indistintamente ya que todos involucran el seguimiento del movimiento de productos en una red de suministro [12]. Por este motivo, el presente trabajo hace uso de los tres términos de manera unívoca.

En un sistema de tracking se puede identificar como entidades primordiales al producto y a la actividad [13]. Para referirnos a la unidad que se desea rastrear se usa el término de Unidad de Recursos Trazables (TRU, por sus siglas en inglés) [14]. Un tipo de unidad trazable es el lote el cual hace referencia a una cantidad que pasa por un mismo proceso [14]. En la industria de alimentos, un lote de producción es lo que se produce de un determinado producto en una unidad de tiempo y lleva un identificador único [15]. Un ejemplo de tracking en una cadena de alimentos se muestra en la Figura 1.2.

1.4.3.1 Requisitos de un Sistema de Trazabilidad

Los sistemas de tracking deben cumplir con las siguientes características básicas [16]:

- Identificación única del producto y sus componentes.
- Información de cuándo y dónde se ha movido el producto.

- Vinculación de la unidad trazable con sus movimientos por medio de un mecanismo que facilite el acceso e ingreso de datos.

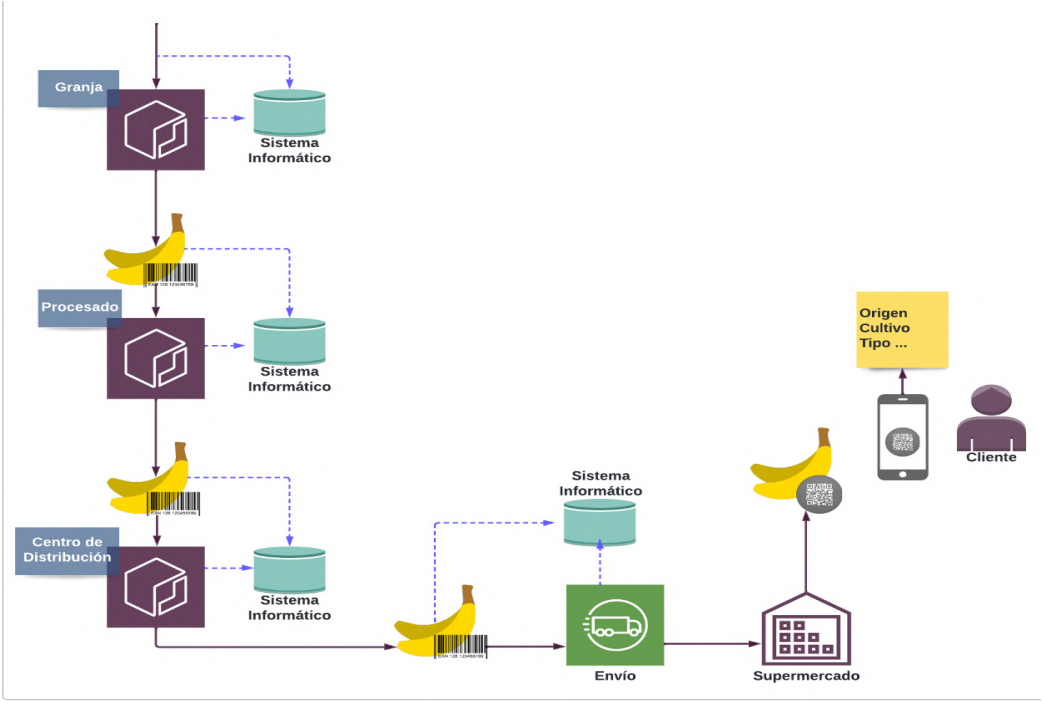


Figura 1.2. Sistema de trazabilidad de una cadena de suministro de alimentos

Los sistemas de trazabilidad suelen ser complejos debido a que requieren de la selección de una variedad de parámetros para ser rastreados. La decisión e identificación de parámetros, así como el diseño del sistema de tracking para un producto dependen de la cadena de suministro y su estructura, del objetivo para el cuál se desee implementar la trazabilidad, de la capacidad tecnológica y económica que se tenga, de la calidad y el grado de precisión que se desee implementar en el sistema, etc. Independientemente del diseño del sistema y las variables seleccionadas, un sistema de tracking debería ser capaz de rastrear el historial del producto a través de su red de suministro completa [12].

Un sistema de tracking debe construirse en base a los siguientes cuatro pilares [17]:

- ***Identificación de productos:*** Cada unidad trazable debe estar acuñada a un identificador único.
- ***Datos para rastrear:*** Se debe elegir el tipo de información que el sistema debe soportar, así como los niveles de confidencialidad de la información rastreada, todo esto en función de los limitantes y capacidades de los que se disponga.
- ***Enrutamiento del producto:*** La trazabilidad completa requiere cubrir toda la red de suministro, la conexión entre las partes es requerido hasta llegar al producto final.

- ***Herramientas de trazabilidad:*** Las herramientas que emplee el sistema de tracking se definen en función de los requisitos, costos y de la compatibilidad con la cadena en cuestión. Las soluciones más comunes para capturar datos son: registros en papel, identificación por radio frecuencia (RFID), códigos de barras, sistemas electrónicos, etc.

1.4.3.2 Tecnologías para la Transmisión del Identificador de Producto en Sistemas de Trazabilidad

Algunas de las tecnologías ampliamente empleadas para la transmisión del identificador de un producto en sistemas de tracking a lo largo de una red de suministro son: los códigos de barras, los códigos QR y la tecnología RFID. A continuación, se muestra una comparación entre estas alternativas.

Tabla 1.3. Comparación de tecnologías empleadas en la transmisión del identificador de producto en cadenas de suministro

Parámetro	**RFID** [18], [19]	**Código de Barras** [7], [18]	**Código de Respuesta Rápida (QR)** [7], [20]
Tecnología	Radiofrecuencia	Óptica	Óptica
Línea de visión	No requerida	Requerida	Requerida, con tan solo un 30% del código puede ser identificado correctamente
Tipo de interacción	Lectura y escritura	Solo de lectura	Solo de lectura
Interferencia	Metales y líquidos interfieren con algunas frecuencias RFID	Suciedad y roturas en el código impreso	Suciedad y roturas en el código impreso
Capacidad de almacenamiento	2000 bytes de datos	Hasta 24 caracteres alfanuméricos (100 bytes)	Hasta 3000 bytes, la información puede vincularse a sitios web, aplicaciones, videos, texto informativo, etc.
Dispositivos requeridos	Tag RFID y su lector	Lector láser o smartphone para la lectura	Smartphone para lectura del código
Intervención del trabajador	Automatizado, en algunos casos requerida	Requerida	Requerida
Rango de escaneo	Alto	Bajo	Alto
Costo	Medio-Alto	Bajo	Bajo

1.4.4 TECNOLOGÍA BLOCKCHAIN

La tecnología de la cadena de bloques o blockchain hace referencia a una red peer-to-peer que tiene la capacidad de almacenar información en una cadena cronológica de bloques, que es actualizada continuamente [21]. La cadena de bloques puede ser vista como un libro mayor o registro digital replicado, distribuido e inmutable que brinda la posibilidad de realizar transacciones seguras entre dos participantes sin autoridad central [6].

La cadena de bloques tiene bases en la criptografía por medio del uso de funciones hash. Una función hash es un algoritmo matemático que permite transformar datos de entrada de cualquier longitud, en una salida en bits con una longitud fija [22]. A partir de una salida, determinar la entrada que generó esta salida es estadísticamente improbable. Además, estas funciones son deterministas, lo que significa que para una entrada específica se tendrá siempre la misma salida de manera consistente [23].

El funcionamiento de la cadena de bloques se lleva a cabo a través del proceso de transacciones y el proceso de validación de bloques.

1.4.4.1 Proceso de Transacción en Blockchain

El proceso de transacción peer-to-peer se fundamenta en los siguientes algoritmos [22]:

- ***Algoritmo de generación de claves:*** Genera de manera simultánea una clave de firma privada y una clave de verificación pública. El par de claves que se generan guardan una relación matemática entre ambas y a partir de la clave privada no puede derivarse la clave pública.
- ***Algoritmo de firma:*** Procesa el mensaje y vincula la clave privada del remitente a dicho mensaje.
- ***Algoritmo de verificación de firma:*** Por medio de la clave pública se garantiza que se empleó la clave privada correspondiente para firmar el mensaje encriptado por la función hashing empleada.

Pensemos que un participante A desea enviar información a un participante B a través de una blockchain, primero se codifica la información con la función de hashing que utilice la blockchain y con la clave privada del propietario de la información, se crea una firma digital única basada en estos datos [22]. A continuación, se envía la información junto con la firma digital y la clave pública del propietario. Para evaluar si el remitente del mensaje es el propietario legítimo de la información, el destinatario procesa la información recibida con la misma función hashing que utilizó el remitente. Por otro lado, se emplea la clave pública

del remitente para descifrar la firma digital y obtener la información encriptada enviada. Si los dos resultados anteriores son iguales entonces se verifica que el remitente es el propietario legítimo de la información [17]. Esta solución permite que los participantes de la red puedan realizar transacciones sin importar el grado de confianza que tengan entre sí, además permite no depender de terceros.

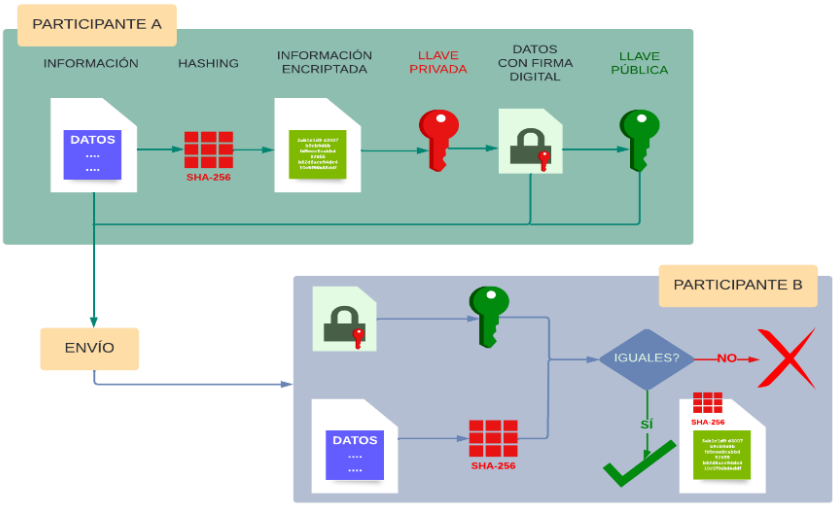


Figura 1.3. Proceso de transacción en blockchain

1.4.4.2 Proceso de Validación en Blockchain

Las transacciones llevadas a cabo en la blockchain deben ser validadas por la red antes de ser registradas en el libro mayor que representa la cadena de bloques [17]. Las cadenas de bloques tienen dos tipos de nodos. Los nodos livianos que se conectan a la blockchain por medio de clientes de billeteras digitales, los cuales realizan, verifican y propagan transacciones en la red. Los nodos completos que tienen la capacidad de ejecutar el software central de la cadena de bloques y poseen una copia completa y actualizada de toda la red, son los encargados de validar las transacciones, que consiste en agrupar las transacciones en un nuevo bloque y crear un hash correspondiente a este bloque para registrarlo en la red por medio de un algoritmo de consenso entre los nodos [22].

Los algoritmos de consenso buscan definir qué nodo realizará la validación del siguiente bloque con el fin de garantizar que no se corrompan los datos a procesar y se garantice la descentralización de la red.

Uno de los algoritmos de consenso más comunes es el de la prueba de trabajo (PoW, por sus siglas en inglés). Este garantiza la legitimidad de la validación de datos para la creación de un nuevo bloque por medio de la competencia entre nodos para resolver un acertijo matemático que requiere de altas capacidades de procesamiento y que debe ser resuelto en el menor tiempo posible para ganar la competencia, el nodo ganador del acertijo tiene

una recompensa económica con la criptomoneda que maneje la red, esto con el fin de incentivar la honestidad en el proceso [22]. El acertijo matemático a resolver consiste en obtener el nuevo hash de encabezado para ese bloque de manera que sea precedido por una cierta cantidad de ceros. La dificultad de la cantidad de ceros es establecida por la red de manera automática en cada validación. Debido a que el hash del bloque a validar está conformado por el hash del bloque anterior, la raíz de Merkle que contiene el hash que representa a todas las transacciones pasadas y un número generado aleatoriamente para cada bloque denominado “nonce”, el único valor que puede ser modificado para resolver el acertijo es el nonce [22].

Cuando un nodo encuentra el nonce que resuelve el acertijo, trasmite el nuevo bloque a toda la red y cada nodo completo hace una referencia cruzada a las partes constituyentes del árbol Merkle para corroborar que las transacciones validadas convergen en la raíz de Merkle [17]. El árbol de Merkle es una representación ramificada de cada hash de transacción y las uniones con otros hashes de otras transacciones, hasta conseguir la raíz de Merkle, que es el hash único de todas las transacciones realizadas en la red [22]. De esta manera, se confirma que el nodo que resolvió el acertijo no manipuló la información. Una vez confirmado la legitimidad del nuevo bloque, cada nodo agrega el nuevo bloque a su copia individual de la red, actualizando la cadena de bloques [24]. En el Anexo I, se visualiza un diagrama del proceso de validación en blockchain y de la estructura del árbol Merkle.

1.4.4.3 Contratos Inteligentes

Los contratos inteligentes son scripts de código que se ejecutan de manera automática y se almacenan en el libro mayor de blockchain en una dirección particular [22]. Los contratos inteligentes permiten codificar cualquier lógica de negocio relacionada con el intercambio de bienes o servicios, permitiendo la verificación y liquidación de promesas contractuales sin intervención humana [25]. Las obligaciones codificadas a través de contratos inteligentes presentan todos los beneficios de la cadena de bloques como: seguridad, verificabilidad, transparencia e inmutabilidad [22]. El flujo de funcionamiento de los contratos inteligentes involucra [25]:

Tabla 1.4. Flujo de funcionamiento de los contratos inteligentes

Parte	Implicación
Definición de términos del contrato	- Las partes establecen las condiciones, términos y activos involucrados. - Se crean scripts electrónicos que contienen lo acordado entre las partes bajo una lógica si-entonces (“if-then-else-what”) y son almacenados en la blockchain.

Eventos	- Cualquier información externa definida en el contrato desencadena su ejecución.
Ejecución y transferencia de valor	- Se ejecuta en la blockchain la lógica comercial relacionada con el evento desencadenado.
Asentamiento	- Los cambios realizados sobre el activo en cuestión se reflejan en el libro mayor de blockchain como un registro auditable e inmutable.

Figura 1.4. Funcionamiento de los contratos inteligentes en blockchain

1.4.4.4 Blockchain en la Trazabilidad de Cadenas de Suministro

La importancia que presenta la adopción de blockchain en cualquier red de suministro, se debe a que permite almacenar datos en un registro digital distribuido e inalterable, así como obtener estos datos de manera inmediata, compartida y transparente. Además, permite hacer seguimiento de pedidos, pagos, cuentas, detalles de producción, tiempos de entrega y cualquier tipo de información relevante a un proceso [25]. Por otro lado, los usuarios de esta tecnología comparten una única fuente fidedigna de información que se encuentra guardada de manera idéntica en varios servidores denominados nodos que están distribuidos por toda la red y que no puede ser alterada una vez registrada, en esta se pueden revisar todos los detalles de un proceso, generando confianza en los usuarios y consumidores, así como eficiencia durante el proceso. En este sentido, uno de los argumentos más convincentes para emplear blockchain en la trazabilidad de cadenas de suministro, es la necesidad del manejo y almacenamiento de datos a prueba de manipulaciones cuando los participantes de dicha cadena de suministro deben asegurarse de que los demás participantes con diferentes grados de confianza no alteren los datos después de haber sido ingresados al sistema. Finalmente, según los desafíos que enfrentan las cadenas de suministro descritos en el apartado 1.4.2, se puede observar que blockchain provee beneficios en los campos de: transparencia, auditabilidad, trazabilidad, seguridad ante la manipulación de información, inmutabilidad de la información registrada, confianza del consumidor en el proceso.

2 METODOLOGÍA

En el capítulo presentado a continuación, se describe la metodología empleada para el desarrollo del presente trabajo. El estudio llevado a cabo es de carácter explicativo,

adaptativo y exploratorio. Explicativo, con el fin de aproximarse a los conceptos fundamentales de la cadena de bloques y a la implementación de contratos inteligentes como herramienta para emplear blockchain en aplicaciones con un modelo descentralizado. Un estudio adaptativo y exploratorio para aplicar los principios y características de la tecnología blockchain en un sistema de tracking para la cadena de suministro de café de especialidad.

Las técnicas aplicadas para la elaboración del proyecto son la consulta, la observación y la experimentación. Para la recopilación de información sobre la tecnología blockchain y sistemas de seguimiento en cadenas de suministro, se recurre a la consulta de fuentes como revistas científicas, artículos técnicos, libros, páginas web, trabajos de titulación, etc. En cambio, la observación permite identificar las variables a dar seguimiento a lo largo de la cadena de suministro agrícola del café de especialidad según las diferentes etapas de intervención de este, desde el punto de vista de generar confianza en el consumidor sobre su producto. Finalmente, la experimentación facilita la simulación del sistema de tracking descentralizado con blockchain a través del desarrollo de contratos inteligentes y su interacción por medio de una página web y una billetera digital instalada en el navegador.

2.1 REQUERIMIENTOS PARA EL SISTEMA DE TRACKING UTILIZANDO LA TECNOLOGÍA BLOCKCHAIN

Los requerimientos que se mencionan a continuación se basan en el flujo de la cadena de suministro de café descrita en la Tabla 1.1. Cabe mencionar que se tomó como referencia una cadena de suministro de café de comercio directo, cuyo producto es el café de especialidad.

- La unidad trazable que se emplea en el sistema es el lote de café.
- Se requiere identificar el lote de café de manera única, antes de pasar por los diferentes procesos a lo largo de su cadena de suministro.
- Luego de haber identificado al lote de café, es necesario emplear alguna de las tecnologías descritas en el apartado 1.4.3.2, para dar seguimiento al lote de café por medio de su identificación única a lo largo de las etapas de su cadena de suministro.
- La cadena de suministro de café en cuestión presenta un flujo lineal, es decir que no se contemplarán bifurcaciones ni uniones del lote de café inicialmente identificado.
- Los procesos por los que pasa el café de especialidad a lo largo de su cadena de suministro son: cosecha, procesado, catación, venta de grano, bodegaje, transporte hacia empacador, empacado, transporte hacia retailer y comercialización en retailer.

- Los datos a dar seguimiento deben estar acorde con la finalidad de generar confianza en el consumidor sobre su producto, ya que este trabajo está pensado en la necesidad por parte de los consumidores de conocer la procedencia e idoneidad de sus alimentos, en este caso del café de especialidad.
- Empleo de una cadena de bloques pública y sin permisos. Esto debido a que una blockchain pública permite que cualquier usuario pueda unirse, así como desarrollar soluciones de manera directa sobre la red ya puesta en marcha, por lo que se aprovecha el poder computacional de los nodos peer-to-peer vigentes en la red, así como la arquitectura, la funcionalidad y el grado de desarrollo que disponga la red. Por otro lado, una cadena de bloques sin permisos permite que cualquiera pueda participar en el mecanismo de consenso para validar bloques y transacciones [26]. Además, las cadenas de bloques públicas y sin permisos ponen a disposición todas las características que brinda la tecnología blockchain en cuanto a transparencia, auditabilidad, descentralización, inmutabilidad, resistencia a la censura de datos, alta seguridad, etc. [17], [22].
- Empleo de contratos inteligentes para manejar toda la funcionalidad con respecto al seguimiento del café de especialidad a lo largo de su cadena de suministro, los cuales se ejecutan en una blockchain pública y sin permisos.
- Despliegue de los contratos inteligentes en una blockchain de prueba o testnet debido a que se trata de un prototipo y a que las interacciones con las cadenas de bloques públicas y sin permisos requieren del pago de una tarifa para realizar el ingreso de datos por medio de transacciones.
- Desarrollo de una aplicación web que permita interactuar visualmente con los contratos inteligentes ejecutados en la blockchain tanto para escribir los datos de seguimiento como para leer estos datos.
- Se requiere el uso de una billetera digital instalada en el navegador, por parte de todos los participantes que van a ingresar datos a lo largo de la cadena de suministro de café a través de la aplicación web desarrollada. Cabe mencionar que toda aplicación basada en blockchain requiere que los usuarios se conecten a la cadena de bloques por medio de un nodo y debido a que estos usuarios únicamente realizarán el ingreso de datos o transacciones, el nodo de conexión a la blockchain es un nodo liviano y no un nodo completo, como se mencionó en el apartado 1.4.4.2. Así, las billeteras digitales se encargan de manejar la conexión del nodo a la cadena de bloques, facilitando el uso de la tecnología blockchain a todos los usuarios.

- Se asume que todos los participantes de la cadena de suministro del café de especialidad desean contribuir activamente en el sistema de tracking desarrollado por medio del ingreso de los datos a través de la aplicación web desarrollada.

2.2 DEFINICIÓN DE LAS VARIABLES A RASTREAR

En base al estudio de los diferentes procesos llevados a cabo a lo largo de la cadena de suministro del café de especialidad en Ecuador, se ha determinado las variables a dar seguimiento, para lo cual se ha tenido en cuenta la premisa de que la visibilidad de un proceso productivo transparenta la confianza en el mismo. De esta manera, los consumidores tienen mayor seguridad de que su producto proviene de una fuente confiable, se incentiva la integridad en la producción y el trato justo, así como se fideliza a los consumidores, lo que conlleva a un beneficio económico para todos los participantes de la cadena de suministro.

Para este propósito, el sistema de tracking desarrollado da seguimiento a variables que permiten conocer el origen del café, la ubicación del café a lo largo de sus etapas de transformación, características que describen los procesos de transformación, así como el precio por unidad de peso que conllevan dichas intervenciones, teniendo como unidad trazable el lote de café. Las variables seleccionadas, según las etapas de transformación del café descritas en la Tabla 1.1 y una etapa inicial correspondiente al registro de la granja en el sistema de tracking, se muestran en la Tabla 2.1.

Tabla 2.1. Variables a dar seguimiento con el sistema de tracking

Etapa	Variables
Registro de la granja	- Número de registro - Nombre de la granja del cultivo de cafeto - Dirección de ubicación de la granja - Latitud y longitud de la ubicación de la granja
Cosecha	- Proveedor de la semilla de café - Tipo de semilla de café - Familia de la semilla de café - Fertilizantes utilizados en el cultivo - Fecha y hora de cosecha del lote de café - Porcentaje de humedad de los granos cosechados - Peso en kilogramos del lote cosechado
Procesado	- Dirección de ubicación del lugar de procesado - Latitud y longitud de la ubicación del lugar de procesado - Tipo de secado - Porcentaje de humedad después del secado - Término de tueste - Temperatura de tueste - Imagen de los granos de café durante el tueste - Fecha y hora de finalización de tueste - Fecha y hora de finalización de molienda

	- Precio en USD por kilogramo de café procesado - Peso en kilogramos del lote de café procesado
Catación	- Puntuación de catación - Precio en USD por el servicio de catación
Venta de Café	- Peso en kilogramos del lote de café vendido - Precio de venta en USD por kilogramo de café vendido
Bodegaje	- Dirección de ubicación de la bodega - Latitud y longitud de la ubicación de la bodega - Fecha y hora de llegada del lote en la bodega - Tiempo de bodegaje en días - Precio en USD por kilogramo de café y por día almacenado
Transporte hacia Empacador	- Tipo de transporte empleado en el envío - Fecha y hora de recogida en la bodega - Precio en USD del transporte
Empacado	- Dirección de ubicación del empacador - Latitud y longitud de la ubicación del empacador - Fecha y hora de llegada del lote en el empacador - Fecha y hora de empacado de las fundas de café a partir del lote de café recibido desde la bodega - Precio en USD por kilogramo de café empacado
Transporte hacia Retailer	- Tipo de transporte empleado en el envío - Fecha y hora de recogida en el empacador - Precio en USD del transporte
Comercialización en Retailer	- Nombre del almacén del retailer - Dirección de ubicación del almacén del retailer - Latitud y longitud de la ubicación del almacén - Fecha y hora de llegada al almacén del retailer - Nombre del punto de venta del retailer - Dirección de ubicación del punto de venta del retailer - Latitud y longitud de la ubicación del punto de venta - Fecha y hora de llegada al punto de venta del retailer - Tipo de transporte empleado en el envío desde el almacén hasta el punto de venta del retailer - Precio en USD del transporte - Precio en USD por kilogramo del lote de café comercializado en el retailer

2.3 ARQUITECTURA DEL SISTEMA DE TRACKING

La arquitectura completa del sistema de tracking de la cadena de suministro del café de especialidad con base en la tecnología de la cadena de bloques consta de 3 componentes principales que son:

1. Contratos inteligentes desarrollados en el lenguaje de programación Solidity y ejecutados en la cadena de bloques de prueba Rinkeby basada en el ecosistema de desarrollo de la red principal de Ethereum, los cuales describen toda la lógica del sistema de tracking implementado. Las funcionalidades principales que deben cumplir los contratos inteligentes para el sistema de tracking son:

- Generar un identificador único para cada lote de café a dar seguimiento.
- Permitir el ingreso de usuarios con los roles respectivos en el sistema, lo que habilita la edición de la información del lote de café según el rol.
- Permitir el ingreso de información referente a cada etapa de transformación del lote de café únicamente si la etapa en la que se encuentra el lote de café corresponde con el rol del editor.
- Permitir la actualización de información de perfil de los usuarios ingresados en el sistema.
- Permitir la lectura de los datos ingresados en cada etapa de transformación.
- Permitir la lectura de los datos sobre los usuarios ingresados en el sistema.

2. Integración entre la aplicación web con los contratos inteligentes por medio de la billetera digital MetaMask, instalada como extensión del navegador web.
3. Aplicación web basada en React como librería de desarrollo y Ethers.js como librería para interactuar con el ecosistema de la cadena de bloques de Ethereum. Esta será la interfaz para que los diferentes participantes que intervienen en la cadena de suministro de café puedan ingresar los datos en la blockchain, con respecto a su intervención. Los participantes de la cadena de suministro pueden editar la información de su café a través de la transferencia del identificador único del lote por medio de un código QR, por lo cual la aplicación web debe permitir escanear el código QR e ingresar los datos correspondientes del lote de café en cuestión. Así también, todos los participantes de la cadena de suministro y el consumidor final pueden acceder a la información del café que pertenece a un lote específico, por medio del escaneo del código QR correspondiente.

En la Figura 2.1 se observa un esquema de la arquitectura propuesta para este proyecto. Cada uno de los componentes del sistema se expondrán en las siguientes secciones del presente capítulo.

2.4 IMPLEMENTACIÓN DE LOS CONTRATOS INTELIGENTES

La implementación de los contratos inteligentes se llevó a cabo en el lenguaje de programación Solidity, empleando el entorno de desarrollo y marco de pruebas de Truffle, que permite administrar contratos inteligentes, crear scripts para desplegarlos, generar tests para depurar el código y ejecutarlos sobre la Máquina Virtual de Ethereum (EVM). Además, se utilizó Ganache como herramienta que permite correr una cadena de bloques

localmente en la memoria, basada en Ethereum, para implementar los contratos inteligentes y ejecutar las pruebas de los contratos desarrollados con la herramienta de Truffle. Por otro lado, la ejecución final de los contratos inteligentes para el prototipo del sistema de tracking se realizó en la red de prueba o testnet de Ethereum, Rinkeby, esto debido a que la implementación real en la red de Ethereum tiene costo, sin embargo, se requiere hacer pública la implementación y ejecución de los contratos inteligentes en la red por lo que no basta con el entorno de prueba local provisto por la herramienta Ganache.

Las herramientas empleadas para el desarrollo e implementación de los contratos inteligentes, así como la definición de la estructura y funcionalidades de estos, se describen a continuación.

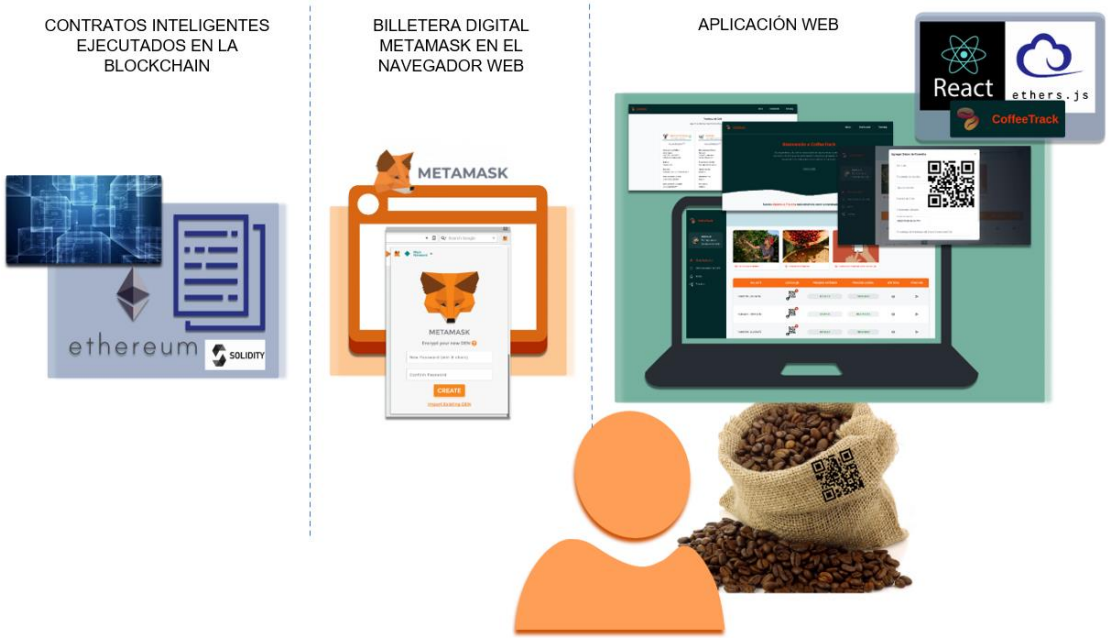


Figura 2.1. Arquitectura del sistema de tracking

2.4.1 CADENA DE BLOQUES DE ETHEREUM

La cadena de bloques seleccionada para el sistema de tracking implementado es la blockchain de Ethereum, ya que cumple con los requisitos de ser una cadena de bloques pública y sin permisos, definidos en la Sección 2.1.

Otras características de la blockchain de Ethereum que fundamentan la elección de esta plataforma para el desarrollo del presente proyecto se describen a continuación:

a) Máquina Virtual de Ethereum

Implementa la Máquina Virtual de Ethereum (EVM, por sus siglas en inglés) que se trata de un servidor único cuyo estado es actualizado por el consenso de todos los nodos participantes de la red, define las reglas de cálculo de cada nuevo estado válido de bloque

en bloque [27]. El protocolo de Ethereum tiene como objetivo mantener el funcionamiento de manera ininterrumpida, continua e inmutable de esta máquina de estado distribuida. Dentro del contexto de este proyecto, la EVM al ser un entorno de compilación en tiempo de ejecución permite compilar a bytecode los contratos inteligentes desarrollados en el lenguaje de programación Solidity y ejecutarlos a través de códigos de operación sobre la EVM.

b) Mecanismo de consenso para validación de bloques

Actualmente la red utiliza el mecanismo de consenso de prueba de trabajo (PoW) para la validación de los bloques. Pese a que la prueba de trabajo permite crear bloques válidos de manera sistemática, garantizando la seguridad y descentralización de la cadena, se puede pensar en el alto coste energético y medioambiental como resultado de la competición entre nodos para resolver el problema matemático propuesto. Por lo que el algoritmo de prueba de trabajo no es una solución escalable para la implementación de las cadenas de bloques. En este sentido, la red de Ethereum tiene como objetivo migrar al algoritmo de consenso de prueba de participación (PoS, por sus siglas en inglés) para finales del presente año. Este algoritmo consiste en la elección del nodo validador del bloque en función de la cantidad de criptomonedas de la red (ethers) que posea el nodo, todos los nodos participantes de la elección del validador, bloquean sus fondos en un depósito común y no pueden retirarlos hasta que el bloque sea validado, de esta manera se incentiva la honestidad en el proceso de validación debido a que si el nodo elegido manipula los datos pierde sus fondos [27]. Por otro lado, el nodo validador es elegido por un algoritmo aleatorio en función de la cantidad de ethers retenidos que garantiza que un mismo nodo no pueda ser elegido repetidas veces de manera seguida. De esta forma, la prueba de participación elimina totalmente el gasto computacional innecesario presente en la prueba de trabajo.

c) Contratos Inteligentes

Permiten el desarrollo y despliegue de contratos inteligentes. El lenguaje de programación de los contratos más empleado en Ethereum es Solidity. Este es un lenguaje de alto nivel orientado a objetos, que está diseñado para apuntar a la EVM y se encuentra influenciado por los lenguajes C++, Python y JavaScript [28].

d) Aplicaciones Descentralizadas

Facilita el desarrollo de aplicaciones descentralizadas (Dapps, por sus siglas en inglés) debido a que permite la escritura de contratos inteligentes. Una Dapp es una aplicación

digital que emplea blockchain para mantener los datos de la aplicación y manejar su sintaxis operativa en servidores que no sean centralizados, combinando un contrato inteligente con una interfaz de usuario [27], siendo esta la misma arquitectura sobre la cual se desarrolla el presente trabajo.

e) Criptomoneda y Tarifas de Gas

El ether (ETH) es la criptomoneda que usa Ethereum. La criptomoneda en una blockchain pública y sin permisos es el combustible que permite poner en funcionamiento la red ya que se utiliza para pagar los recursos computacionales empleados por los nodos durante la validación de bloques, lo que mantiene la descentralización y seguridad de la red [27]. Cabe mencionar que una parte de esta remuneración viene de la tarifa que pagan los usuarios para interactuar con la blockchain por medio de transacciones ya sea para realizar pagos entre participantes o para implementar y usar contratos inteligentes, mientras que otra parte se obtiene de la creación de nuevas criptomonedas que se da en cada validación de un nuevo bloque.

La unidad que mide la cantidad de recursos computacionales necesarios para llevar a cabo transacciones en la red de Ethereum, se denomina gas y corresponde al valor que debe pagar un usuario de la red. Este valor se calcula con la siguiente fórmula [27]:

$$T_{tx} = U_{gas}(\text{límite}) \times \left(T_{base} + P \right) \tag{2.1}$$

La definición de los parámetros de la fórmula anterior se visualiza en la tabla del Anexo II. El uso de tarifas de gas para validar transacciones también es una forma de prevenir transacciones de tipo spam por parte de actores maliciosos que buscan generar bucles hostiles en la red [27]. Sin embargo, la popularidad en el uso de la blockchain de Ethereum ha generado un incremento en la congestión de la red, aumentando el valor de las tarifas de gas a pagar y disminuyendo la velocidad de las transacciones validadas por segundo. Por esta razón, varias soluciones de escalado se están llevando a cabo, algunas de estas se describen en el Anexo III.

Debido a las tarifas de gas que requiere la implementación de contratos inteligentes, para este proyecto se ha utilizado una cadena de bloques de prueba basada en Ethereum, la cual permite usar criptomonedas (ETH) de prueba que no tienen valor real, para simular el despliegue y uso de los contratos sobre una blockchain pública y sin permisos.

Por otro lado, si se desea llevar a cabo la implementación de los contratos inteligentes sobre una cadena de bloques real del tipo pública y sin permisos, con el fin de reducir los costos por las tarifas de gas, se ha de considerar la utilización de una solución de escalado

de la blockchain de Ethereum, como son las cadenas laterales. En este sentido, se utilizaría la cadena de bloques Polygon la cual es compatible con la EVM.

La configuración y uso para ambas implementaciones se describe a lo largo de las siguientes secciones.

2.4.2 CONFIGURACIÓN DE LA CADENA DE BLOQUES TESTNET DE RINKEBY EN LA BILLETERA WEB DE METAMASK

Para interactuar con el ecosistema de la blockchain de Ethereum se requiere de una billetera criptográfica que administre la clave privada y pública del usuario para almacenar e intercambiar criptomonedas y acceder a aplicaciones descentralizadas a través de un inicio de sesión seguro. Para este proyecto se emplea la billetera web MetaMask, la cual es una plataforma de software, abierta y gratuita que es instalada como extensión en el navegador web. MetaMask se encuentra precargado con conexiones rápidas a diferentes cadenas de bloques de Ethereum, en este caso se emplearán conexiones a la blockchain de prueba de Rinkeby.

Estas conexiones facilitan que los usuarios no tengan que configurar y ejecutar un nodo de red completo que conlleva la descarga de una copia de la blockchain completa y la ejecución de la EVM, sino que interactúan como nodos livianos por medio de la billetera digital.

Una vez instalada la extensión de MetaMask en el navegador, creada la clave privada y pública de la billetera a través de la generación de la frase secreta de recuperación y la contraseña de ingreso (esto se explica en detalle en el manual de usuario del Anexo XL), se procede a añadir la cadena de bloques a utilizar. Para agregar la red de Rinkeby en MetaMask, se siguen los pasos del Anexo IV. El Anexo V describe el procedimiento para añadir la cadena de bloques Polygon en la billetera MetaMask en caso de realizar el despliegue de los contratos en una red real, teniendo en cuenta que esto requiere de criptomonedas con valor real.

2.4.3 CONFIGURACIÓN DEL ENTORNO DE DESARROLLO PARA LOS CONTRATOS INTELIGENTES CON TRUFFLE

Para inicializar un proyecto con Truffle, se sigue la siguiente configuración en una terminal de comandos:

1. Instalación de Truffle: *npm install -g truffle*
2. Creación de un directorio de trabajo para el proyecto

3. Inicialización del proyecto dentro de la carpeta creada: *truffle init*

El proyecto creado muestra la siguiente estructura:

- Carpeta *contracts*: Se agregan todos los contratos inteligentes del proyecto. Contiene archivos .sol, correspondientes al lenguaje Solidity.
- Carpeta *migrations*: Contiene los scripts para desplegar los contratos inteligentes. Los archivos se escriben en el lenguaje JavaScript.
- Carpeta *test*: Se agregan todos los archivos de test que permiten probar la funcionalidad de los contratos desarrollados. Contiene archivos escritos en JavaScript.
- Archivo *truffle-config.js*: Es el archivo principal de configuración del proyecto de Truffle. Se pueden editar configuraciones referentes a la conexión con diferentes redes, características de las migraciones, versión de Solidity para la compilación, configuraciones para ejecutar pruebas, etc.

En el Anexo VI, se muestra las capturas de pantalla correspondientes a los comandos ejecutados en la terminal para la creación del proyecto con Truffle y el árbol de archivos y carpetas del proyecto creado.

2.4.4 ESTRUCTURA Y DISEÑO DE LOS CONTRATOS INTELIGENTES

Los contratos inteligentes manejan toda la funcionalidad de almacenamiento, asignación, lectura de datos y asociación entre usuarios con la unidad trazable del lote de café del sistema de tracking. El sistema cuenta de 10 etapas, en donde cada etapa tiene un actor con un rol asignado que es quien interactúa con el contrato inteligente para actualizar la información sobre las variables descritas en la Tabla 2.1 con respecto al estado actual del lote de café en la cadena de suministro.

El funcionamiento del sistema empieza con el registro de la granja en donde se llevó a cabo el cultivo del lote de café como unidad trazable, por parte del administrador del sistema, para esto se emplea la función *añadirDatosGranja* que actualiza los valores de las variables de estado del contrato inteligente referentes al registro de la granja, esta función también crea el identificador único del lote de café y define que el siguiente rol de modificación corresponde al *AGRICULTOR*, a continuación, se emite el evento *DatosGranjaSeteados* en la blockchain. Luego, corresponde ingresar los datos sobre la cosecha por parte de un usuario con rol de *AGRICULTOR*, para esto un modificador de función, valida que el usuario tenga el rol requerido con respecto al siguiente estado o rol de modificación que presenta la unidad trazable del lote de café a la que se apunta para

añadir información, si el rol corresponde con el estado asignado, entonces el usuario agrega la información con la función *añadirDatosCosecha*, se define el siguiente rol de modificación como *PROCESADOR* y se emite el evento *CosechaRealizada* en la cadena de bloques. Lo anterior, se repite a lo largo de las siguientes etapas de intervención del lote de café, en donde cada etapa asigna el siguiente rol de modificación como sigue: *AGRICULTOR, PROCESADOR, CATADOR, VENDEDOR, BODEGA, TRANSPORTISTA_EMPACADOR, EMPACADOR, TRANSPORTISTA_RETAILER, RETAILER*.

Para escribir los datos en las funciones, además de la información referente al proceso, se requiere ingresar como argumento el identificador único generado para el lote de café, con el fin de saber a qué lote se apunta. El identificador único de cada lote será transmitido a través de los diferentes participantes de la cadena de suministro del lote. Una vez que cada participante termina su intervención, se pone de acuerdo con el siguiente participante en cuanto a temas de envío del producto con su identificador único presente en forma de un código QR, pagos, etc., entonces se registra en el sistema de tracking desarrollado, sus datos definitivos, para ser escritos en la blockchain.

Como se señaló, los eventos se desencadenan al ejecutarse una función en los contratos inteligentes. El uso de eventos es primordial para comunicarse con las interfaces de usuario o Dapps y mostrar información al usuario. En el sistema desarrollado, todos los eventos desencadenados tienen como argumentos la dirección de cuenta del usuario que llama la función de modificación y el identificador único de lote, esto facilita que luego el código de la aplicación web filtre los eventos ya sea por usuario modificador o por identificador de lote de café para leer la información ingresada en los registros de la cadena de bloques con respecto a cada etapa de la cadena de suministro de café.

En el Anexo VII, se muestra el flujo completo de funcionamiento del sistema por medio de un diagrama de estados.

El administrador del sistema también es el encargo de ingresar a los usuarios que intervienen, con su dirección de cuenta pública de billetera MetaMask con la que se llevará a cabo la interacción con la aplicación web y el rol o roles respectivos según la función de los usuarios. Debido al modelo de la cadena de suministro con la que se está trabajando, descrita en la Tabla 1.1, el agricultor que cultiva y cosecha el lote de café, es también quien vende su café luego de haberlo procesado y catado, por lo que la dirección de cuenta de este usuario debe tener los roles tanto de agricultor como de vendedor de café.

En resumen, las acciones que puede realizar un usuario según su rol asignado a lo largo de las etapas de intervención de un lote de café se muestran a continuación:

Tabla 2.2. Acciones que pueden realizar los usuarios según sus roles

Etapa	Rol	Acciones
Registro de la granja	ADMINISTRADOR	- Ingreso de usuarios en el sistema y asignación de sus roles. - Creación de nuevos lotes de café. - Ingreso de información referente a la granja de cultivo del lote de café creado.
Cosecha	AGRICULTOR	- Modificación de su información de perfil. - Ingreso de información referente a los datos de cosecha del lote de café.
Procesado	PROCESADOR	- Modificación de su información de perfil. - Ingreso de información referente a los datos de procesamiento del lote de café.
Catación	CATADOR	- Modificación de su información de perfil. - Ingreso de información referente a los datos de catación del lote de café.
Venta de Café	VENDEDOR	- Modificación de su información de perfil. - Ingreso de información referente a los datos de la venta del lote de café.
Bodegaje	BODEGA	- Modificación de su información de perfil. - Ingreso de información referente a los datos de almacenamiento del lote de café.
Transporte hacia Empacador	TRANSPORTISTA_ EMPACADOR	- Modificación de su información de perfil. - Ingreso de información referente a los datos de transporte del lote de hacia empacador.
Empacador	EMPACADOR	- Modificación de su información de perfil. - Ingreso de información referente a los datos de empacado del lote de café.
Transporte hacia Retailer	TRANSPORTISTA_ RETAILER	- Modificación de su información de perfil. - Ingreso de información referente a los datos de transporte del lote de café hacia retailer.
Comercialización en Retailer	RETAILER	- Modificación de su información de perfil. - Ingreso de información referente a los datos de comercialización del lote en el retailer.

Se codificaron cinco contratos inteligentes: *SupplyChainStorage.sol*, *SupplyChainStorage2.sol*, *CoffeeSupplyChain.sol*, *CoffeeSupplyChain2.sol*, *SupplyChainUser.sol*. Los dos primeros contratos se encargan principalmente de realizar el almacenamiento de los datos referentes a los lotes de café como a los usuarios del sistema en la blockchain, otras funcionalidades que realizan son: definir el siguiente rol de modificación en la cadena de suministro, realizar el cálculo para la creación del identificador único del lote de café, definir los tipos de datos de las variables de estado que se escribirán en la blockchain, escribir los datos en la blockchain según las etapas y obtener la información a través de funciones que, por medio del identificador único de lote, apuntan a

los datos requeridos. Debido al límite de codificación de 24kBytes que presentan los contratos inteligentes del ecosistema de Ethereum, se requirió dividir la lógica presente en el contrato en dos partes, así el contrato *SupplyChainStorage.sol* presenta las funciones para almacenar los usuarios y las etapas desde el registro de la granja hasta la venta del lote de café, mientras que el contrato *SupplyChainStorage2.sol* tiene las funciones de las etapas desde el bodegaje hasta la comercialización en retailer. Los siguientes dos contratos (*CoffeeSupplyChain.sol*, *CoffeeSupplyChain2.sol*) se encargan de emitir los eventos en la blockchain, de validar si el rol del usuario corresponde con el siguiente rol de modificación especificado para el del lote de café y llamar a las funciones de los primeros contratos para escribir y leer datos en la blockchain. De manera similar a los primeros contratos, se requirió dividir la codificación de los contratos en dos partes. Finalmente, el contrato restante se encarga de emitir los eventos en la blockchain con respecto al manejo de usuarios en el sistema, para esto también llama a las funciones de los dos primeros contratos.

Para permitir las llamadas y el uso de las funciones de los dos primeros contratos, estos requieren implementar las funciones *autorizarLlamada* y *desautorizarLlamada*, las cuales habilitan las llamadas por parte de otros contratos al ingresar la dirección de estos contratos como argumento de las funciones de autorización. Así también, el contrato *SupplyChainStorage2.sol* requiere ser habilitado para llamar al primer contrato ya que usa algunas de sus funciones a razón de la división de la lógica del contrato.

Las funciones que implementa cada contrato se muestran en el diagrama de modelo del Anexo VIII. Por otro lado, la interacción entre los usuarios con cada contrato e interacción entre contratos enfocado en las funciones de escritura que desencadenan los eventos emitidos en la blockchain, se muestran en el diagrama de secuencia del Anexo IX, para las etapas desde el registro de la granja hasta la venta de café, mientras que en el Anexo X se muestra el diagrama de secuencia para las etapas desde bodegaje hasta la comercialización en retailer. En el Anexo XI, se muestra el diagrama de secuencia enfocado en las funciones de lectura de los contratos desarrollados.

En el Anexo XII, se visualiza el diagrama de entidad relación del sistema en donde se especifican las variables con su tipo de dato, las funciones, los atributos y las llamadas de función que presentan las diferentes entidades. Se puede observar que el tipo de dato para las variables numéricas de punto flotante son de tipo *string* debido a que no existe el tipo de dato *float* en Solidity.

En el apartado 2.3 se mencionaron las funcionalidades principales que deben cumplir los contratos inteligentes, la lógica programática de estas funciones se explica a continuación a través de diagramas de flujo.

a) Generación del identificador único de lote de café

Se empleó la función de hashing criptográfico *Keccak-256* la cual está integrada en Solidity y permite calcular una salida única de 32 bytes (256 bits) a partir de cualquier cantidad de entradas [29]. Una de las aplicaciones de las funciones de hashing criptográficas es la creación de identificaciones únicas y deterministas a partir de un conjunto de datos de entrada, siendo esta aplicación útil para el presente caso.

Como parámetros de entrada de la función de hashing se definieron la dirección de la cuenta de MetaMask de quien ejecuta la función, que es el administrador y la marca de tiempo de generación del bloque en la blockchain, este segundo parámetro garantiza de que cada identificador de lote sea único ya que cada registro de una granja se realiza a través de una transacción que es agregada en un bloque de transacciones, en donde la generación del nuevo bloque tiene una marca de tiempo única.

Debido a que la función de hashing *Keccak-256* acepta únicamente entradas en bytes, se requiere convertir los parámetros en bytes usando la función *abi.encodePacked*. Luego, el resultado se convierte en un entero sin signo de 160 bits que corresponde a los últimos 20 bytes de la salida de la función *Keccak-256*. El resultado finalmente es almacenado en un tipo de dato *address* existente en Solidity, el cual es equivalente a 20 bytes (160 bits). La codificación de lo anterior se muestra en la Figura 2.2.

```
uint256 tmpData = uint256(keccak256(abi.encodePacked(msg.sender, block.timestamp)));
address batchNo = address(uint160(tmpData));
```

Figura 2.2. Generación del identificador único de lote usando la función Keccak-256

La función de hashing es parte de la función *setearDatosGranja* del contrato *SupplyChainStorage.sol* que es llamada por la función *añadirDatosGranja* del contrato *CoffeeSupplyChain.sol* que a su vez es ejecutada por el administrador del sistema. En el Anexo XIII, se muestra el diagrama de flujo de la función *añadirDatosGranja*.

b) Ingreso de usuarios con los roles respectivos en el sistema

El ingreso de usuarios con sus roles en el sistema se lleva a cabo por parte del administrador a través de la función *ingresarUsuario* del contrato *SupplyChainUser.sol* la cual invoca a la función *setearUsuario* del contrato *SupplyChainStorage.sol*. Las variables

para definir a los usuarios son: dirección de cuenta MetaMask, nombre completo, email, rol(es), estado de actividad, imagen de perfil. Cabe mencionar que para cumplir con el requisito de que una misma cuenta de MetaMask de un usuario pueda ser asignada con dos roles como en el caso de los roles *AGRICULTOR* y *VENDEDOR*, el tipo de dato de la variable de estado rol es un *array* de datos tipo *string*, que admite como máximo 2 elementos. El diagrama de flujo del proceso para el ingreso de usuarios se muestra en el Anexo XIV.

c) Ingreso de información del lote de café con respecto a su etapa de transformación

En el Anexo XV, se visualiza el diagrama de flujo de la etapa de cosecha llevada a cabo por un usuario con el rol de *AGRICULTOR* al llamar la función *añadirDatosCosecha* del contrato *CoffeeSupplyChain.sol*. Las demás etapas tienen la misma lógica de funcionamiento, difiriendo únicamente en las funciones que utilizan.

d) Modificación de los datos de perfil de los usuarios ingresados en el sistema

Los usuarios ingresados en el sistema pueden actualizar la información sobre su perfil, iniciando sesión a través de la dirección de cuenta de su billetera MetaMask en la página web desarrollada (esto se describe a detalle en las secciones siguientes).

La información que pueden modificar los usuarios excluye a su dirección de cuenta MetaMask, ya que con esta acceden a su tablero personalizado en la página web y es definida únicamente por el administrador durante el registro del usuario en el sistema.

En el Anexo XVI, se presenta el diagrama de flujo de la función *actualizarUsuario* del contrato *SupplyChainUser.sol* que lleva a cabo la actualización de información de perfil por parte de los diferentes usuarios.

e) Lectura de los datos ingresados en cada etapa de transformación

En el Anexo XVII, se muestra el diagrama de flujo de la función *obtenerDatosProcesado* del contrato *CoffeeSupplyChain.sol* que lleva a cabo la lectura de información de la etapa de procesado. Las demás etapas tienen la misma lógica de funcionamiento, difiriendo únicamente en las funciones que utilizan.

f) Lectura de datos sobre los usuarios

El Anexo XVIII exhibe el diagrama de flujo de la función *obtenerUsuario* del contrato *SupplyChainUser.sol* que lleva a cabo la lectura de información de perfil de los usuarios.

Todas las funciones de lectura descritas en los contratos inteligentes son del tipo *view* lo que significa que estas funciones únicamente leen datos almacenados en la blockchain sin alterar las variables de estado definidas en los contratos, en consecuencia, estas funciones no requieren tarifa de transacción para ser invocadas por parte de cualquier usuario. Por otro lado, todas las funciones de escritura de los contratos si necesitan de una tarifa de transacción.

2.4.5 CONFIGURACIÓN DE LA BLOCKCHAIN LOCAL BASADA EN ETHEREUM CON GANACHE

Para setear la red local con Ganache, inicialmente se debe instalar la aplicación de escritorio y abrirla, a continuación, se siguen los siguientes pasos:

1. Para crear una nueva blockchain local, se selecciona el botón New Workspace Ethereum.
2. En la pestaña de espacio de trabajo, definir el nombre de este y seleccionar el archivo *truffle-config.js* del proyecto de Truffle que se va a ejecutar.
3. En la pestaña de servidor, se configura el hostname y el puerto que aceptarán las conexiones RPC. Las conexiones RPC significan llamada a procedimiento remoto y permiten comunicarse con servidores de forma remota para ejecutar programas en una ubicación separada [30], en este caso la conexión RPC habilita el acceso a un nodo de servidor de cadena de bloques de Ganache. Se mantiene la configuración predefinida por Ganache.
4. En la pestaña de cuentas y claves, se especifica la cantidad de cuentas y su balance inicial en ethers, requerida.
5. En la pestaña referente a la cadena de bloques, se define los parámetros para el precio de la tarifa de transacción y la bifurcación de la red Ethereum que se utilizará. Las bifurcaciones en blockchain permiten romper la cadena de bloques y crear una cadena nueva a partir del punto de ruptura de la cadena de bloques existente [31], esta es la forma en la que Ganache permite simular el entorno de cadena de bloques de manera local. Se mantiene la configuración predefinida por Ganache para estos parámetros.
6. Finalmente, se guarda la configuración del espacio de trabajo.

En el Anexo XIX, se muestra las capturas de pantalla de la configuración mencionada.

2.4.6 DEFINICIÓN DEL ARCHIVO DE CONFIGURACIÓN DE TRUFFLE

En el archivo *truffle-config.js* se definirán las redes a conectarse en el momento del despliegue de los contratos inteligentes sobre dichas redes, la versión de compilación de Solidity por parte de la EVM y el complemento para verificar los contratos inteligentes en las redes del ecosistema de Ethereum.

a) Configuración de la conexión a las redes durante el despliegue de los contratos inteligentes

Los contratos inteligentes se desplegaron tanto en la red local provista por Ganache como entorno de prueba y en la testnet Rinkeby para hacer público el sistema de tracking sobre el ecosistema de Ethereum y permitir que los usuarios puedan explorar sus transacciones llevadas a cabo a través de la herramienta web de Etherscan.

En el código, la definición de las redes se lo realiza a través de un tipo de dato objeto de JavaScript cuya palabra clave es *networks*, este objeto principal contiene otros objetos con la información de despliegue para cada una de las redes.

Para la configuración de la red local con Ganache se establece el host, el puerto, las unidades de gas (límite de gas) de la tarifa de transacción y el número identificador de red. Mientras que para la testnet de Rinkeby se requiere definir el proveedor de nodo, el número identificador de red, las unidades de gas (límite de gas) de la tarifa de transacción, la dirección pública de cuenta MetaMask desde la cual se van a desplegar los contratos y los parámetros de *skyDryRun*, *timeoutBlocks* y *confirmations.* El significado de cada uno de los parámetros utilizados se puede consultar en el Anexo XX.

El valor del límite de gas en ambos casos, se lo estableció en base a prueba y error hasta que la transacción correspondiente a la implementación de los contratos en la blockchain local y testnet, se realice respectivamente. Por otro lado, la dirección de cuenta de MetaMask que se establece debe corresponder al propietario o administrador del sistema.

La conexión a una blockchain se realiza a través de nodos. En el caso de la red local, Ganache ya se encarga de realizar estas conexiones. Por otro lado, las cadenas de bloques ya sea de prueba o reales del ecosistema de Ethereum no cuentan con las facilidades que provee Ganache al ser un entorno local, por lo que se debería implementar un nodo propio. Sin embargo, la implementación de un nodo propio requiere de infraestructura, mantenimiento, tiempo, etc., siendo la opción más adecuada el uso de un proveedor de nodos que se encargue del manejo completo de estos. En este sentido, el proveedor de nodos que se utiliza a lo largo del proyecto tanto para el despliegue de los

contratos como para la conexión con la aplicación web es Infura. El servicio principal de Infura es gratuito, bajo esta modalidad el servicio tiene algunas limitaciones como la cantidad de proyectos que se pueden crear para conectarse con un nodo, la cantidad de peticiones por día que se pueden hacer, etc., el servicio se puede escalar y mejorar a través de opciones de pago, sin embargo, para el presente proyecto es suficiente e ideal el servicio gratuito que provee Infura.

Para obtener un nodo en Infura se requiere crear un proyecto en su página y obtener una clave API, para esto se siguen los pasos del Anexo XXI. Para utilizar el proveedor de nodo se debe establecer la firma de la transacción llevada a cabo en el momento de despliegue de los contratos en la red. Truffle puede realizar la firma de transacciones a través del paquete *HDWalletProvider*, por lo que debe ser instalado y definido en el archivo de configuración. El objeto *HDWalletProvider* acepta dos argumentos, el primero corresponde a la frase secreta o mnemónico de la billetera digital MetaMask con la que se va a desplegar el contrato, este valor es privado y se obtiene en el momento de creación de una cuenta en la billetera digital de MetaMask (este proceso se explica en el manual de usuario del Anexo XL), el segundo argumento es la URL con la clave API obtenida del nodo de Infura creado.

Es importante mencionar que tanto el mnemónico como la clave API privada del nodo de Infura no deben ser expuestos públicamente, por lo que sus valores fueron almacenados en variables de entorno en un archivo privado.

b) Versión de compilación

La versión de Solidity empleada en los contratos inteligentes es la 0.8.16.

c) Complemento para verificación de los contratos en el ecosistema de Ethereum

La verificación de contratos inteligentes es necesaria para promover la transparencia y garantizar la seguridad por parte de los usuarios en las aplicaciones con blockchain ya que esta tecnología se implementa a través de transacciones lo que involucra un costo económico para los usuarios. Al verificar un contrato desplegado en la blockchain, su código fuente se hace público y puede ser consultado a través del explorador web Etherscan.

Para añadir el complemento de verificación en la configuración de Truffle, se siguen los pasos del Anexo XXII. Por otro lado, el Anexo XXIII muestra el código completo del archivo de configuración truffle-config.js.

2.4.7 CONFIGURACIÓN DEL SCRIPT DE MIGRACIÓN

Un archivo de migración se encarga de automatizar la implementación de los contratos en la cadena de bloques, permite establecer los pasos requeridos para vincular contratos y configurar los datos iniciales de estos, en caso de requerirlo.

En este caso, el archivo de migración define el despliegue de los contratos de manera asíncrona pero ordenada. El orden de despliegue, los datos iniciales requeridos y las funciones ejecutadas para vinculación entre contratos, se describen a continuación.

Tabla 2.3. Detalles del script de migración para los contratos inteligentes

Orden	Contrato	Valores iniciales para el constructor	Función de vinculación entre contratos
1	SupplyChainStorage	No aplica	autorizarLlamada(), se ejecuta 4 veces con las direcciones de los contratos 2, 3, 4 y 5 como argumento
2	SupplyChainStorage2	Dirección de despliegue del contrato 1	autorizarLlamada(), se ejecuta 1 vez con la dirección del contrato 4
3	CoffeeSupplyChain		No aplica
4	CoffeeSupplyChain2	Dirección de despliegue del contrato 1 y 2	No aplica
5	SupplyChainUser	Dirección de despliegue del contrato 1	No aplica

El código del archivo de migración escrito en JavaScript se muestra en el Anexo XXIV.

2.4.8 PRUEBAS DE LOS CONTRATOS INTELIGENTES CON TRUFFLE Y GANACHE

El entorno de desarrollo de Truffle provee de un marco de pruebas automatizado para los contratos inteligentes, el cual implícitamente utiliza las librerías de JavaScript, Mocha para generar el entorno de pruebas y Chai para realizar validaciones. Por esta razón, los archivos deben ser escritos en JavaScript.

El detalle de cada caso de prueba codificado se visualiza en el Anexo XXV. Para llevar a cabo las pruebas se requiere ejecutar la aplicación de escritorio de la blockchain local de Ganache y ejecutar los siguientes comandos en una terminal del directorio del proyecto:

1. Compilación de los contratos inteligentes: *truffle compile*
2. Ejecución de las pruebas en la red local: *truffle test –network ganache*

En total se realizaron 28 casos de prueba. El registro de consola de todos los casos de prueba completados exitosamente al correr los tests, se muestran en el Anexo XXVI.

2.4.9 DESPLIEGUE DE LOS CONTRATOS INTELIGENTES EN LAS CADENAS DE BLOQUES

El despliegue, migración o implementación de un contrato inteligente sobre una cadena de bloques hace referencia al envío de una transacción a la blockchain que contiene el código compilado del contrato inteligente. Las transacciones en la blockchain requieren especificar un destinatario, sin embargo, las transacciones provenientes de contratos inteligentes no especifican ningún destinatario.

Para realizar la implementación de los contratos se requiere:

- El código en bytes de los contratos, esto se obtiene a través de su compilación.
- Disponer de ether en la cuenta de la billetera MetaMask para pagar las transacciones. En el caso de la red local con Ganache, previamente se especificó la cantidad de ethers de las cuentas (Sección 2.4.5). Para la red testnet de Rinkeby anteriormente también se obtuvo ethers a partir de un grifo de ethers sin valor (Anexo IV).
- El script de migración definido en la Sección 2.4.7 y la configuración de las redes definido en la Sección 2.4.6.

Para realizar la migración de los contratos, se ejecutan los siguientes comandos en una terminal del directorio del proyecto.

Tabla 2.4. Comandos de migración para la red local y testnet Rinkeby

Red	Etapa	Comando
Local y Testnet Rinkeby	Compilación	*truffle compile*
Local	Migración	*truffle migrate –network ganache*
Testnet Rinkeby		*truffle migrate –network rinkeby*

Los registros de consola obtenidos de las migraciones de los contratos en ambas redes se encuentran en el Anexo XXVII.

2.4.10 VERIFICACIÓN DE LOS CONTRATOS INTELIGENTES

Para ejecutar la verificación de los contratos, se utiliza el comando: *truffle run verify NombreContrato –network rinkeby*, este comando se ejecutó cinco veces con los diferentes nombres de los contratos. Las capturas de pantalla correspondientes a la ejecución del comando en la terminal, así como la visualización en Etherscan del contrato verificado, se muestran en el Anexo XXVIII.

La codificación del proyecto con Truffle correspondiente a los contratos, las migraciones, las pruebas, la configuración desarrollados para el sistema de tracking del café, se puede consultar a través del siguiente repositorio en GitHub: https://github.com/NathaliaBarreiros/coffee-supply-chain-tracking-system-ethereum.

2.5 APLICACIÓN WEB DEL SISTEMA DE TRACKING

Para el sistema de tracking codificado a través de los contratos inteligentes, una interfaz web es vital para permitir que los usuarios puedan interactuar con los contratos alojados en la blockchain. La construcción de la interfaz web de usuario se realizó empleando la biblioteca de desarrollo para interfaces de usuario, gratuita y de código abierto basada en JavaScript, React. A continuación, se explican las características principales que implementa la aplicación web para conectarse con los contratos inteligentes y ejecutar sus funcionalidades, así como el modelo de la aplicación web según el flujo de proceso de uso para los diferentes usuarios.

2.5.1 CARACTERÍSTICAS DE IMPLEMENTACIÓN DE LA APLICACIÓN WEB

2.5.1.1 Conexión de la Dapp con la Blockchain a través de MetaMask

Para interactuar con la cadena de bloques de prueba pública de Rinkeby se emplea la biblioteca API de JavaScript, Ethers.js. La billetera MetaMask brinda un proveedor de nodo ya que de manera implícita usa Infura (mencionado en la Sección 2.4.6), este es un conector hacia la blockchain; por otro lado, Ethers.js crea un proveedor y un firmante.

Un proveedor (*provider*) es una clase que brinda una abstracción para una conexión a la blockchain. Este proporciona acceso de solo lectura a la blockchain. Mientras que un firmante (*signer*) es una clase que tiene acceso a una firma con un par de clave pública y privada [32], lo que habilita el uso de la dirección de cuenta MetaMask de un usuario para firmar mensajes y transacciones e interactuar con la blockchain. En el Anexo XXIX, se exhibe un esquema detallado de la interacción entre Ethers.js, el navegador web y la cadena de bloques.

Para las transacciones de escritura se necesita la firma del usuario con su billetera MetaMask, la codificación de esto usando Ethers.js, se muestra a continuación:

```
export const coffeeSupplyChainWriter = () => {
  const provider = new ethers.providers.Web3Provider(window.ethereum);
  const signer = provider.getSigner();
  return new ethers.Contract(window.coffeeSupplyChainAddress, coffeeSupplyChainABI.abi, signer);
};
```

Figura 2.3. Proveedor para transacciones de escritura con MetaMask

De la figura anterior, la clase *Web3Provider* inyecta el objeto *window.ethereum* que es la conexión que brinda MetaMask. Obtenido el proveedor, se puede acceder al firmante a través del método *getSigner().* A continuación, para crear una instancia del contrato con el que se interactúa, se usa el objeto *Contract.* Este admite como argumentos, la dirección del contrato desplegado en la blockchain, el ABI del contrato y el firmante *(signer).* Este objeto es una abstracción del código fuente del contrato alojado en la blockchain. El ABI o Interfaz Binaria de Aplicación, es un archivo JSON que contiene todas las funciones disponibles y la forma de codificación y decodificación de datos de un contrato [32], es producto de la compilación del código fuente en Solidity.

La aplicación web utiliza únicamente los contratos: *CoffeeSupplyChain.sol*, *CoffeeSupplyChain2.sol* y *SupplyChainUser.sol*, ya que estos de manera implícita usan las funciones de los contratos de almacenamiento: *SupplyChainStorage.sol* y *SupplyChainStorage2.sol*. Para cada uno de los 3 contratos, se crearon respectivamente 3 proveedores diferentes.

Las funciones de lectura no requieren del firmante (*signer),* solo del proveedor. La codificación de este proveedor se muestra en la Figura 2.4. Como los 3 contratos tienen funciones de lectura, se crearon 3 proveedores para cada contrato.

```
export const coffeeSupplyChainReader = () => {
    const provider = new ethers.providers.Web3Provider(window.ethereum);
    return new ethers.Contract(window.coffeeSupplyChainAddress, coffeeSupplyChainABI.abi, provider);
};
```

Figura 2.4. Proveedor para transacciones de lectura con MetaMask

De esta manera, los partícipes de la modificación del lote de café a lo largo de su cadena de suministro requieren tener instalada la extensión de MetaMask en su navegador para interactuar con la aplicación web. Sin embargo, la aplicación también tiene otro tipo de usuarios que son los consumidores del café de especialidad, quienes únicamente van a consultar el historial del lote de su café. Así, para facilitar el acceso a la información para este tipo de usuarios sin requerir instalar la extensión de MetaMask y considerando que estos no deben firmar ninguna transacción, para las funciones de lectura, se utiliza Infura directamente como proveedor de conexión al nodo de la blockchain en lugar de MetaMask que intrínsecamente emplea Infura.

Para definir una abstracción de proveedor con Infura, se requiere especificar el nombre de la blockchain que aloja los contratos inteligentes, así como la clave API del nodo generado en Infura para el proyecto (del Anexo XXI). La codificación de este proveedor se muestra en la Figura 2.5. De igual manera, se crearon 3 proveedores diferentes para los 3 contratos.

```
export const infuraCoffeeSupplyChainReader = () => {
    const provider = new ethers.providers.InfuraProvider('rinkeby', process.env.REACT_APP_INFURA_PROJECT_ID);
    return new ethers.Contract(window.coffeeSupplyChainAddress, coffeeSupplyChainABI.abi, provider);
};
```

Figura 2.5. Proveedor para transacciones de lectura con Infura

2.5.1.2 Escritura y Lectura de Datos en los Contratos Inteligentes

Para la escritura de datos en los contratos, se usa el proveedor para la función de escritura antes especificado, el cual devuelve una instancia del contrato al que se apunta. Se usa la instancia del contrato como un objeto, con el cual se puede acceder a sus métodos que serían las funciones de escritura del contrato, teniendo como argumentos de estas funciones, los valores ingresados por el usuario a través de un formulario presente en el panel de control del usuario según su rol obtenido al acceder a la aplicación web con su cuenta MetaMask previamente registrada por el administrador del sistema. La codificación de la función de escritura *añadirDatosVentaCafé*, se muestra en la Figura 2.6. Lo mismo se realiza para cada una de las funciones de escritura.

```
const HandleSubmit = (values) => {
  const coffee1Contract = coffeeSupplyChainWriter();
  return coffee1Contract.addCoffeeSellData(values.batchNo, values.coffeeSellingBatchWeight, values.beanPricePerKilo);
};
```

Figura 2.6. Función de escritura de datos de la aplicación web

Para las funciones de lectura se utiliza el tipo de llamada estática *.callStatic*, seguido de la función de lectura del contrato. En la aplicación se emplean tanto el proveedor obtenido con MetaMask como el proveedor de Infura, dependiendo que si el usuario al que se está sirviendo la información de lectura obtenida, es un modificador en la cadena de suministro o es únicamente un consumidor. A continuación, se muestra un ejemplo de la codificación de la función de lectura *obtenerDatosEmpacado*.

```
const coffee1ReaderInfura = await infuraCoffeeSupplyChainReader();
const info = await coffee1ReaderInfura.callStatic.getPackData(values.batchNo);
```

Figura 2.7. Función de lectura de datos de la aplicación web

2.5.1.3 Escucha de Eventos

La escucha de eventos es útil debido a que las llamadas de las funciones de escritura se realizan fuera de la cadena como una transacción que es manejada por Ethers.js, luego la transacción debe ser validada y ejecutada por la red, que no es un proceso instantáneo, entonces la emisión de eventos permite confirmar la transacción realizada en la blockchain a través de la escucha de estos eventos por parte de la aplicación web. Para escuchar un

evento nuevo, se emplea el método *.on()*, cuyo primer argumento es el nombre del evento que se codificó en el contrato y una función que emplea los datos obtenidos del evento para llevar a cabo una acción con los datos devueltos del evento. Un ejemplo de codificación de escucha de un evento implementado en la aplicación web, se muestra en el Anexo XXX.

Además, los eventos ya emitidos en la blockchain pueden ser filtrados. Para esto se emplea el método .queryFilter() cuyo argumento es el objeto a filtrar usando el método .filters.NombreEvento() que tiene como argumentos los temas del filtro. En la aplicación, todos los eventos pueden ser filtrados tanto por la dirección de cuenta MetaMask del usuario que ejecutó la transacción de modificación del lote de café, como por el identificador único del lote de café. En el Anexo XXXI, se muestra un ejemplo del código de filtrado.

2.5.1.4 IPFS para el Almacenamiento de Imágenes

Debido a que almacenar archivos masivos como imágenes en la blockchain es muy costoso y poco eficiente, como alternativa se puede emplear IPFS. IPFS o Sistema de Archivos Interplanetario, es un sistema de almacenamiento distribuido peer-to-peer y uso compartido de archivos [33].

En la aplicación se usa IPFS para almacenar los datos con formato de imagen, así, se envía el archivo de imagen a IPFS, este identifica a la imagen por su contenido devolviendo un CID (identificador de contenido), que es un hash (definido en la Sección 1.4.4), luego se guarda en la blockchain únicamente el valor del CID identificador de la imagen. El CID también se emplea para obtener la imagen del sistema IPFS y mostrarla en la aplicación. Se utilizó Infura para conectar un nodo de IPFS, el proceso se describe en el Anexo XXXII. La codificación para emplear IPFS se muestra en el Anexo XXXIII.

2.5.1.5 Código QR para Transmisión del Identificador de Lote

Uno de los requerimientos del sistema para dar seguimiento al lote de café, es emplear una tecnología de transmisión del identificador único creado. En base a las tecnologías descritas en el apartado 1.4.3.2, se ha decidido emplear los códigos de respuesta rápida QR. La razón es porque permiten almacenar información que vincula a un sitio web, además es una tecnología de fácil adopción y uso desde el punto de vista de todos los participantes de la cadena de suministro del café y los consumidores, también el costo de su implementación es bajo.

En la aplicación, el código QR almacena la URL de la aplicación con el parámetro del identificador único del lote de café. Así, al escanear el código QR presente en el lote de café físico, los diferentes modificadores del lote pueden acceder a la URL, la cual redirige

a una página de la aplicación web que muestra un formulario para ingresar los datos con respecto a su intervención. La aplicación implementa la funcionalidad de generación del código QR por medio de la librería *qrcode.react*, así como el escaneo del código QR con la librería *react-qr-reader*. El uso de estas librerías se muestra en el código del Anexo XXXIV. Cabe mencionar que también se añadió la funcionalidad de envío de la URL con el parámetro del lote de café, por medio de redes sociales y correo, como alternativa de transmisión del identificador de lote. Además, también se implementó la descarga del código QR generado para ser impreso en el producto físico antes de ser enviado al siguiente participante de la cadena.

El escaneo del código QR también es empleado por el consumidor final, para obtener la URL que le redirige a una página de la aplicación web que muestra el historial de su café.

2.5.1.6 Google Maps API para los Datos de Ubicación

Algunos de los datos que se ingresan en el sistema, corresponden a la información de la dirección y geolocalización (latitud y longitud) del lugar de transformación del lote de café, para facilitar el ingreso de esta información a los usuarios, se emplearon las librerías *@react-google-maps/api* que proporciona enlaces a la API de Google Maps y permite desplegar un mapa, navegar por el mapa y ubicar visualmente un marcador en la dirección requerida y *use-places-autocomplete*, que emplea Google Maps Places API para autocompletar las direcciones en una barra de búsqueda. Además, la API de Google Maps brinda el servicio de geolocalización, usado en la aplicación para obtener los valores de geolocalización a partir de la dirección seleccionada por los usuarios en el mapa. El mapa también se utiliza en la página que muestra el historial del lote de café para ubicar visualmente los puntos de intervención del café.

Para usar la API de Google Maps se requiere obtener una clave API asociada a la creación de un proyecto en su página, la cual autentica las solicitudes al servicio de Google Maps, este proceso se encuentra en el Anexo XXXV. El Anexo XXXVI, muestra la codificación realizada para las funcionalidades que emplean la API de Google Maps.

2.5.2 MODELO DE LA APLICACIÓN WEB Y FLUJOS DE PROCESO DE USO

La aplicación web creada se llama CoffeeTrack y cuenta de 3 vistas principales: la página de Inicio, la página de tablero o panel de control (Dashboard) y la página para la consulta del historial del lote de café (Tracking). El Anexo XXXVII muestra un diagrama de la organización visual de la interfaz web desarrollada.

La página de Inicio se muestra al cargar la URL de la aplicación, esta muestra información sobre el proyecto, la tecnología blockchain para el seguimiento del café y un campo de

contacto para que los usuarios interesados en utilizar el sistema puedan comunicarse con el administrador. La página de Dashboard puede ser accedida únicamente por los usuarios que hayan sido ingresados en el sistema y por el administrador. Todos los usuarios de modificación y el administrador pueden acceder a su panel de control personalizado una vez que hayan iniciado sesión en la aplicación por medio de su cuenta de billetera MetaMask. Por esta razón, estos usuarios requieren tener instalada la extensión de MetaMask en su navegador. El panel de control de cada usuario muestra el nombre del usuario y su rol y permite que estos puedan actualizar la información de su perfil y agregar los datos de modificación de un lote de café específico por medio de formularios cuyos campos de entrada corresponden a los argumentos de las funciones de los contratos codificados. Para agregar la información de intervención de un lote, existen 3 posibilidades:

1. Agregar la información y el identificador de lote por parte del usuario, esta alternativa puede ser emplea cuando se tiene únicamente el identificador del lote sin un código QR ni URL asociada.

2. Agregar la información a través de la lectura de un código QR que, al contener la URL con el identificador del lote no requiere el ingreso del identificador de lote por parte del usuario, ya que este valor se pre setea por el sistema. Esto abre una ventana emergente en donde se debe mostrar el código QR a través de la cámara del computador, cuando se lee efectivamente el código, se abre otra ventana emergente con el formulario para ingresar los datos.

3. Agregar la información a través de una URL directamente, la cual pudo ser enviada por medio de una red social o correo. En este caso, la URL redirige al usuario a la página de Tracking para ese lote de café específico, aquí se muestra el historial de tracking hasta el proceso anterior realizado y un botón que redirige a un formulario para el nuevo proceso a ingresar, cuyo campo de entrada correspondiente al identificador de lote ya está pre seteado.

En cualquiera de los 3 casos anteriores, cuando el usuario ingresa los datos en el formulario y selecciona el botón para agregar los datos en la blockchain, la extensión de MetaMask se abre y solicita la confirmación de la transacción que se llevará a cabo para escribir estos datos en la cadena de bloques, el usuario puede confirmar la transacción o no. El usuario debe disponer de la cantidad de ethers suficiente para que la transacción se lleve a cabo. Sea que la transacción se realice o no, esto se notifica al usuario en su panel de control.

Cabe mencionar que ningún lote de café puede ser modificado por un usuario cuyo rol no corresponda con el rol actual del lote a modificar, esto justamente fue la funcionalidad que se codificó en los contratos inteligentes.

En cualquier caso, en el que un usuario quiera ingresar información sobre un lote de café y no haya iniciado sesión en la aplicación por medio de MetaMask, no podrá acceder a su panel de control y por ende a ningún formulario de ingreso de datos. Solicitándole que inicialice sesión con su billetera MetaMask y de no disponer de la extensión, se muestra un botón que abre una página de MetaMask para la instalación de la extensión y creación de una cuenta en la billetera.

El panel de control de los usuarios también muestra una tabla con los lotes de café que han sido intervenidos por el usuario. Esta tabla cuenta con la siguiente información:

- ID del lote, el cual cuenta con botones para enviar la URL con el ID por redes sociales y correo, para copiar la URL y para copiar únicamente el ID del lote.
- Código QR, el cual tiene botones que habilitan la descarga del código y la maximización de la imagen del QR.
- Proceso anterior, se especifica cuál fue el proceso que se intervino antes.
- Proceso actual, identifica cual es el proceso en el que se encuentra el lote de café y que aún no cuenta con información ingresada.
- Ver todo, es un ícono que abre una ventana emergente para visualizar todos los procesos del lote y su estado de intervención a través de una línea de tiempo.
- Tracking, es un ícono que redirige a la página de tracking con información sobre el historial de un lote en específico.

La página de Tracking tiene dos botones, uno que redirige al historial de un lote de ejemplo y otro que abre una ventana emergente para escanear un código QR, lo cual redirige a la página del historial de un lote en específico.

Cada página del historial de un lote muestra la información ingresada en la blockchain para cada etapa, así como cierta información sobre el usuario que llevó a cabo la intervención. La página también despliega un mapa en donde se visualizan los puntos de localización de las diferentes etapas de intervención.

Todas las páginas presentan enlaces a las otras dos vistas. El Anexo XXXVIII muestra los diagramas de flujo de los procesos de uso de la aplicación, para el usuario administrador, los usuarios modificadores y el consumidor.

2.5.3 DESPLIEGUE DE LA APLICACIÓN EN VERCEL

El alojamiento de la aplicación web creada es requerido para publicar la aplicación en Internet y que pueda ser accedida por los usuarios, para esto se empleó Vercel. Esta es una plataforma en la nube para implementaciones sin servidor [34]. El sistema desarrollado en sí no tiene servidor ya que los datos se almacenan en la blockchain, por lo que el uso de Vercel es adecuado. Vercel también facilita conectar el repositorio de código de GitHub del proyecto para implementar la rama principal con un dominio personalizado gratis. Además, la implementación es gratuita siempre que se trate de un proyecto personal y no comercial, como es el presente caso. En el Anexo XXXIX, se muestra el proceso para llevar a cabo la implementación del proyecto con Vercel.

El dominio bajo el cual se puede acceder a la aplicación es el siguiente: https://coffeetrack.vercel.app/.

EL código completo correspondiente a la interfaz de usuario o aplicación web se puede consultar a través del siguiente repositorio en GitHub: https://github.com/NathaliaBarreiros/coffee-supply-chain-client.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

3.1.1 CONTRATOS INTELIGENTES VERIFICADOS DESPLEGADOS EN LA BLOCKCHAIN TESTNET RINKEBY

Los 5 contratos inteligentes descritos en los apartados anteriores fueron verificados y alojados públicamente en la red testnet de Rinkeby, los contratos pueden ser consultados a través de las siguientes direcciones en el explorador de Etherscan. Todos los contratos fueron desplegados por la dirección de cuenta MetaMask del administrador del sistema.

Tabla 3.1. Dirección de consulta de los contratos desplegados en la testnet de Rinkeby

Contrato	Dirección de Contrato
SupplyChainStorage	0x8d634Bf42F06d7904dD46D8Ed4Ab345230A03b7d (https://rinkeby.etherscan.io/address/0x8d634bf42f06d7904dd46d8ed4ab345230a03b7d)
SupplyChainStorage2	0x1e3DDD059f0F01258A7baC74dfBA2cA43C33A449

	(https://rinkeby.etherscan.io/address/0x1e3DDD059f0F01258A7baC74dfBA2cA43C33A449)
CoffeeSupplyChain	0xdf0C594655C466B0b37CeFc519f38Ea8fEB465F9 (https://rinkeby.etherscan.io/address/0xdf0C594655C466B0b37CeFc519f38Ea8fEB465F9)
CoffeeSupplyChain2	0xcf76465C29A32F11D6A27a009eE7CB500669c5Ff (https://rinkeby.etherscan.io/address/0xcf76465C29A32F11D6A27a009eE7CB500669c5Ff)
SupplyChainUser	0xbf87Fd7e3416311dbef4F00e2ce73950A0F2a0D2 (https://rinkeby.etherscan.io/address/0xbf87Fd7e3416311dbef4F00e2ce73950A0F2a0D2)

3.1.2 PRUEBAS REALIZADAS DEL SISTEMA DE TRACKING CON LA APLICACIÓN WEB

Los resultados mostrados a continuación corresponden al inicio de sesión en la aplicación de trazabilidad CoffeeTrack, al ingreso de los datos con respecto a las diferentes etapas de intervención de un lote de café a lo largo de la cadena de suministro, por parte del usuario con el rol correspondiente. Así también, se visualizará el ingreso de usuarios en el sistema, como la actualización de los datos de perfil por parte de un usuario y la consulta del lote de café una vez culminada su intervención, lo que corresponde a la consulta que realizaría un consumidor con el código QR impreso en la funda de su café.

3.1.2.1 Inicio de Sesión en la Aplicación de Trazabilidad CoffeeTrack

Para iniciar sesión en la aplicación, se debe acceder a la URL https://coffeetrack.vercel.app y seleccionar la pestaña Dashboard, esto redirige a la página de Login de la aplicación, esto se muestra en la Figura 3.1.

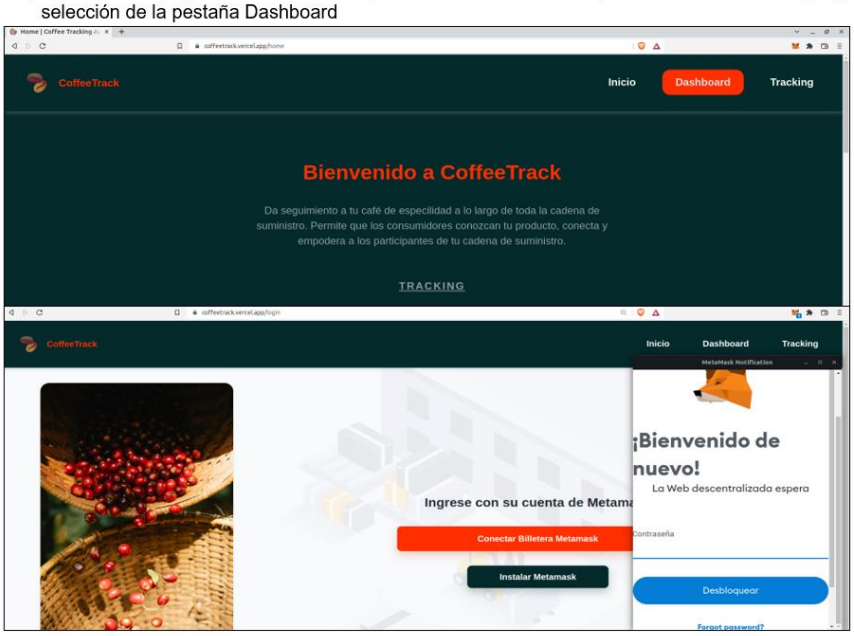
Existen 3 posibilidades al intentar iniciar sesión, las cuales se exhiben en la Figura 3.2:

a) El usuario tiene instalada la extensión de MetaMask en su navegador y se inicia sesión con su cuenta de billetera MetaMask. Para lo cual se abre la extensión de MetaMask, se ingresa a la cuenta de la billetera del usuario y se selecciona el botón Conectar Billetera MetaMask en la aplicación. Si la cuenta con la que inicia sesión está registrada en el sistema, se le redirige a su panel de control.

b) El usuario tiene instalada la extensión de MetaMask en su navegador, pero la cuenta con la que intenta iniciar sesión no está registrada en el sistema, se le notifica que su cuenta no está registrada y se solicita comunicarse con el administrador por correo.

c) El usuario no tiene instalada la extensión de MetaMask, se le solicita instalar MetaMask para iniciar sesión. Entonces el usuario puede seleccionar el botón de Instalar MetaMask lo que abre la página de instalación de MetaMask.

a) Ingreso a la página de CoffeeTrack por medio de la URL https://coffeetrack.vercel.app/ y selección de la pestaña Dashboard

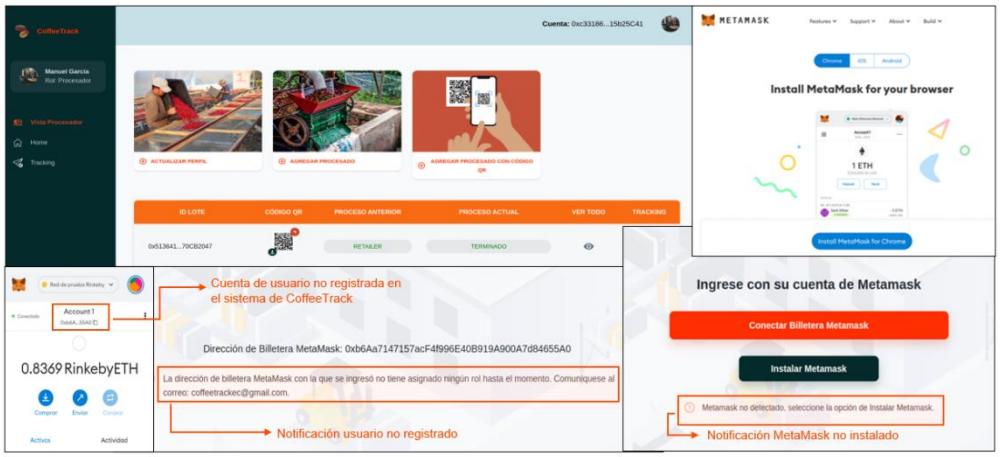


b) Redirección a la página de Login para iniciar sesión con MetaMask

Figura 3.1. Ingreso a la página de CoffeeTrack y redirección a la página de Login

a) Inicio de sesión con MetaMask exitoso: usuario registrado en el sistema y redirección al Dashboard del usuario

d) Página de instalación de MetaMask



b) Inicio de sesión con MetaMask no exitoso: usuario no registrado en el sistema y solicitud de contactarse al correo de CoffeeTrack

c) Extensión de MetaMask no instalada en el navegador: conexión fallida con la aplicación CoffeeTrack

Figura 3.2. Posibilidades al iniciar sesión en CoffeeTrack con MetaMask

3.1.2.2 Prueba del Registro de Granja y Creación de Nuevo ID de Lote

En la Figura 3.3, se muestra el registro de una granja de cafeto lo que genera la creación de un nuevo ID de lote de café que se lleva a cabo en el panel de control del administrador del sistema, para lo cual se llena la información en el formulario mostrado y los datos referentes a la ubicación de la granja se establecen automáticamente una vez seleccionado el botón para BUSCAR EN MAPA y colocado el marcador en la ubicación deseada. Una vez completada la información, se selecciona el botón de REGISTRAR GRANJA, esto abre la extensión de MetaMask, la cual solicita la confirmación de la transacción. Una vez

confirmada la transacción y validada en la blockchain Rinkeby, se notifica al usuario en el panel de control. En este caso, el lote con ID 0x286af1...d1D70608 fue creado.

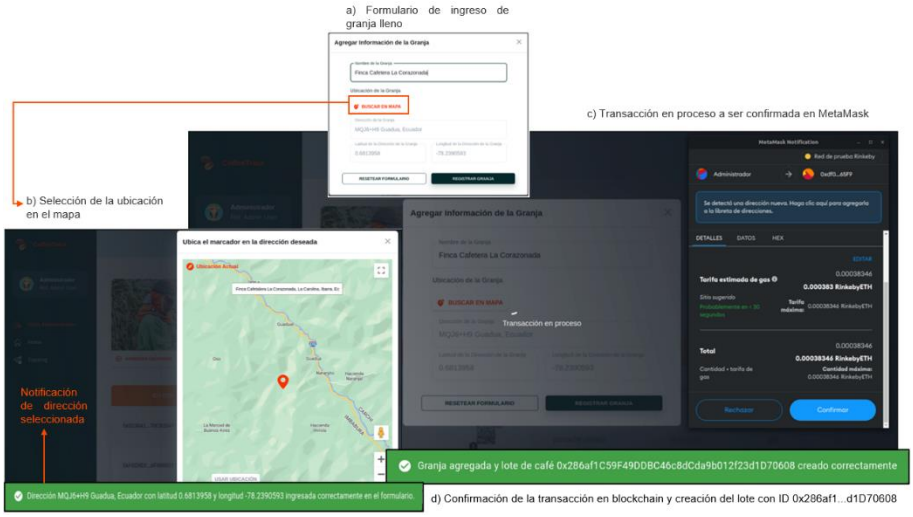


Figura 3.3. Formulario de registro de granja, transacción en proceso en MetaMask y confirmación de transacción realizada en blockchain

Cuando se crea el nuevo lote, este se visualiza en la tabla desplegada en el panel de control del administrador. A continuación, el administrador deberá compartir la información del ID del lote con el siguiente usuario que es el agricultor de la granja ingresada y vendedor de su lote de café. En la Figura 3.4, se visualizan las opciones para compartir el ID, en este caso se envía la URL del lote en un mensaje por Telegram. El ícono de la columna VER TODO de la tabla, permite visualizar la línea de tiempo de los procesos llevados a cabo en el lote específico. Así, el ingreso de la granja se ha realizado y se encuentra en proceso la cosecha y el ingreso de su información en el sistema.

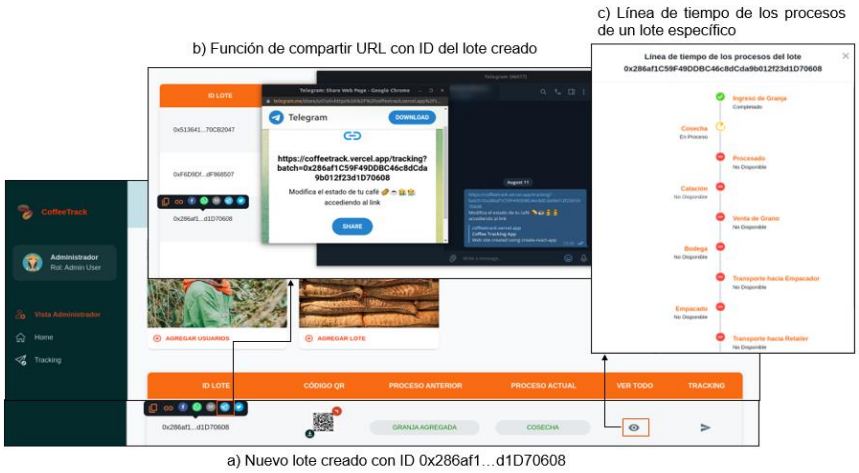


Figura 3.4. Lote nuevo mostrado en tabla y envío de la URL del lote por Telegram

En la Figura 3.5 se muestra la página de redirección al dar clic en el ícono de la columna TRACKING para el lote específico. Esta página exhibe la información registrada sobre el lote, en este caso solo se encuentran disponibles los datos de registro de granja, también se puede visualizar la transacción en el explorador Etherscan.

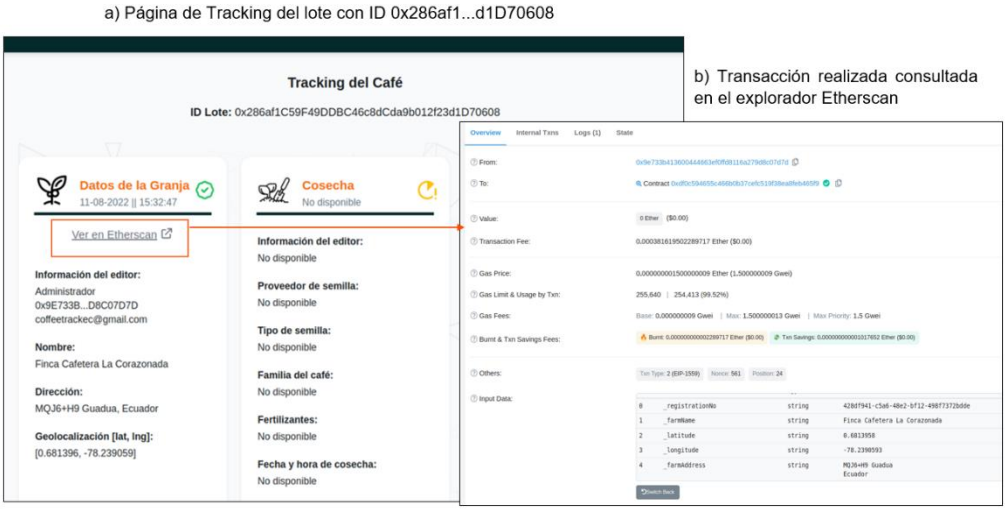


Figura 3.5. Página de Tracking del lote creado y consulta de la transacción en Etherscan

3.1.2.3 Prueba del Registro de Cosecha

En la Figura 3.6 se muestra la redirección realizada a partir de la URL enviada por Telegram desde el administrador al agricultor. Debido a que el administrador no tiene el producto físico, lo más conveniente es el envío de la URL. La página de redirección corresponde a la del resumen de trazabilidad del lote de café específico, esta tiene habilitada el botón de REGISTRAR datos de la etapa correspondiente, al dar clic en el botón, este reenvía al formulario para ingresar los datos de intervención, siempre y cuando el usuario haya iniciado sesión en la página web previamente con MetaMask, de lo contrario, se reenvía a la página de Login y se realiza el proceso del apartado 3.1.2.1. Si el rol del usuario no corresponde con el rol actual del lote, no se muestra el formulario y se notifica al usuario.

La redirección a partir de la URL de un lote específico y el inicio de sesión es similar para todos los demás usuarios con los diferentes roles y el formulario al que se redirige al usuario, tiene el ID de lote pre seteado. En el caso de que se use un código QR del lote presente en el producto físico, este también redirige a la misma URL de Tracking del lote específico en el cual se va a ingresar los datos.

La Figura 3.7 muestra la transacción en proceso llevada a cabo una vez llenado el formulario y seleccionado el botón de AGREGAR DATOS, la extensión de MetaMask se abre y solicita confirmación. Cuando se confirma la transacción, se notifica al usuario.

La Figura 3.8 muestra el nuevo lote de café intervenido por el usuario con rol de agricultor en la tabla de su panel de control. A continuación, el agricultor debe transmitir el ID del lote, en este caso lo realiza por medio de la descarga del código QR del lote para imprimirlo en el producto físico a ser enviado al procesador. También se puede enviar la URL con el ID de lote por una red social o copiar el ID únicamente y enviarlo. El ícono de la columna VER TODO despliega la línea de tiempo de los procesos del lote específico, se observa que el proceso de cosecha ya terminó y el procesado está en espera. El ícono de la columna de TRACKING redirige a la página de trazabilidad del lote donde se muestran los datos de los procesos realizados y se puede consultar la transacción realizada en Etherscan.

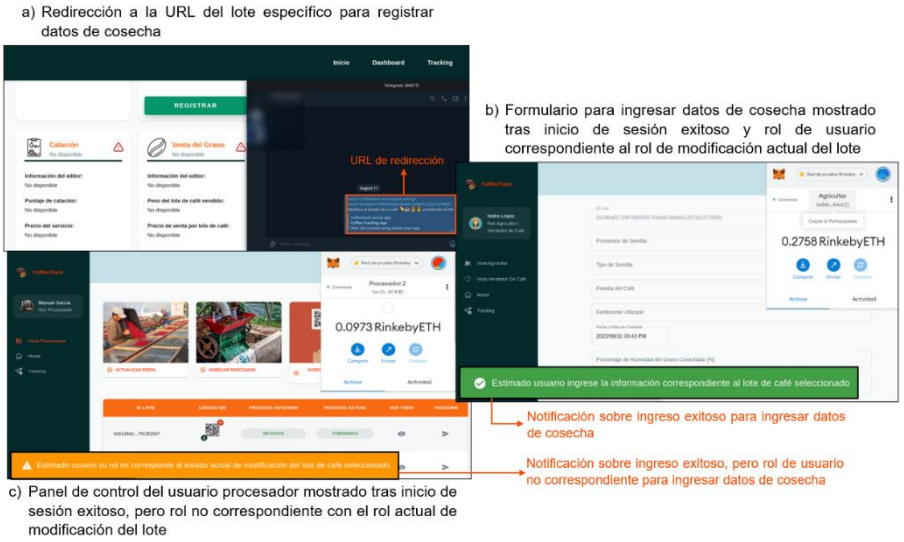


Figura 3.6. Redirección a la URL del lote específico para registrar datos de cosecha

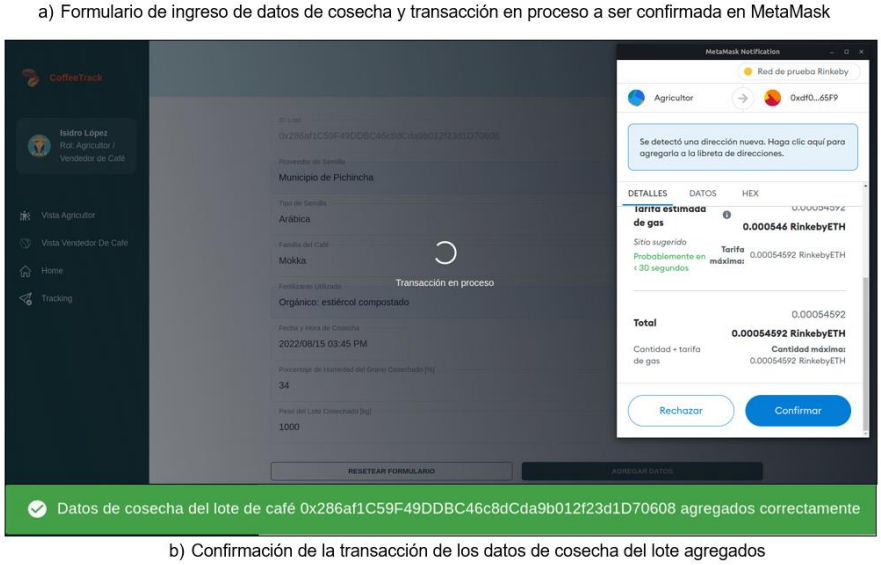


Figura 3.7. Formulario de datos de cosecha, transacción en proceso y confirmación

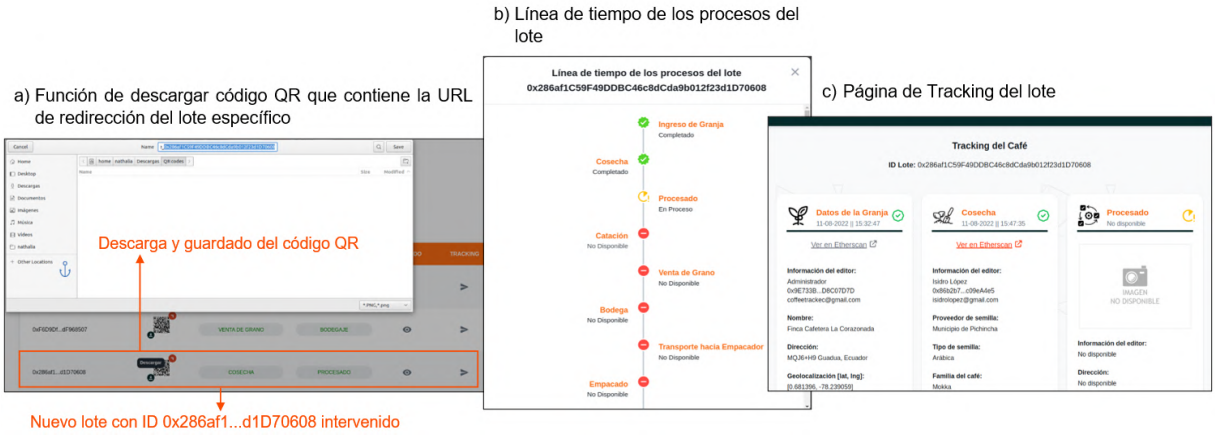


Figura 3.8. Lote mostrado en tabla tras la confirmación de transacción de cosecha

3.1.2.4 Prueba del Registro de Procesado

En este caso se muestra el ingreso de información de procesado a partir del escaneo de un código QR cuando el usuario ya ha ingresado a su panel de control con su cuenta MetaMask. La Figura 3.9 muestra el panel de control donde el usuario procesador elije la opción de AGREGAR PROCESADO CON CÓDIGO QR, esto abre el lector QR y al mostrar el código en la cámara, se habilita el botón para ingresar datos en el formulario si el código leído es correcto, es decir el rol de modificación del lote es igual al del procesador y se notifica al usuario. De lo contrario se le notifica que el código no es válido, manteniendo deshabilitado el botón que permite dirigirse al formulario.

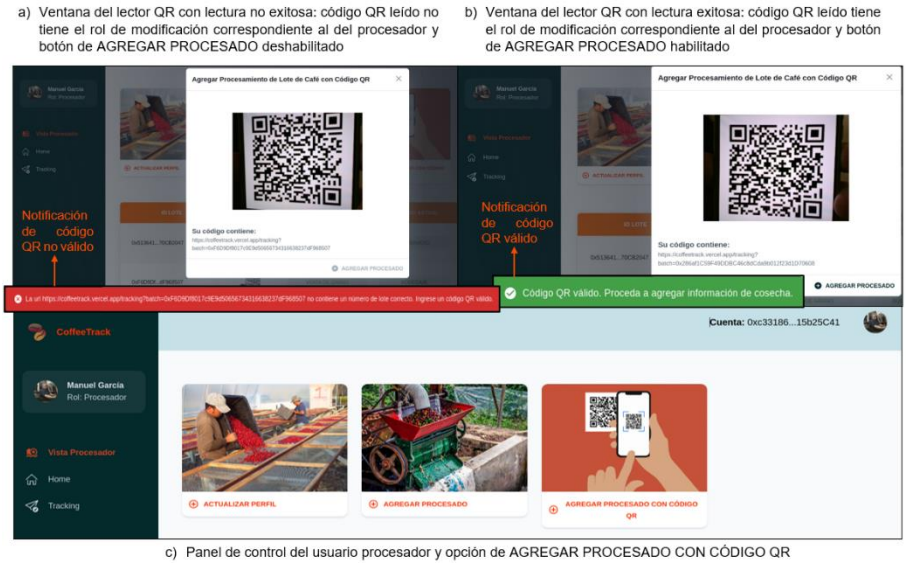
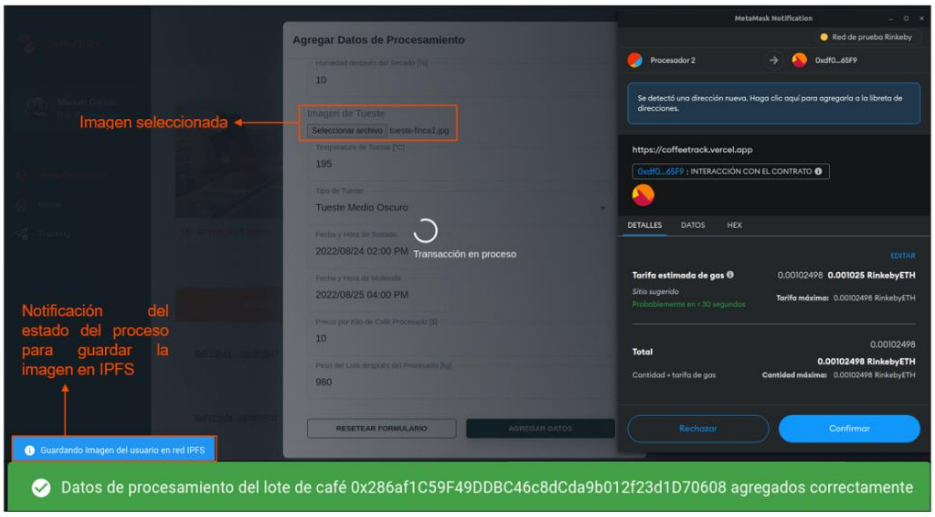


Figura 3.9. Ingreso de datos de procesado a través de código QR

La Figura 3.10 muestra el formulario para ingresar datos del procesado, entre estos el campo para seleccionar la imagen de tueste la cual será enviada a la red descentralizada de IPFS. Cuando los campos son completados, se habilita el botón para agregar datos y

se notifica sobre el estado del proceso de envío de la imagen a IPFS, si el proceso es correcto se abre la extensión de MetaMask para confirmar la transacción, luego se notifica la confirmación de la transacción al usuario.

a) Formulario, transacción en proceso a ser confirmada en MetaMask y notificación del proceso de almacenamiento de imagen en la red IPFS



b) Confirmación de la transacción de los datos de procesado agregados

Figura 3.10. Formulario de datos de procesado, transacción en proceso y confirmación

En la Figura 3.11, se visualiza el nuevo lote de café intervenido mostrado en la tabla del panel de control del usuario procesador, la opción de copiar únicamente el ID del lote para algún caso requerido, así como la ventana emergente de la línea de tiempo de los procesos del lote con el procesado terminado y la catación en espera, y la redirección a la página de Tracking del lote donde la información de procesado ya se muestra.

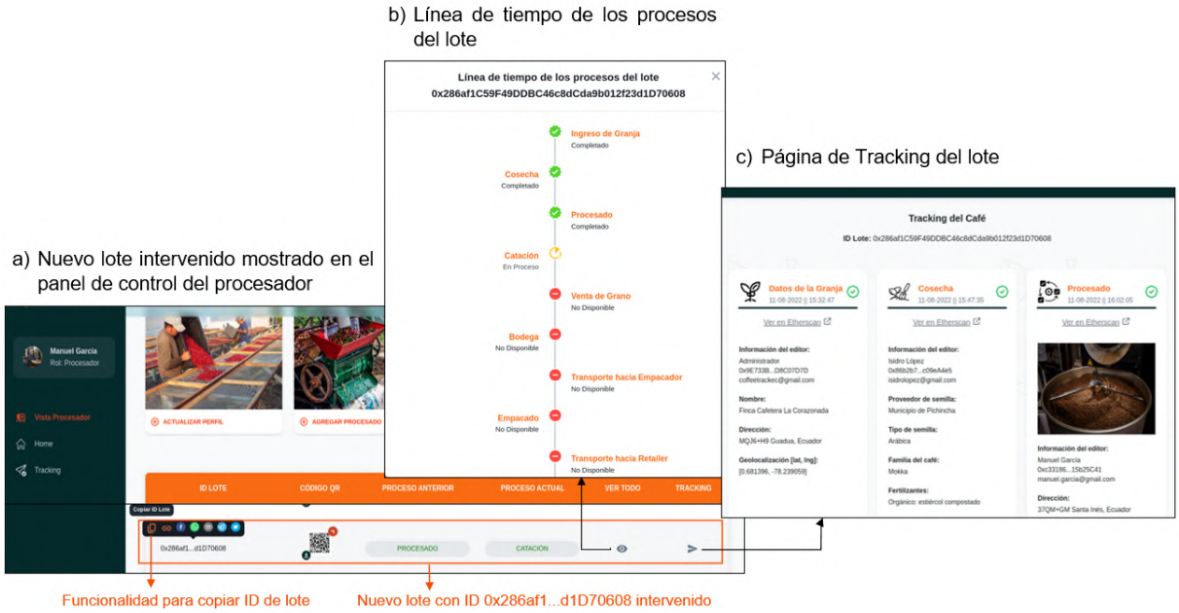


Figura 3.11. Lote mostrado en tabla tras la confirmación de transacción de procesado

3.1.2.5 Prueba del Registro de Catación

En la Figura 3.12 se muestra el ingreso de los datos de catación sin código QR ni URL de redirección a través de la opción AGREGAR CATACIÓN, lo que abre un formulario cuyo campo de ID de lote no está pre seteado y el usuario modificador debe ingresar el valor, esta es la primera opción de ingreso de datos descrita en el apartado 2.5.2. También se muestra la transacción en proceso de confirmación por parte de MetaMask.

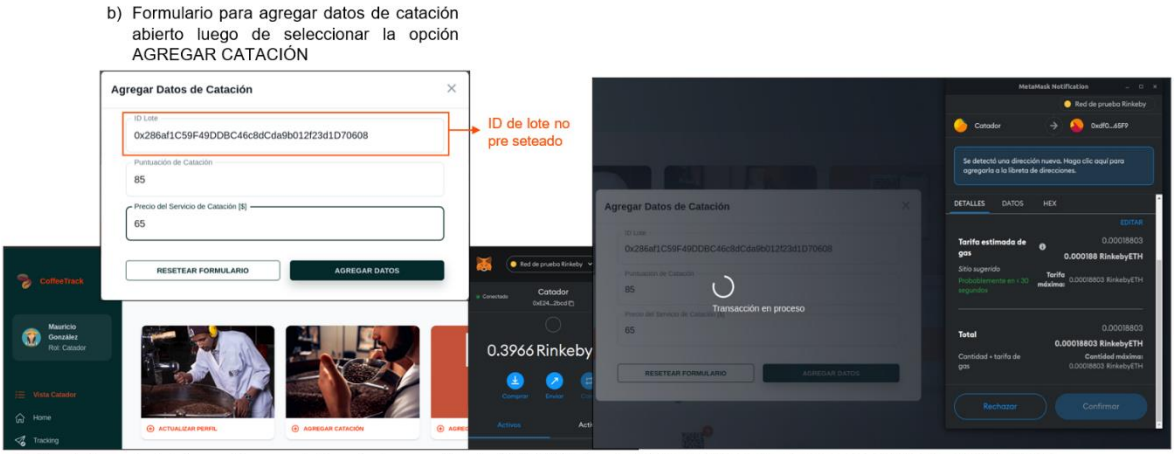


Figura 3.12. Formulario de datos de catación y transacción en proceso de confirmación

La Figura 3.13 muestra el nuevo lote intervenido en la tabla del panel de control del catador tras la confirmación de la transacción, así como la línea de tiempo de los procesos del lote con la catación terminada y la venta de grano en espera, y la página de Tracking con los datos de catación ya actualizados.

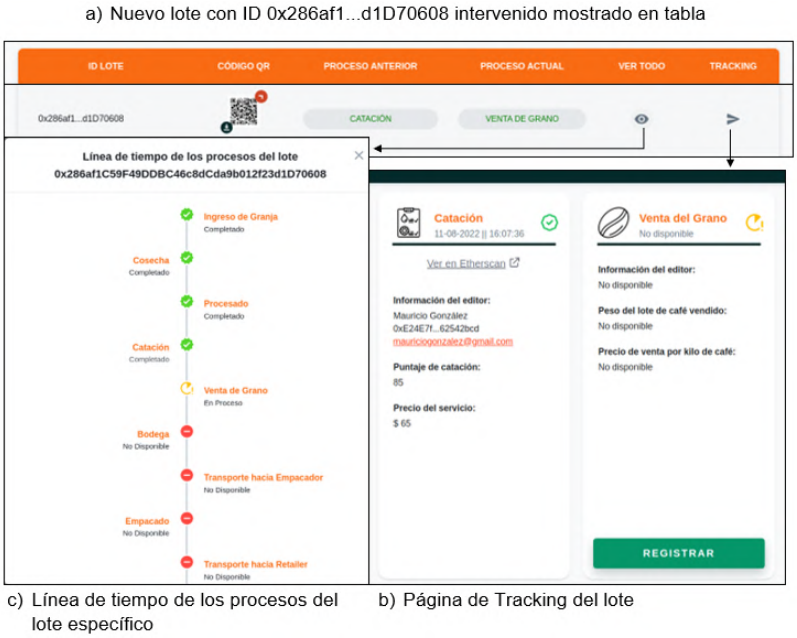


Figura 3.13. Lote mostrado en tabla tras la confirmación de transacción de catación

3.1.2.6 Prueba del Registro de Venta del Lote de Café

En la Figura 3.14 se muestra el proceso de redirección al formulario de venta de café a partir de una URL que pudo ser enviada por una red social u obtenida del escaneo del código QR del producto físico y la transacción en proceso tras llenar el formulario. Cabe mencionar que el usuario que realiza la venta del lote de café es el mismo agricultor, por lo que el panel de control muestra dos roles para este usuario.

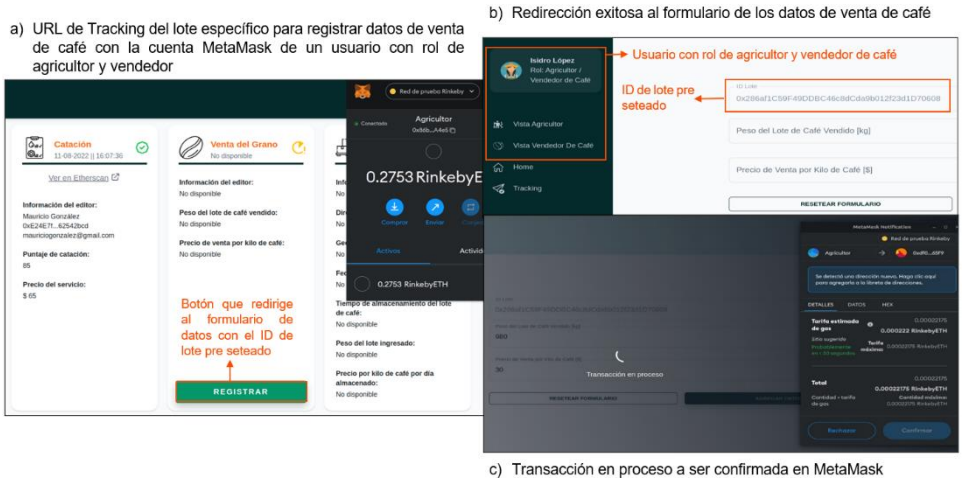


Figura 3.14. Formulario de venta de café y transacción en proceso de confirmación

La Figura 3.15 muestra el lote de café intervenido agregado en la tabla del panel de control del usuario, la línea de tiempo de procesos con el proceso de venta de café completado y la página de Tracking del lote con los datos de venta de café actualizados.

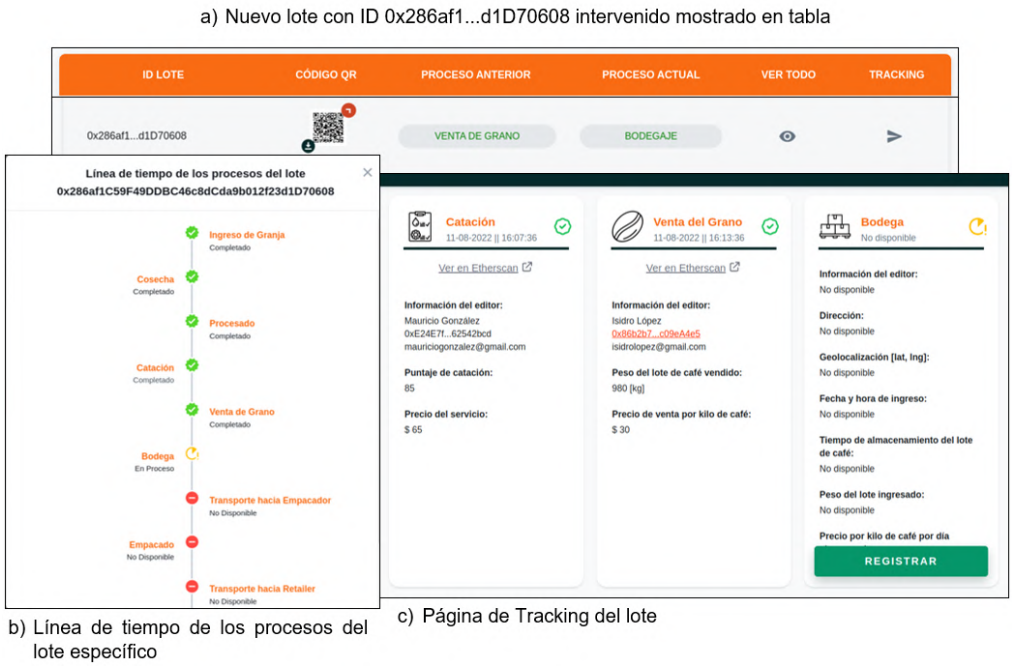


Figura 3.15. Lote mostrado en tabla tras la confirmación de transacción de venta de café

3.1.2.7 Prueba del Registro de Bodegaje

En la Figura 3.16 se muestra la transacción en proceso, la confirmación de la transacción, el nuevo lote intervenido mostrado en la tabla del usuario, la página de Tracking con los nuevos datos actualizados de la etapa de bodegaje.

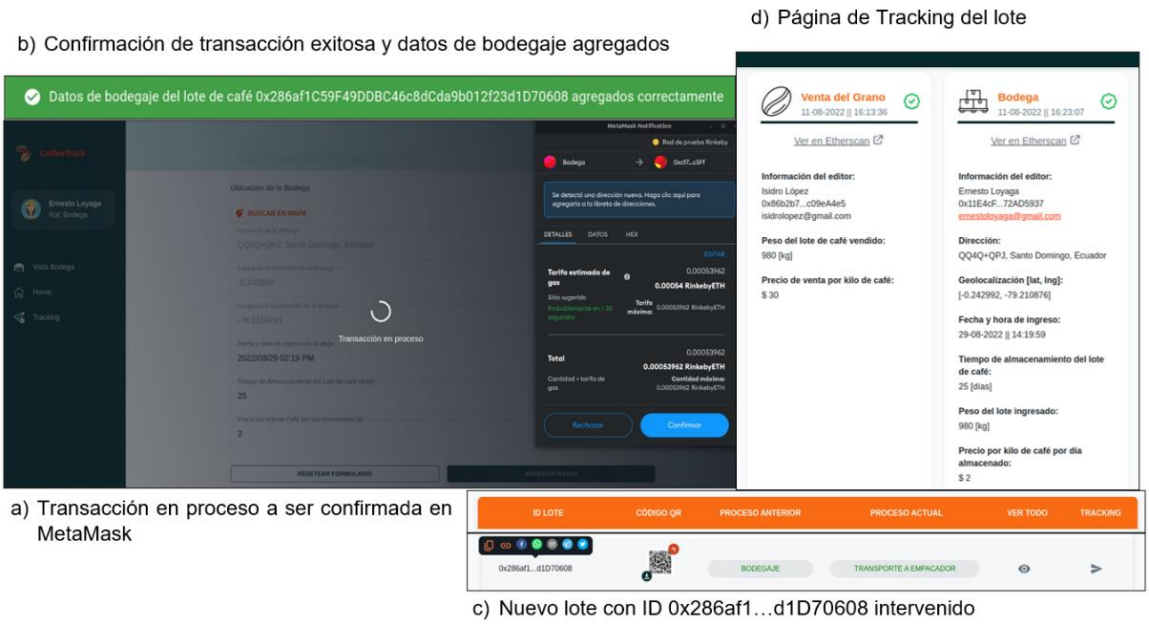


Figura 3.16. Registro de la etapa de bodegaje

3.1.2.8 Prueba del Registro de Transporte a Empacador

En la Figura 3.17 se muestra la transacción en proceso, la confirmación de la transacción, el nuevo lote intervenido mostrado en la tabla del usuario, la página de Tracking con los nuevos datos actualizados de la etapa de transporte a empacador.

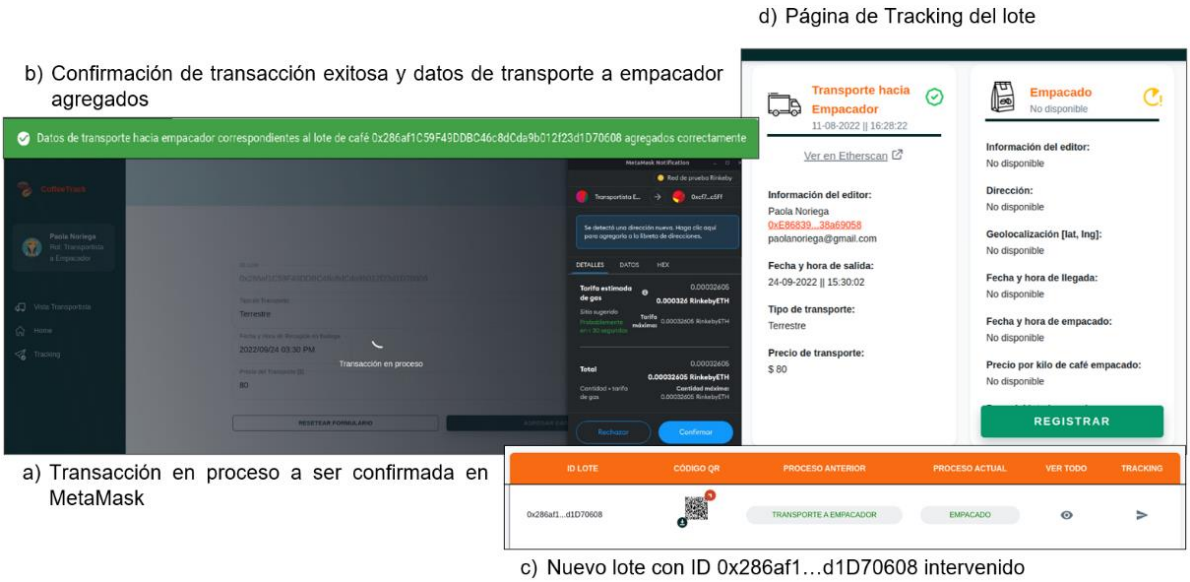


Figura 3.17. Registro de la etapa de transporte a empacador

3.1.2.9 Prueba del Registro de Empacado

En la Figura 3.18 se muestra la transacción en proceso, la confirmación de la transacción, el nuevo lote intervenido mostrado en la tabla del usuario, la página de Tracking con los nuevos datos actualizados de la etapa de empacado.

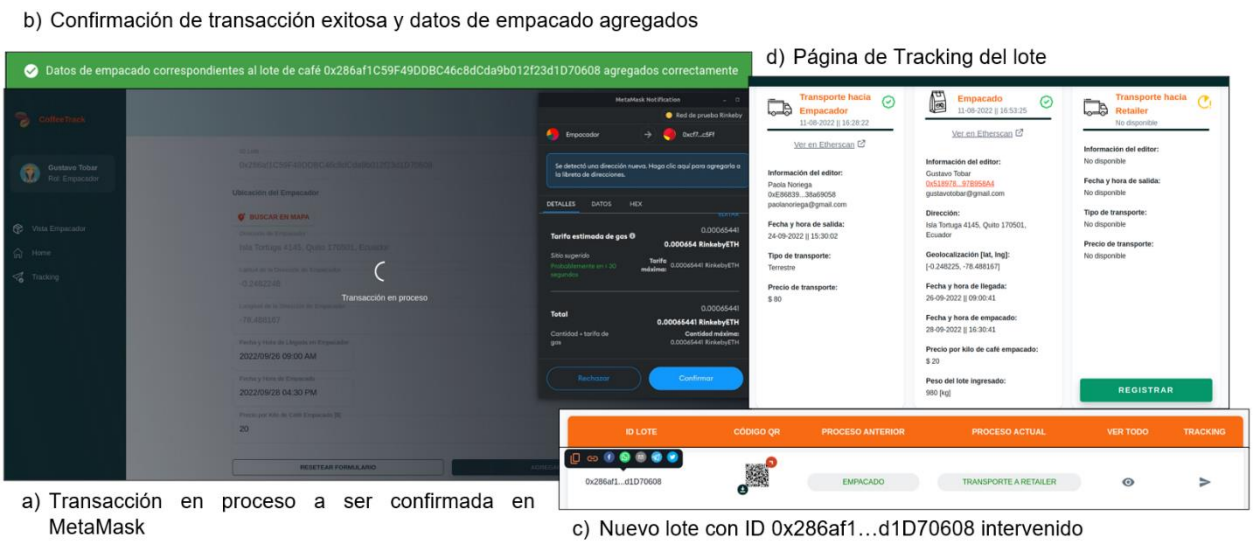


Figura 3.18. Registro de la etapa de empacado

3.1.2.10 Prueba del Registro de Transporte a Retailer

En la Figura 3.19 se muestra la transacción en proceso, la confirmación de la transacción, el nuevo lote intervenido mostrado en la tabla del usuario, la página de Tracking con los nuevos datos actualizados de la etapa de transporte a retailer.

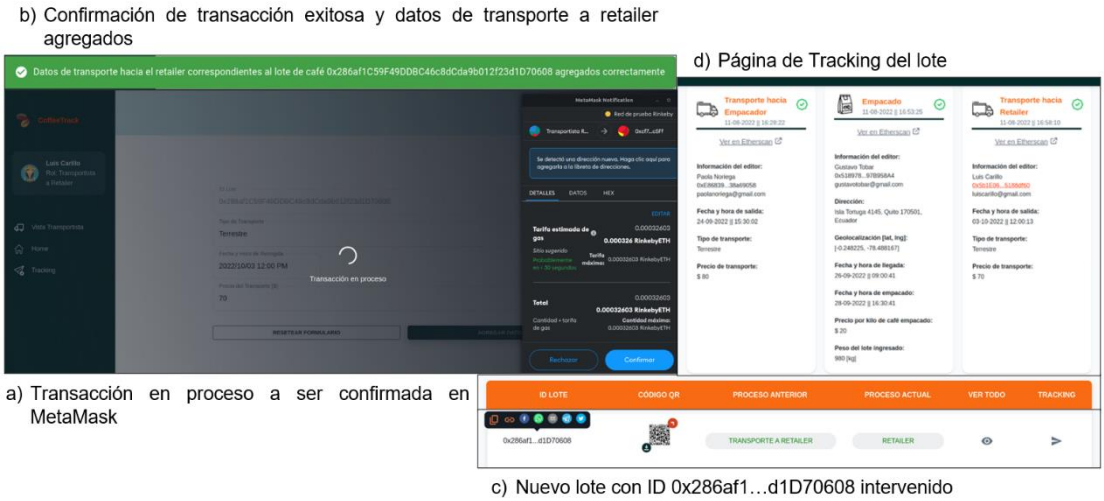


Figura 3.19. Registro de la etapa de transporte a retailer

3.1.2.11 Prueba del Registro de Comercialización en Retailer

En la Figura 3.20 se muestra la transacción en proceso, la confirmación de la transacción, el nuevo lote intervenido mostrado en la tabla del usuario, la página de Tracking con los nuevos datos actualizados de la etapa de comercialización en retailer.

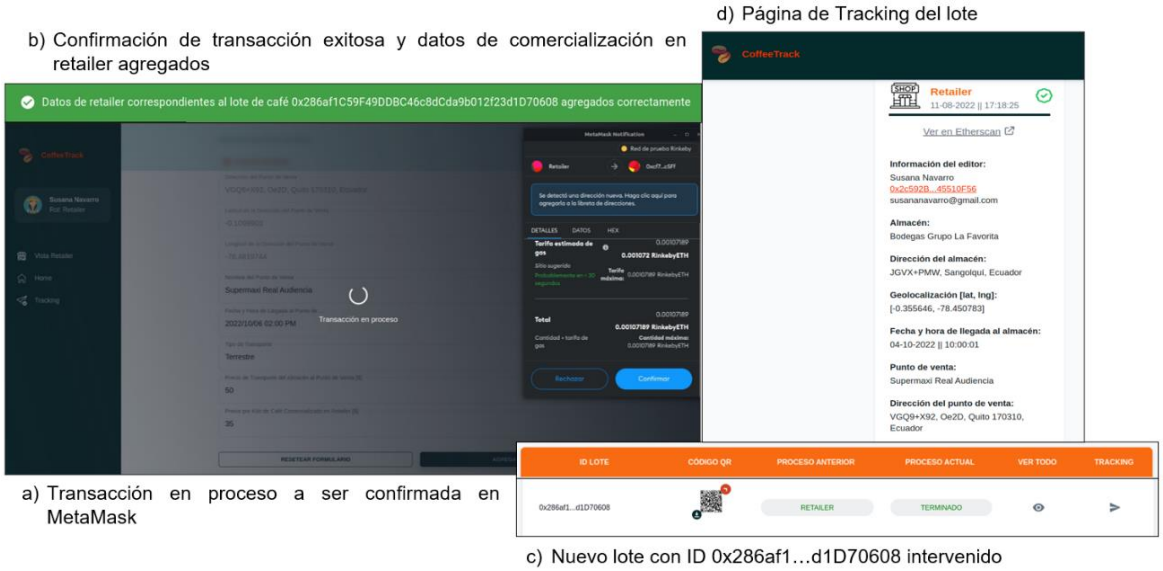


Figura 3.20. Registro de la etapa de comercialización en retailer

3.1.2.12 Prueba del Registro de Usuarios

La Figura 3.21 exhibe la opción de AGREGAR USUARIOS en el panel de control del administrador, el formulario con los datos del usuario Manuel García, la transacción en proceso y la notificación de la transacción exitosa y por ende el ingreso del nuevo usuario en el sistema. En la Figura 3.22, se observa el caso de intento de ingreso de la cuenta de un usuario ya existente en el sistema, en donde la aplicación lo detecta y no permite que la extensión de MetaMask se abra para llevar a cabo la transacción, así como se notifica al administrador de que la cuenta ya fue registrada para otro usuario.

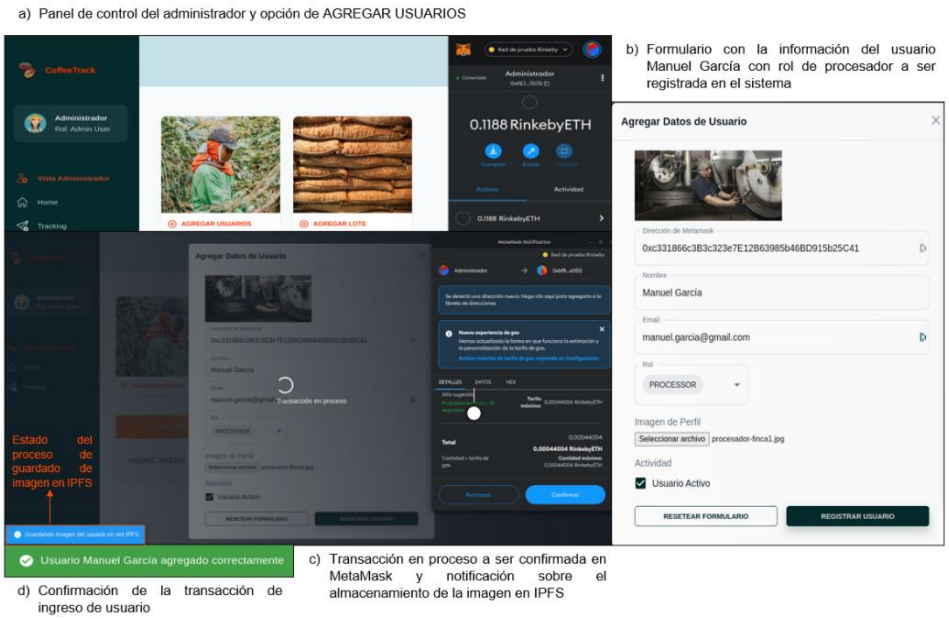
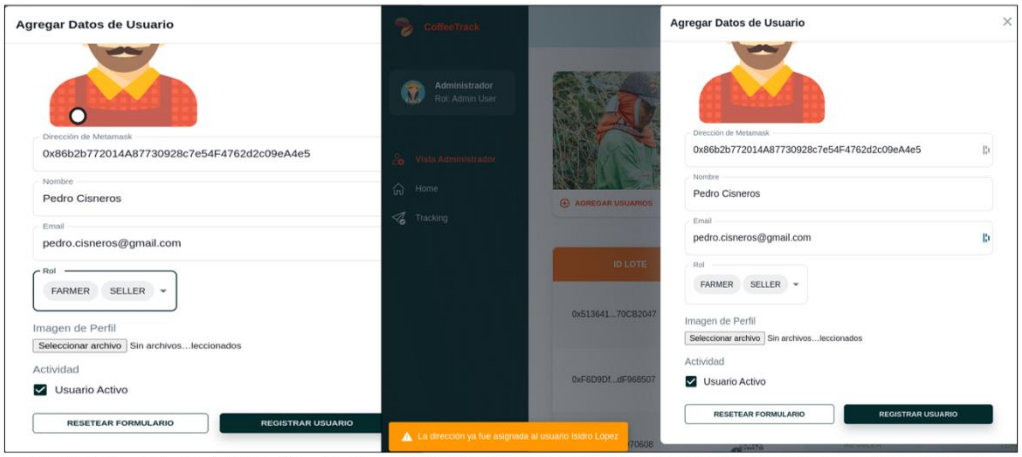


Figura 3.21. Registro de un usuario nuevo en el sistema



a) Formulario con la información de un usuario con cuenta de MetaMask 0x86b277…9eA4e5

b) Notificación sobre dirección de usuario ya asignada a otro usuario en el sistema, no se permite realizar la conexión con MetaMask y llevar a cabo la transacción

Figura 3.22. Registro de usuario fallido al intentar ingresar cuenta de usuario ya existente

3.1.2.13 Prueba de la Actualización de Perfil

La Figura 3.23 exhibe la opción de ACTUALIZAR PERFIL en el panel de control del usuario Mauricio González con rol de catador, el formulario con los datos de perfil anteriores del usuario, el formulario con los datos de correo e imagen de perfil a ser actualizados, la transacción en proceso, la notificación de la transacción exitosa y por ende la actualización de los datos de perfil (imagen y correo) en el sistema.

3.1.2.14 Prueba de la Consulta de Trazabilidad de una Funda de Café de Especialidad a través de su Código QR

La Figura 3.24 muestra la sección de Tracking de la aplicación que permite al consumidor escanear el código QR presente en su producto para conocer los datos de trazabilidad del lote al que pertenece su café de especialidad. También, se observa las lecturas del código QR realizadas en la aplicación de CoffeeTrack y en otra aplicación, las cuales devuelven la URL de consulta que redirige a la página específica de Tracking del lote de café. En la Figura 3.25, se observa la página de Tracking con los datos de los primeros procesos de modificación, así como el mapa que permite visualizar los lugares de intervención por los que ha pasado el lote de café consultado.

El video de funcionamiento del sistema correspondiente a las pruebas realizadas, así como los enlaces correspondientes a los entregables del proyecto se pueden consultar en el Anexo XLI.

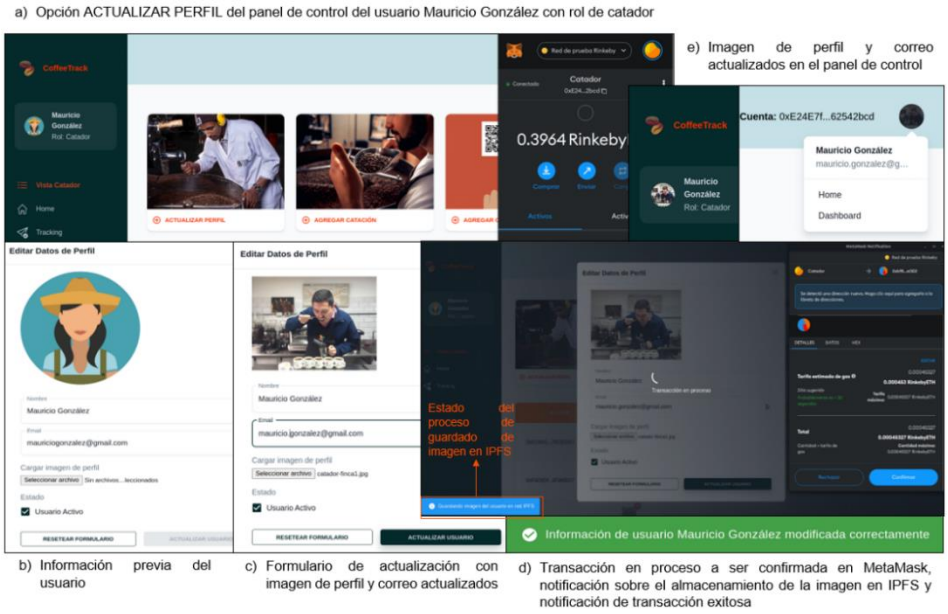


Figura 3.23. Actualización de datos de perfil de un usuario con rol de catador

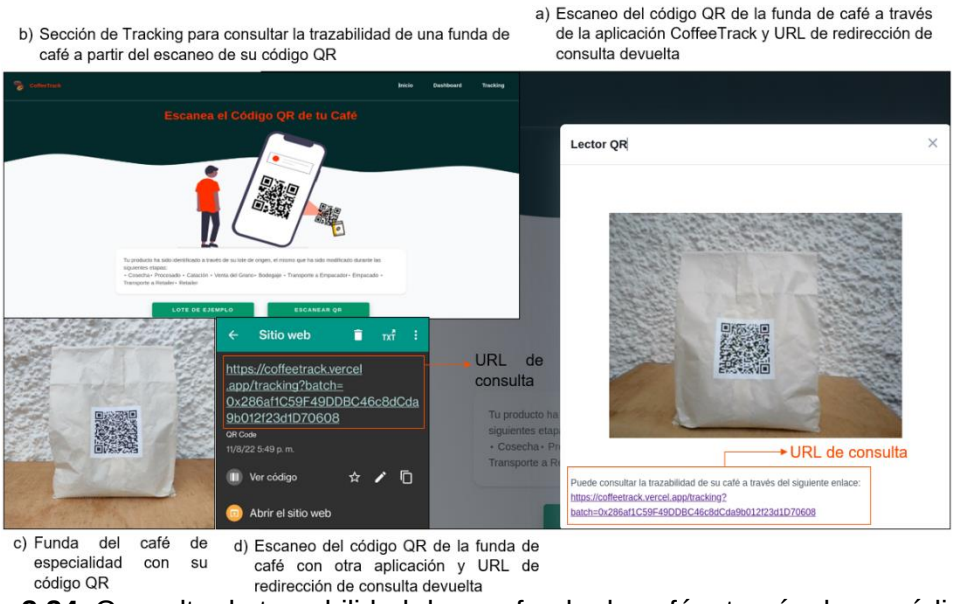


Figura 3.24. Consulta de trazabilidad de una funda de café a través de su código QR

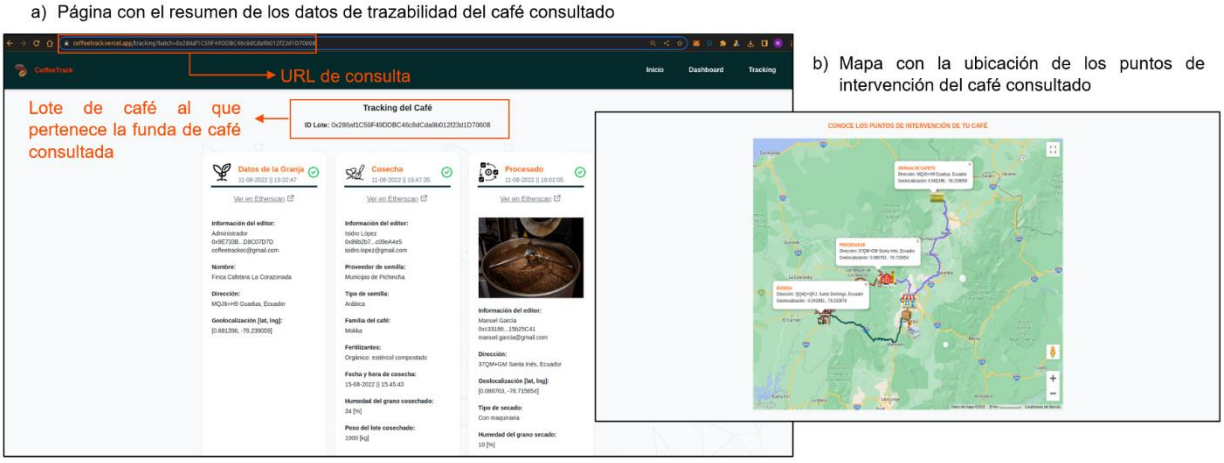


Figura 3.25. Página de trazabilidad del lote al que pertenece el café consultado

3.2 CONCLUSIONES

- Se diseñó y simuló un sistema de trazabilidad para la cadena de suministro agrícola del café de especialidad, utilizando la cadena de bloques de prueba Rinkeby, del ecosistema de la blockchain de Ethereum, para la implementación y ejecución de los contratos inteligentes, los cuales permiten automatizar el proceso de registro de los datos de trazabilidad del lote de café como unidad trazable a lo largo de las diferentes etapas de intervención en la red de suministro, para lo cual se desarrolló la aplicación web CoffeeTrack que permite la interacción de los usuarios con los contratos inteligentes alojados en la blockchain, por medio de la extensión de la billetera digital MetaMask instalada en el navegador web.

- La aplicación web desarrollada además de permitir el registro de datos en el sistema por parte de los participantes modificadores, habilita la opción de consulta del historial de trazabilidad de una funda de café de especialidad para los consumidores, con el fin de conocer la información sobre el lote de origen de su café a través del escaneo del código QR presente en la funda.

- En el caso de los pequeños productores de café de especialidad, las micro y medianas empresas que intervienen en las diferentes etapas de transformación del café, el sistema de trazabilidad desarrollado basado en blockchain, tiene el potencial de brindarles la oportunidad de tener una participación y visibilidad más amplia en el mercado por medio de transparentar la información de su cadena de suministro a los consumidores con tan solo la consulta de un código QR, evitando el requerimiento de intermediarios que manejen su información.

- El uso de la tecnología de la cadena de bloques en el sistema de trazabilidad de la cadena de valor agrícola del café de especialidad que se llevó a cabo, permite dar identidad digital a un producto físico y conectarlo tanto con los consumidores como con los participantes modificadores para que estos puedan consultar la información del café de manera inmediata y compartida, a diferencia de los sistemas tradicionales de trazabilidad donde la información no está disponible a lo largo de toda la cadena de valor, para todas las partes interesadas y de manera inmediata.

- Pese a que el sistema de trazabilidad desarrollado tiene un administrador, el manejo y el almacenamiento de la información se realiza de manera descentralizada, debido al uso de blockchain, lo que garantiza que la validación del ingreso de información en el sistema se lleve a cabo por un algoritmo de consenso entre los nodos de la red peer-to-peer y no únicamente por una entidad centralizada. Lo anterior provee una ventaja

competitiva en términos de transparencia y confianza por parte del consumidor para con su producto, que justamente es lo que necesita un producto como el café de especialidad para ser impulsado.

- El sistema de trazabilidad desarrollado permitió conectar la información de todos los participantes de la cadena de suministro del café de especialidad, lo cual resuelve uno de los desafíos que presentan las cadenas de valor agrícolas, ya que normalmente los sistemas de trazabilidad no ofrecen marcos de trazabilidad completa por la dificultad de interacción entre los participantes y la integración de sus datos bajo una misma tecnología.

- El sistema de tracking con blockchain permite que la información registrada en la red no pueda modificarse en el futuro, además los datos almacenados no son corrompibles por ataques de un solo nodo.

- Mediante el entorno de pruebas ejecutado para los contratos inteligentes, se pudo validar el correcto funcionamiento de la lógica implementada en estos y verificarlos para que su código fuente en Solidity esté disponible para todos los usuarios que deseen consultarlo a través del explorador Etherscan y que estos tengan seguridad de cómo se está llevando el manejo de sus datos. Así también, las pruebas descritas en el capítulo anterior permitieron validar el funcionamiento del sistema completo.

- Tanto el código fuente de los contratos inteligentes como de la aplicación web publicados abiertamente en los repositorios de GitHub del proyecto, permiten que futuros desarrolladores colaboren e implementen soluciones en base al trabajo realizado.

- Se logró entender los conceptos y las ventajas que provee la tecnología de la cadena de bloques para generar una alternativa en el marco de la trazabilidad de las cadenas de suministro, solventando las necesidades de estas en los temas de transparencia, confianza, descentralización en el almacenamiento de datos e interconexión entre los participantes de la cadena de valor.

- Se logró identificar los participantes de la cadena de valor agrícola del café de especialidad, así como las variables a ser rastreadas, las cuales brindan información sobre el origen del café, la ubicación del lote de café a lo largo de sus etapas de transformación, las características que describen estos procesos de transformación y el precio por unidad de peso que conlleva dichas intervenciones. Todo esto desde el

punto de vista de un flujo lineal de la cadena de suministro en donde no se consideran ni bifurcaciones ni uniones del lote de café inicialmente identificado.

3.3 RECOMENDACIONES

- Se recomienda que el administrador del sistema que implemente los contratos en la cadena de bloques real y realice el ingreso de las granjas de cultivo de café como de los usuarios modificadores en el sistema, sea una parte interesada en fortalecer el ecosistema de la comercialización del café de especialidad en Ecuador y de su cadena de valor, como puede ser la Asociación de Cafés Especiales del Ecuador (ACEDE).

- Se recomienda que se utilice la cadena de bloques Polygon para el despliegue real de los contratos inteligentes en una red pública y sin permisos. Esto debido a que la blockchain de Polygon presenta menos congestión en su red, permitiendo que la implementación y ejecución de los contratos tenga un menor costo en la tarifa de gas que se debe pagar para realizar las transacciones.

- Se recomienda como una perspectiva futura del presente trabajo, la integración de la tecnología IoT para automatizar totalmente la escritura de los contratos inteligentes, por medio de la obtención de información directamente de una red de sensores inteligentes, evitando el ingreso de datos por parte de los usuarios.

- También se puede pensar en escalar el proyecto para integrar la trazabilidad de datos con mayor granularidad sobre el proceso para usar la información para hacer análisis de datos, predicciones, tracking inverso, etc., todo esto enfocado no tanto en la información que se brinda a los consumidores sobre la idoneidad del producto, sino más bien desde una perspectiva de mejorar y agilizar la cadena de suministro de valor del café de especialidad. Así también, se podría considerar integrar en el sistema, un flujo de cadenas de suministro no lineales que consideren bifurcaciones y uniones del producto inicialmente identificado. Por otro lado, también se puede trabajar en el manejo y almacenamiento de información con respecto a estándares sanitarios, certificaciones en el campo del café de especialidad, así como permitir que los contratos inteligentes automaticen las transacciones de pago de los servicios prestados y las compras llevadas a cabo, cada vez que la red de suministro involucre otro participante, esto con el fin de obtener la potencialidad completa de blockchain en cuanto a completar transacciones en línea sin ningún intermediario entre participantes que no necesariamente tienen un alto grado de confianza entre sí.

- Se recomienda iniciar un plan piloto para la implementación del presente diseño de sistema de trazabilidad y llevar a cabo un análisis económico sobre los costos de las transacciones efectuadas para interactuar con los contratos inteligentes para definir estrategias de adopción del sistema con el fin de impulsar a los pequeños productores y empresas que intervienen en la cadena de suministro del café de especialidad, con la visión de que la transparencia en la trazabilidad de sus productos les permitirá empoderarse económicamente.
- Finalmente, también se recomienda desarrollar una aplicación móvil, además de la aplicación web realizada, la cual facilite a los usuarios una interacción más rápida e inmediata con el sistema de trazabilidad.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] N. Petit, "Ethiopia's Coffee Sector: A Bitter or Better Future?," *Journal of Agrarian Change*, vol. 7, no. 2, pp. 225–263, 2007, doi: 10.1111/J.1471-0366.2007.00145.X.

[2] Rikolto Non-governmental Organization, "Café ecuatoriano, aromatizando la economía nacional," *Rikolto en Latinoamérica*, 2020. https://latinoamerica.rikolto.org/es/project/cafe-ecuatoriano-aromatizando-la-economia-nacional (accessed Jul. 03, 2022).

[3] N. Kshetri, "1 Blockchain's roles in meeting key supply chain management objectives," *Int J Inf Manage*, vol. 39, pp. 80–89, 2017, doi: 10.1016/j.ijinfomgt.2017.12.005.

[4] D. Lu, "Relationship and Integration," in *Fundamentals of Supply Chain Management*, 1st ed., Frederikesberg, Denmark: Ventus Publishing ApS, 2011, pp. 103–104.

[5] S. Chain and L. Myllymaa, "Impact of blockchain on sustainable supply chain practices: A study on blockchain technology's benefits and current barriers in sustainable SCM," Jönköping University, 2021. Accessed: Jun. 19, 2022. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1574301/FULLTEXT01.pdf

[6] P. Palamara and G. Miragliotta, "TRACING AND TRACKING WITH THE BLOCKCHAIN," Politecnico di Milano, 2017. Accessed: Jun. 17, 2022. [Online]. Available: https://www.politesi.polimi.it/bitstream/10589/139387/3/2018_04_Palamara.pdf

[7] J. G. Kiano, "A Mobile application to improve tracking and verification of products in supply chain logistics using blockchain technology," Strathmore University, 2018. Accessed: Jun. 19, 2022. [Online]. Available: https://su-plus.strathmore.edu/handle/11071/5957

[8] Supply Chain North Carolina State University, "Trust in Supply Chain Relationships: What Does It Mean to Trust? - Part I," *Supply Chain Resource Cooperative*, 2003. https://scm.ncsu.edu/scm-articles/article/a-sustainable-energy-platform-for-the-eu-and-the-world-2 (accessed Jun. 27, 2022).

[9] S. Islam, L. Manning, and J. M. Cullen, “Advances in traceability systems in agri-food supply chains,” in *Developing smart agri-food supply chains : using technology to improve safety and quality*, 1st ed., London: Burleigh Dodds Science Publishing, 2021, pp. 3–28. doi: 10.19103/AS.2021.0097.01.

[10] P. Olsen and M. Borit, “How to define traceability,” *Trends Food Sci Technol*, vol. 29, no. 2, pp. 142–150, 2013, doi: 10.1016/J.TIFS.2012.10.003.

[11] T. Pizzuti and G. Mirabelli, “The Global Track&Trace System for food: General framework and functioning principles,” *J Food Eng*, vol. 159, pp. 16–35, 2015, doi: 10.1016/J.JFOODENG.2015.03.001.

[12] T. Bosona and G. Gebresenbet, “Food traceability as an integral part of logistics management in food and agricultural supply chain,” *Food Control*, vol. 33, no. 1, pp. 32–48, 2013, doi: 10.1016/J.FOODCONT.2013.02.004.

[13] T. Moe, “Perspectives on traceability in food manufacture,” *Trends Food Sci Technol*, vol. 9, no. 5, pp. 211–214, 1998, doi: 10.1016/S0924-2244(98)00037-5.

[14] H. M. Kim, M. S. Fox, and M. Grüninger, “An Ontology for Quality Management — Enabling Quality Problem Identification and Tracing,” *BT Technology Journal*, vol. 17, no. 4, pp. 131–140, 1999, doi: 10.1023/A:1009611528866.

[15] P. Olsen, “Food traceability in theory and in practice,” The Arctic University of Norway, 2018. Accessed: Jun. 28, 2022. [Online]. Available: https://hdl.handle.net/10037/15408

[16] M. M. Aung and Y. S. Chang, “Traceability in a food supply chain: Safety and quality perspectives,” *Food Control*, vol. 39, no. 1, pp. 172–184, May 2014, doi: 10.1016/J.FOODCONT.2013.11.007.

[17] A. Jeppsson and O. Oskar, “Blockchains as a solution for traceability and transparency,” Lund University, 2017. Accessed: Jun. 28, 2022. [Online]. Available: https://lup.lub.lu.se/student-papers/search/publication/8919957

[18] G. R.T. White, G. Gardiner, G. Prabhakar, and A. Abd Razak, “A Comparison of Barcoding and RFID Technologies in Practice,” *Journal of Information, Information Technology, and Organizations (Years 1-3)*, vol. 2, pp. 119–132, 2007, doi: 10.28945/142.

[19] G. Azuara, J. L. Tornos, and J. L. Salazar, “Improving RFID traceability systems with verifiable quality,” *Industrial Management & Data Systems*, vol. 112, no. 3, pp. 340–359, 2012, doi: 10.1108/02635571211210022/FULL/XML.

[20] A. S. Gonnagar, “Product traceability in manufacturing industries: Business case and pilot project.,” Instituto Politécnico de Coimbra, 2018.

[21] P. M. Lourenço Costa and F. Figueiredo Correia, “Supply Chain Management with Blockchain Technologies,” University of Porto, 2018. Accessed: Jun. 30, 2022. [Online]. Available: https://www.linkedin.com/in/pedro-costa-3279996b

[22] C. Brennan and W. Lunn, “Blockchain: The Trust Disrupter,” London, UK, 2016. Accessed: Jul. 01, 2022. [Online]. Available: https://www.finextra.com/finextra-downloads/newsdocs/document-1063851711.pdf

[23] S. Ray, “Cryptographic Hashing,” *Hackernoon*, 2017. https://hackernoon.com/cryptographic-hashing-c25da23609c3 (accessed Jul. 01, 2022).

[24] F. Tschorsch and B. Scheuermann, “Bitcoin and beyond: A technical survey on decentralized digital currencies,” *IEEE Communications Surveys and Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016, doi: 10.1109/COMST.2016.2535718.

[25] S. Agarwal, “Blockchain technology in Supply Chain and Logistics,” Massachusetts Institute of Technology, 2018. Accessed: Jul. 02, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Blockchain-technology-in-Supply-Chain-and-Logistics-Agarwal/4a42733ee1c9e8416346331753da9339f75d69fd

[26] B. Vitaris, “Public vs. Private Blockchain: What’s the Difference?,” *Permission Blog*, 2021. https://permission.io/blog/public-vs-private-blockchain/ (accessed Jul. 25, 2022).

[27] Ethereum Organization, “Ethereum development documentation | ethereum.org,” *Ethereum Organization Site*, 2022. https://ethereum.org/en/developers/docs/ (accessed Aug. 29, 2022).

[28] Solidity Language Development Community, “Solidity — Solidity 0.8.16 documentation,” *Solidity Documentation*, 2022. https://docs.soliditylang.org/en/v0.8.16/ (accessed Aug. 29, 2022).

[29] Blockchain Engineer Resource Community, “Keccak256 hash function in Solidity,” *Crypto Market Pool Site*, 2022. https://cryptomarketpool.com/keccak256/ (accessed Aug. 04, 2022).

[30] Exodus Movement Incorporation, “What is a custom RPC? How do I add a network to MetaMask?,” *Exodus Support Site*, 2022. https://support.exodus.com/article/1791-what-is-a-remote-procedure-call-rpc (accessed Aug. 06, 2022).

[31] V. Garg, “Ethereum Blockchain - Understand Ganache Blockchain User Interface,” *Oracle Fusion Middleware Blog*, 2022. https://www.soawork.com/2020/06/ethereum-blockchain-understand-ganache-blockchain-ui-options.html (accessed Aug. 06, 2022).

[32] Ethers.js Development Community, “Getting Started,” *Ethers.js Documentation*, 2022. https://docs.ethers.io/v5/getting-started/#getting-started--connecting (accessed Aug. 08, 2022).

[33] IPFS Development Community, “What is IPFS? | IPFS Docs,” *IPFS Documentation*, 2019. https://docs.ipfs.tech/concepts/what-is-ipfs/#decentralization (accessed Aug. 08, 2022).

[34] Vercel Incorporation, “Introduction to Vercel,” *Vercel Documentation*, 2022. https://vercel.com/docs (accessed Aug. 08, 2022).

[35] R. Ginsburg, “What are Gas Fees and How Can We Fix Them?,” *NFT Now Guides*, 2022. https://nftnow.com/guides/what-are-gas-fees-and-how-can-we-fix-them/ (accessed Jul. 31, 2022).

5 ANEXOS

ANEXO I. Diagrama del proceso de validación en blockchain y estructura del árbol Merkle

ANEXO II. Definición de los parámetros de la fórmula (2.1) empleada para calcular la tarifa de gas que debe pagar un usuario al realizar transacciones en la blockchain de Ethereum

ANEXO III. Soluciones de escalado de la cadena de bloques de Ethereum para reducir el costo de la tarifa de transacción al usar la red

ANEXO IV. Procedimiento para agregar la testnet de Rinkeby en la billetera digital de MetaMask

ANEXO V. Procedimiento para agregar la red de Polygon en la billetera digital de MetaMask

ANEXO VI. Capturas de pantalla correspondientes a la configuración del proyecto con Truffle

ANEXO VII. Diagrama de estados del flujo de funcionamiento del sistema de tracking implementado en los contratos inteligentes

ANEXO VIII. Diagrama de modelo del sistema que especifica cada una de las funciones implementadas en los contratos inteligentes

ANEXO IX. Diagrama de secuencia que representa la interacción entre los usuarios con cada contrato desarrollado e interacción entre contratos, enfocado en las funciones de escritura y en los eventos emitidos en la blockchain para las etapas desde el registro de la granja hasta la venta de café

ANEXO X. Diagrama de secuencia que representa la interacción entre los usuarios con cada contrato desarrollado e interacción entre contratos, enfocado en las funciones de escritura y en los eventos emitidos en la blockchain para las etapas desde bodegaje hasta la comercialización en retailer

ANEXO XI. Diagrama de secuencia que representa la interacción entre los usuarios con cada contrato desarrollado e interacción entre contratos, enfocado en las funciones de lectura para todas las etapas de la cadena de suministro de café

ANEXO XII. Diagrama de entidad relación del sistema que especifica las variables con su tipo de dato, las funciones, los atributos y las llamadas de función que presentan las diferentes entidades

ANEXO XIII. Diagrama de flujo de la función *añadirDatosGranja* que lleva a cabo la creación del identificador único para la unidad trazable del lote de café

ANEXO XIV. Diagrama de flujo del proceso de ingreso de usuarios en el sistema, realizado por el administrador

ANEXO XV. Diagrama de flujo de la función *añadirDatosCosecha* que realiza el proceso de ingreso de información con respecto a la cosecha del lote de café

ANEXO XVI. Diagrama de flujo de la función *actualizarUsuario* que lleva a cabo la actualización de información de perfil por parte de los diferentes usuarios

ANEXO XVII. Diagrama de flujo de la función *obtenerDatosProcesado* que lleva a cabo la lectura de información con respecto a la etapa de procesado del lote de café

ANEXO XVIII. Diagrama de flujo de la función *obtenerUsuario* que lleva a cabo la lectura de información de perfil de los usuarios

ANEXO XIX. Capturas de pantalla correspondientes a los pasos para configurar la cadena de bloques local basada en Ethereum con Ganache

ANEXO XX. Definición de los parámetros empleados en la configuración de las redes del ecosistema de Ethereum en el script truffle-config.js

ANEXO XXI. Pasos para crear un proyecto en Infura y obtener un nodo de conexión con una blockchain del ecosistema de Ethereum

ANEXO XXII. Pasos para añadir el complemento de verificación y generar una clave API en Etherscan empleada en la verificación de los contratos inteligentes desplegados en la blockchain

ANEXO XXIII. Script de configuración de Truffle (truffle-config.js) en donde se definen las redes de conexión (local con Ganache, testnet con Rinkeby y real con Polygon), la versión de compilación de los contratos y el complemento de verificación

ANEXO XXIV. Archivo 2_deploy_contracts.js correspondiente a la migración de los contratos inteligentes del sistema de tracking

ANEXO XXV. Especificación de los casos de pruebas codificados para los contratos inteligentes

ANEXO XXVI. Registros de consola de todos los casos de prueba verificados al ejecutar los tests

ANEXO XXVII. Registros de consola de los contratos desplegados en la red local con Ganache y en la red testnet Rinkeby

ANEXO XXVIII. Capturas de pantalla de la ejecución del comando de verificación de contratos en la terminal y la consulta de los contratos verificados en el explorador Etherscan

ANEXO XIX. Diagrama de interacción entre la cadena de bloques Rinkeby y el navegador web por medio de Ethers.js

ANEXO XXX. Script del código para la escucha de eventos nuevos en la blockchain con el método *.on()*

ANEXO XXXI. Script del código para el filtrado de eventos ya emitidos en la blockchain con el método *.queryFilter()*

ANEXO XXXII. Configuración de un nodo de IPFS con Infura

ANEXO XXXIII. Script del código de la conexión con el servicio de IPFS para almacenar archivos tipo imagen

ANEXO XXXIV. Script del código para el uso de las librerías *qrcode.react* para generar un código QR y *react-qr-reader* para leer un código QR

ANEXO XXXV. Pasos para crear un clave API en el servicio de Google Maps API

ANEXO XXXVI. Script del código para el uso de las librerías *@react-google-maps/api* para generar y usar un mapa de Google Maps API y *use-places-autocomplet*e para autocompletar direcciones en una barra de búsqueda

ANEXO XXXVII. Diagrama de la organización visual de la interfaz web desarrollada

ANEXO XXXVIII. Diagramas de flujo de los procesos de uso de la aplicación web

ANEXO XXXIX. Proceso para alojar el proyecto en el servicio de Vercel

ANEXO XL. Manual de usuario: instalación del sistema y uso de la aplicación web para interactuar con el sistema de tracking

ANEXO XLI. Enlaces del video del funcionamiento del sistema de tracking y de los entregables del proyecto

ANEXO I

Diagrama del proceso de validación en blockchain y estructura del árbol Merkle.

Como se muestra en la Figura 5.1, el nonce es la parte móvil del hash del encabezado del bloque, cuyo valor se puede flexionar para resolver la prueba de trabajo (PoW). Los mineros procesan el bloque anterior y la raíz de Merkle con una función de hashing, en este caso la función SHA-256, hasta que encuentran un nonce que prefija el encabezado con suficientes ceros para satisfacer la condición del PoW. La dificultad o la cantidad de ceros que debe preceder el hash se modifica automáticamente con el fin de garantizar que independientemente de la potencia de procesamiento de los nodos, todos los nodos de la red blockchain, puedan tener la misma oportunidad de participación en la validación del bloque actual [22]. En este caso, se observa que el minero AntPool ubicado en Beijing ha resuelto la prueba de trabajo del bloque 415,350 con el nonce 209,636,388. El registro del bloque en la blockchain se realiza con la estampa de tiempo correspondiente a las 12:59:07 horas de la fecha 2016-06-08.

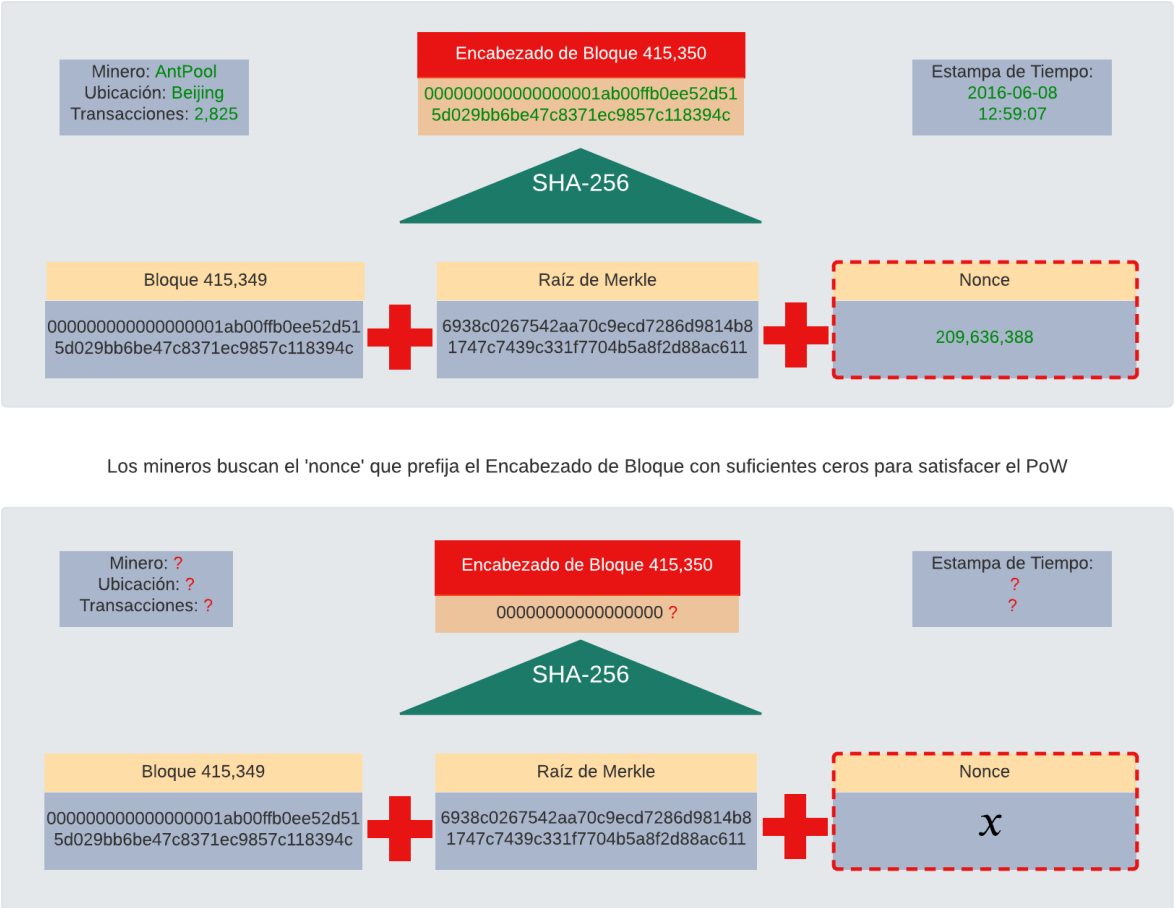


Figura 5.1. Bloque 415,350 extraído con éxito por un minero de AntPool en Beijing agregando 209,636,388 a la raíz de Merkle y al hash del bloque anterior

En la Figura 5.2, se visualiza la representación ramificada que constituye el árbol Merkle. Cada bloque contiene la raíz de Merkle que contiene los hashes de las uniones de todas las transacciones aguas abajo del árbol Merkle. De esta manera, las cadenas de datos que comunican una transacción son procesadas con la función de hashing correspondiente, se emparejan, se concatenan con otras transacciones y se procesan nuevamente con la función de hashing hasta que se forma el hash raíz que se encuentra en la parte superior del árbol Merkle [22].

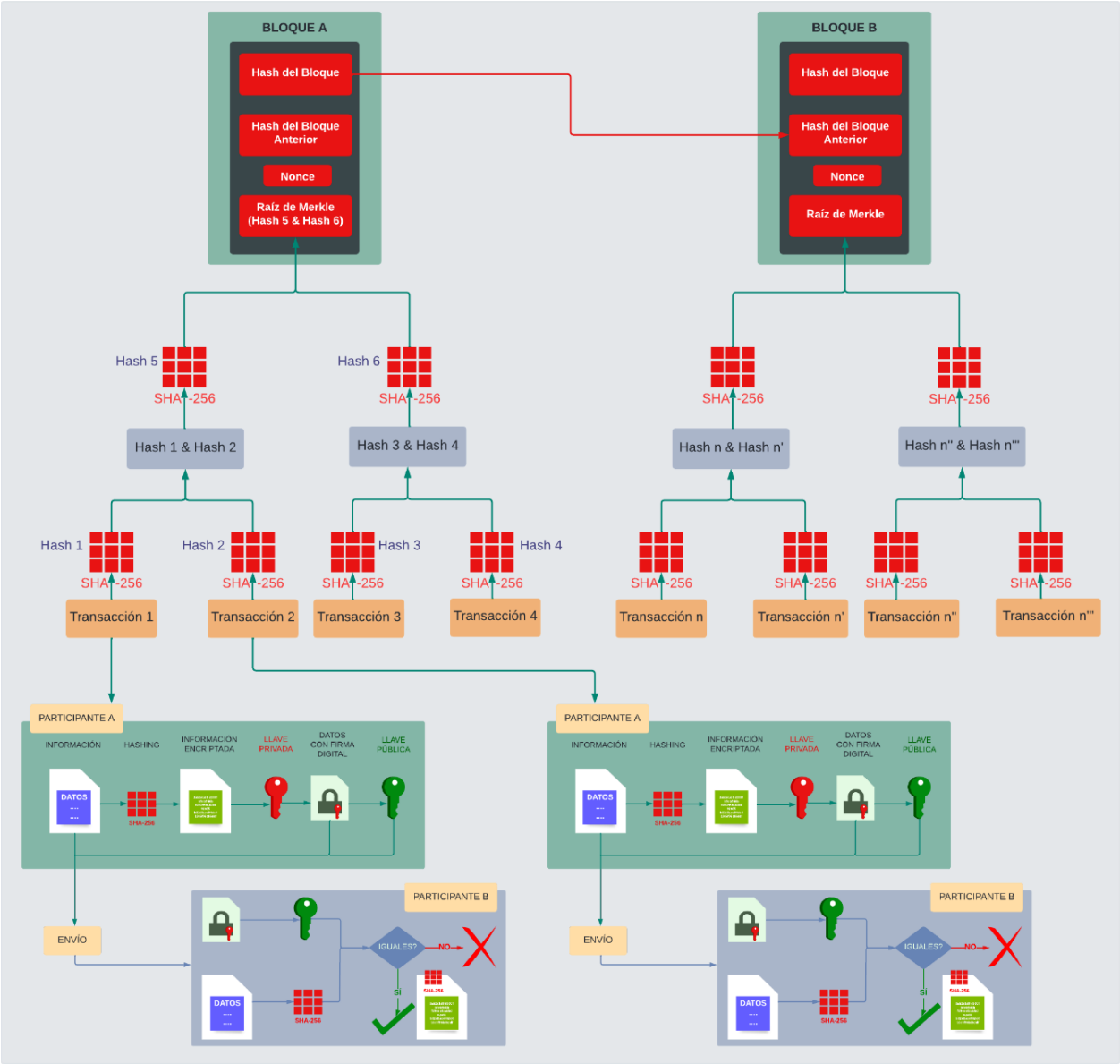


Figura 5.2. Estructura del árbol Merkle de las transacciones en blockchain

ANEXO II

Definición de los parámetros de la fórmula (2.1) empleada para calcular la tarifa de gas que debe pagar un usuario al realizar transacciones en la blockchain de Ethereum.

Tabla 5.1. Definición de los parámetros de la fórmula para calcular la tarifa de transacción en Ethereum

Fórmula	$T_{tx} = U_{gas}(\,límite) \times \left(T_{base} + P \right)$
Parámetro	**Definición**
T_{tx}	Tarifa de transacción. Su unidad es el $[Gwei]$, además $1\ Gwei = 10^{-9}\ ETH$
$U_{\text{gas}}(\,\text{límite})$	Unidades de gas, corresponde a la cantidad de gas empleada para llevar a cabo los cálculos matemáticos de la transacción [35]. El límite mínimo de gas es de 21.000 $[Gwei]$. Su unidad es el $[Gwei]$.
T_{base}	Tarifa base, es el mínimo precio por unidad de gas para incluir la transacción en un bloque. Es calculado en función de la demanda por espacio en el bloque a validar. El producto de esta tarifa por las unidades de gas se elimina de forma permanente de la red con cada transacción, esto como mecanismo para generar escasez de la criptomoneda en la red y aumentar su valoración. Su unidad es el $[Gwei]$.
P	Propina, el producto de este valor por las unidades de gas es la comisión pagada para que los nodos prioricen la validación de la transacción en el bloque actual. No es un valor obligatorio, sin embargo, la congestión en la red es un factor que lleva a incluir una propina para aumentar la velocidad de validación de una transacción [27]. Su unidad es el $[Gwei]$.

ANEXO III

Soluciones de escalado de la cadena de bloques de Ethereum para reducir el costo de la tarifa de transacción al usar la red.

Las premisas para llevar a cabo el escalado de la implementación de Ethereum son aumentar la velocidad de las transacciones y mejorar la ejecución y el rendimiento de estas, sin sacrificar la descentralización y la seguridad de la red. Todo esto permitiendo una adopción masiva de Ethereum.

Existen dos tipos de escalado, el escalado en cadena y fuera de la cadena. En la Tabla 5.2 se resumen ambas.

Tabla 5.2. Tipos de escalado de la blockchain de Ethereum

Tipo de Escalado	**Explicación** [27]
En cadena "on-chain"	- Requiere realizar cambios en el protocolo de Ethereum, es decir, en su capa 1.
Enfoques	
Enfoque	**Detalle**
Fragmentación (sharding)	- Es un método que divide horizontalmente la base de datos de blockchain para repartir la carga. - De esta manera, se crean nuevas cadenas denominadas fragmentos, las cuales validan y procesan las transacciones de manera separada, lo que conlleva a reducir la congestión de la red y aumentar la cantidad de transacciones por segundo llevadas a cabo.
Ethereum 2.0	- Se trata de la migración hacia el algoritmo de consenso por prueba de participación (PoS). - Esta migración, además de reducir el daño medioambiental, permite aumentar la velocidad de las transacciones al no gastar tiempo en la competición de los nodos.
Tipo de Escalado	**Explicación**
Fuera de la cadena "off-chain"	- Son soluciones que se implementan de manera separada a la red principal de Ethereum o capa 1, por lo que no contemplan un cambio en el protocolo.
Enfoques	
Enfoque	**Detalle**
Escalado de capa 2	- Implica que el procesamiento de las transacciones se lleva a cabo en una red separada que depende de la seguridad de la red principal. - Las soluciones de capa 2 hacen énfasis en mejorar la velocidad de la transacción y la escalabilidad, mientras que la descentralización y seguridad son temas de los cuales se ocupa la capa 1.
Rollups	- Se trata de un tipo de escalado de capa 2. - Los rollups se encargan de agrupar múltiples transacciones a la vez y ejecutarlas fuera de la blockchain de Ethereum, para luego publicarlas en la cadena en donde se lleva a cabo el algoritmo de consenso.

Plasma	- Es un tipo de escalado de capa 2. - Plasma es una cadena de bloques separada que se encuentra anclada a la cadena principal de Ethereum. - Utiliza pruebas de fraude para arbitrar disputas.
Validium	- Es un tipo de escalado de capa 2. - Utiliza pruebas de validez para verificar las transacciones fuera de la cadena principal.
Cadenas laterales	- Son cadenas de bloques independientes y compatibles con la EVM cuyo funcionamiento es paralelo a la red principal. - La compatibilidad se lleva a cabo a través de puentes bidireccionales, además implementan su propio algoritmo de consenso y criptomoneda.

ANEXO IV

Procedimiento para agregar la testnet de Rinkeby en la billetera digital de MetaMask.

1. Habilitar el botón para mostrar las redes de prueba a través de la Configuración Avanzada.

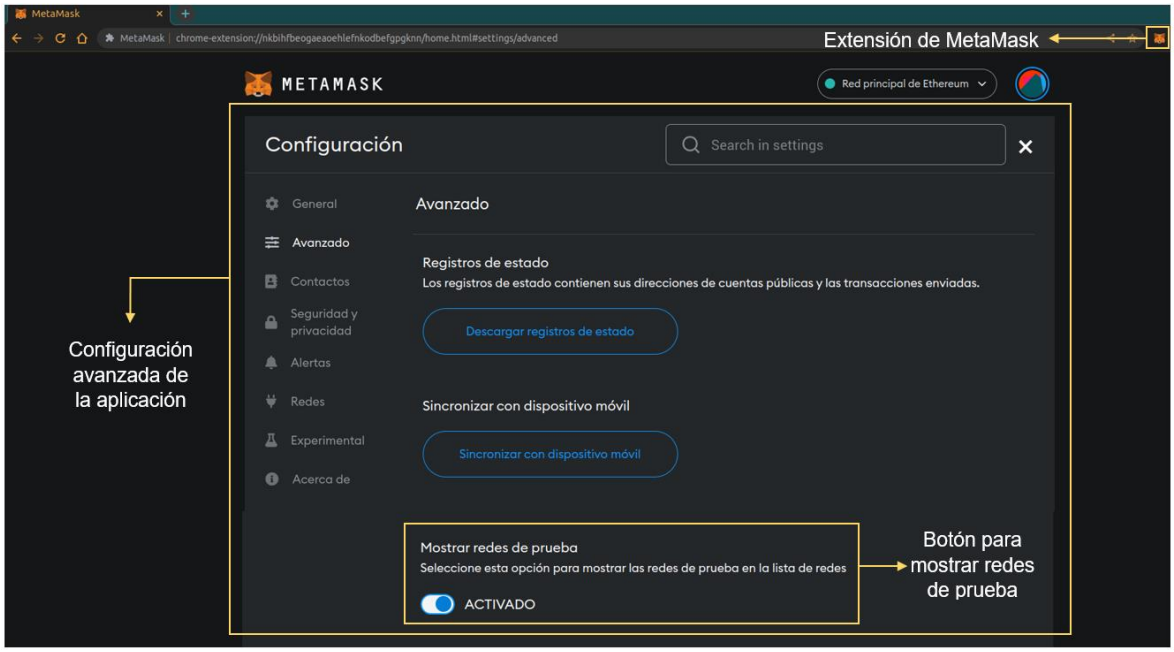


Figura 5.3. Paso 1: Activación de la opción para mostrar redes de prueba en MetaMask

2. Elegir la testnet Rinkeby del desplegable de redes disponibles.

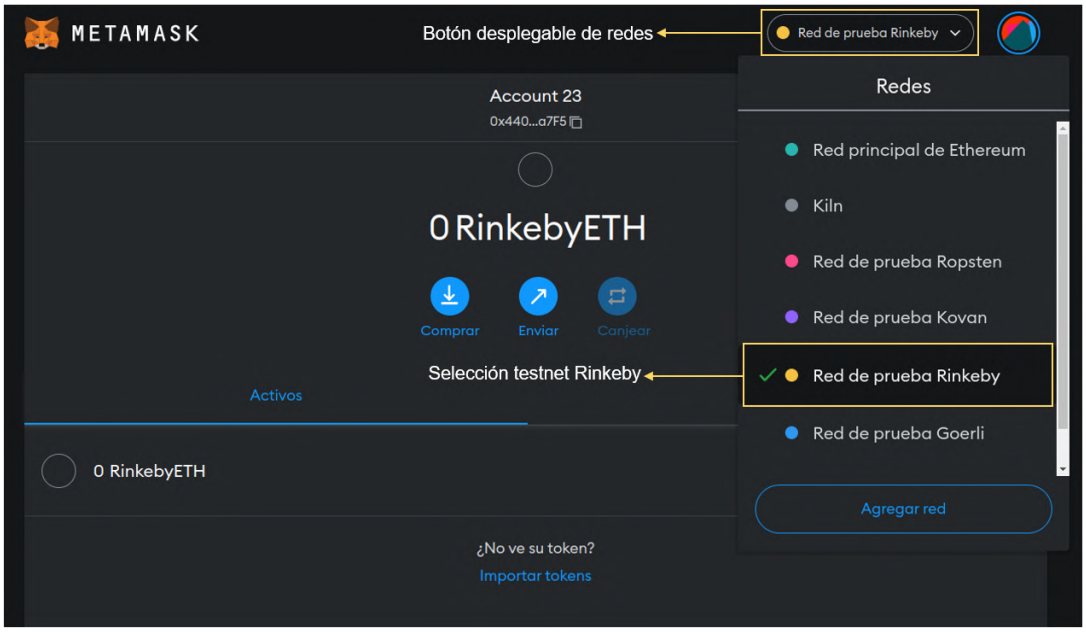


Figura 5.4. Paso 2: Selección de la testnet Rinkeby en MetaMask

3. Agregar ethers de prueba a la red por medio de un grifo de ethers de prueba. Se accede a la URL https://rinkebyfaucet.com/, se ingresa la dirección de la cuenta de la billetera MetaMask y se presiona el botón Send Me ETH. Nos podemos asegurar que la transacción se realizó al consultar la extensión de MetaMask y visualizar que la cuenta dispone de 0.1ETH en la red de Rinkeby.

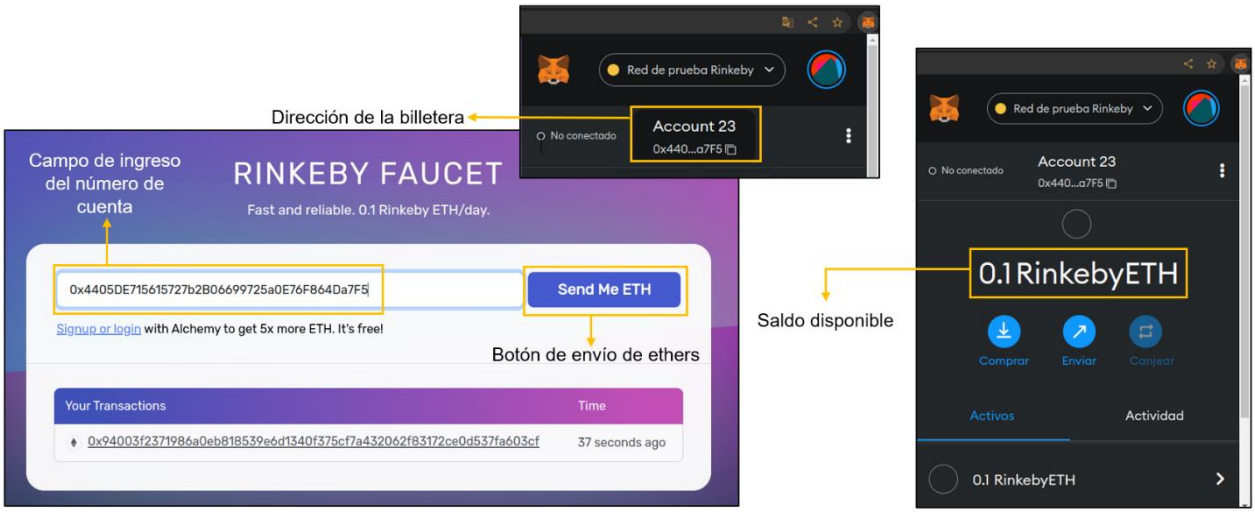


Figura 5.5. Paso 3: Obtención de ethers de prueba

ANEXO V

Procedimiento para agregar la red de Polygon en la billetera digital de MetaMask.

1. En la configuración de MetaMask seleccionar la opción de Redes y la opción de Agregar una red. Ingresar los datos de configuración de la red como sigue:

Tabla 5.3. Parámetros de configuración de la red Polygon en MetaMask

Nombre de la red	Polygon
Nueva dirección URL de RPC	https://polygon-rpc.com/
Identificador de cadena	137
Símbolo de moneda	Matic
Dirección URL del explorador de bloques	https://polygonscan.com/

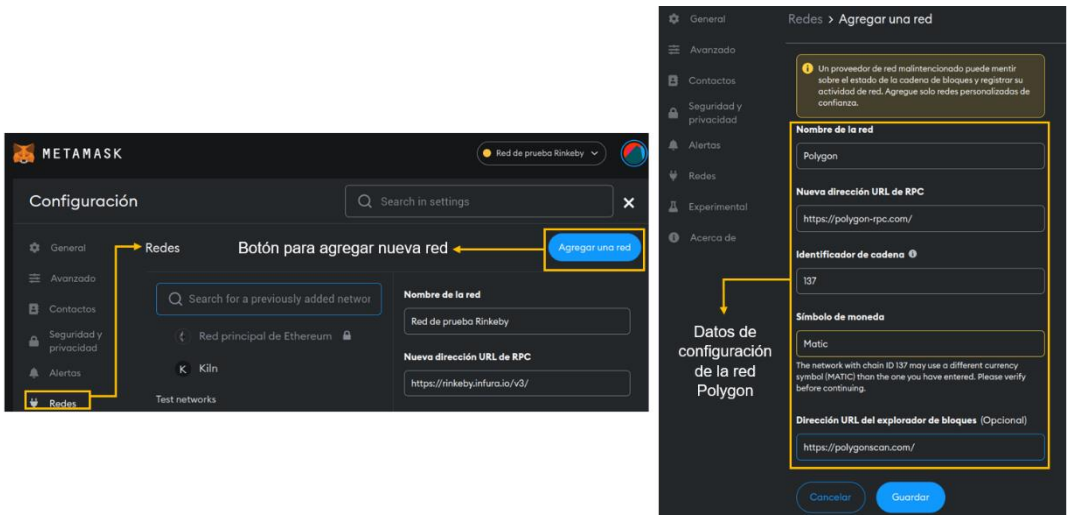


Figura 5.6. Paso 1: Adición y configuración de la red Polygon en MetaMask

2. Para usar la red de Polygon agregada debemos elegirla del desplegable de Redes disponibles.

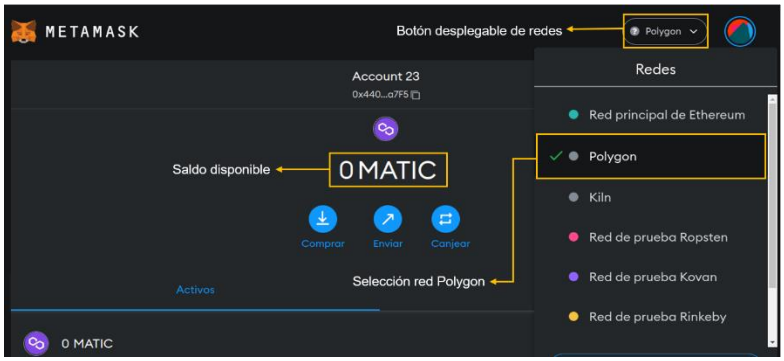


Figura 5.7. Paso 2: Selección de la red Polygon en MetaMask

La criptomoneda para esta red es el Matic. Como se observa en la Figura 5.7, la cuenta de la billetera MetaMask con la red Polygon, dispone de 0 Matic. Al ser esta una red real y requerir de criptomonedas con valor real, no se utilizará para el despliegue de los contratos inteligentes, sin embargo, se muestra la configuración para agregar la red en la billetera MetaMask, que se debería llevar a cabo en caso de ejecutar los contratos sobre esta blockchain del ecosistema de Ethereum.

ANEXO VI

Capturas de pantalla correspondientes a la configuración del entorno de desarrollo del proyecto con Truffle.

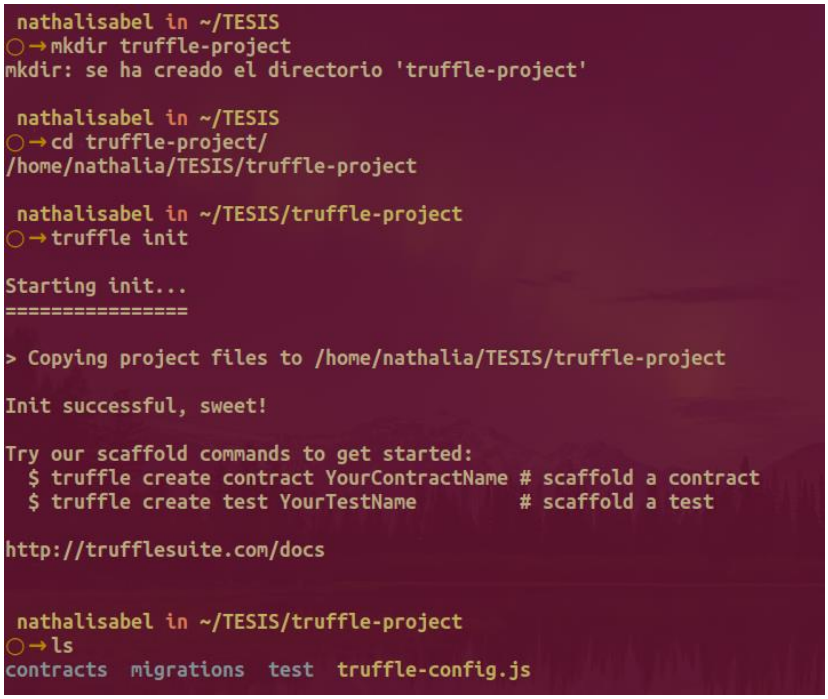


Figura 5.8. Comandos ejecutados en la terminal para la creación del proyecto con Truffle

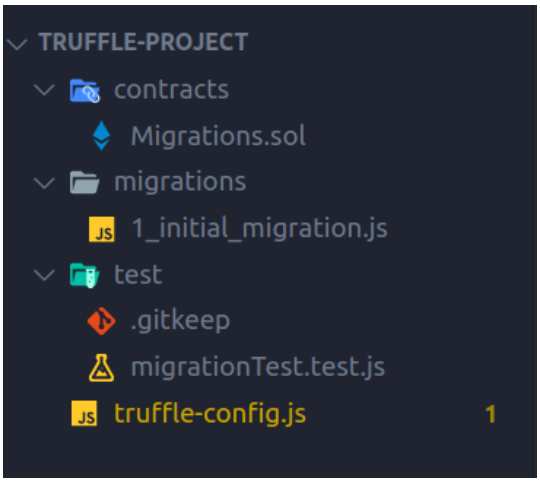


Figura 5.9. Árbol de archivos y carpetas del proyecto creado

ANEXO VII

Diagrama de estados del flujo de funcionamiento del sistema de tracking implementado en los contratos inteligentes.

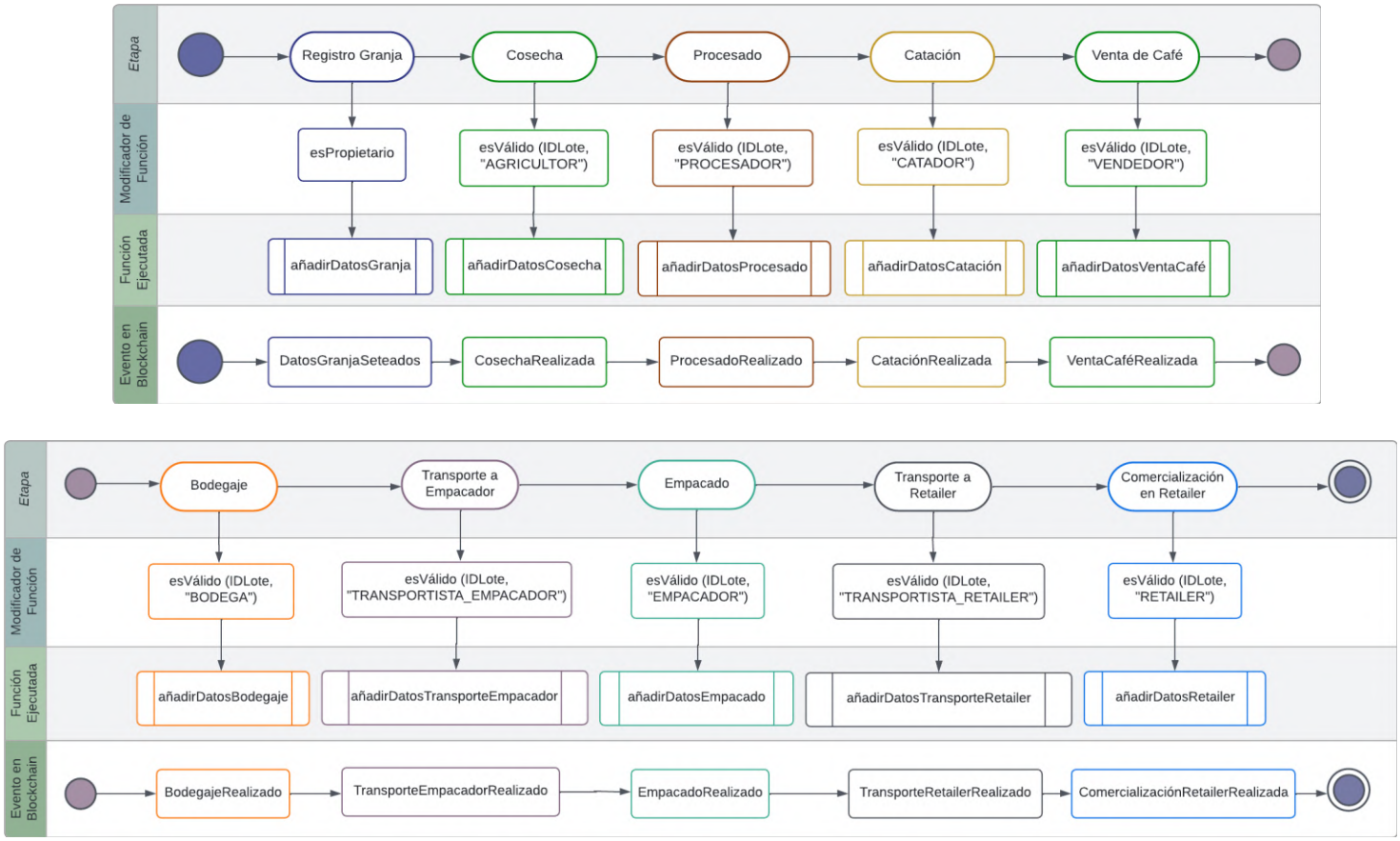


Figura 5.10. Diagrama de estados del sistema

ANEXO VIII

Diagrama de modelo del sistema que especifica cada una de las funciones implementadas en los contratos inteligentes.

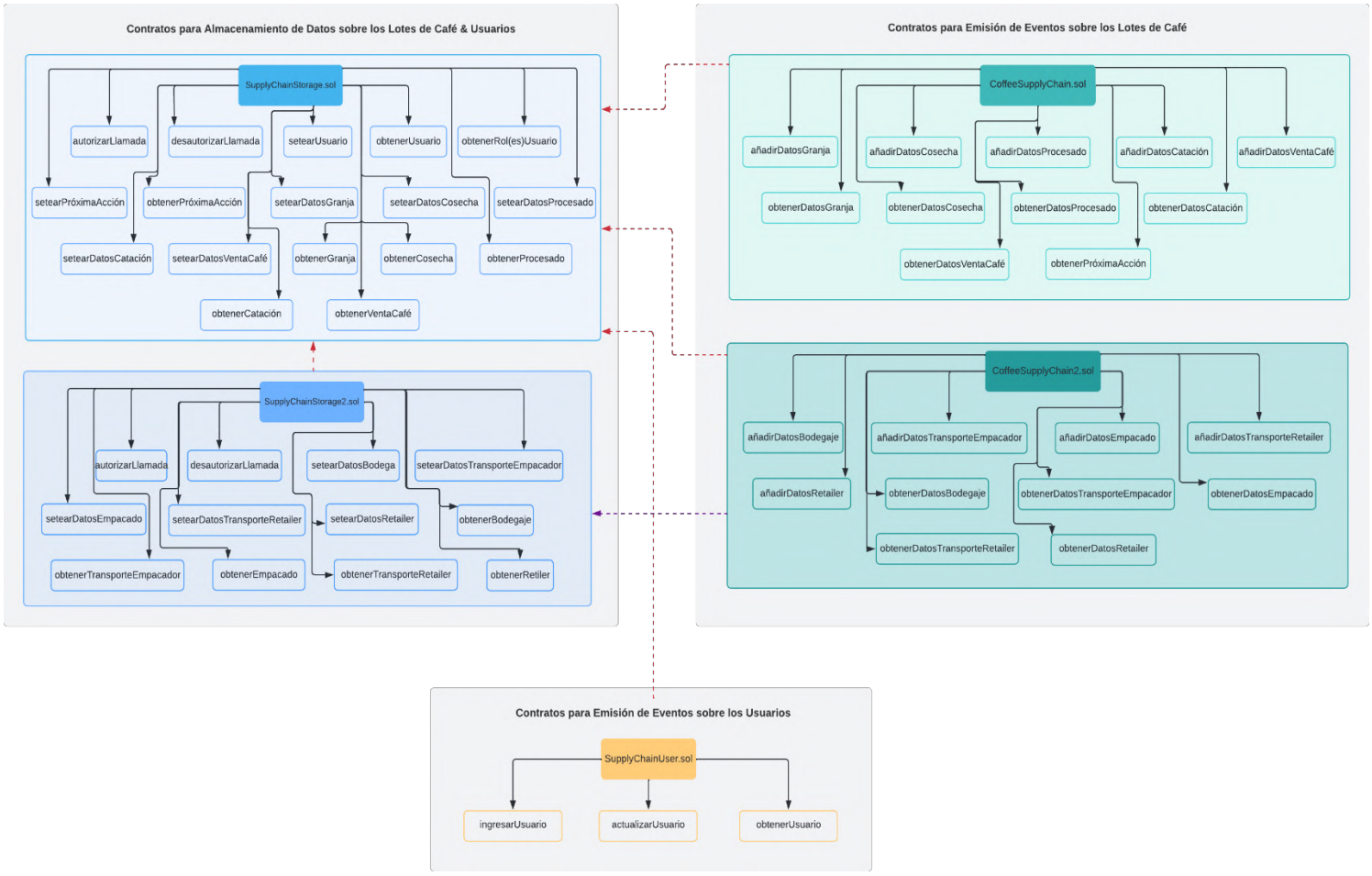


Figura 5.11. Diagrama de modelo de los contratos inteligentes desarrollados

ANEXO IX

Diagrama de secuencia que representa la interacción entre los usuarios con cada contrato desarrollado e interacción entre contratos, enfocado en las funciones de escritura y en los eventos emitidos en la blockchain para las etapas desde el registro de la granja hasta la venta de café.

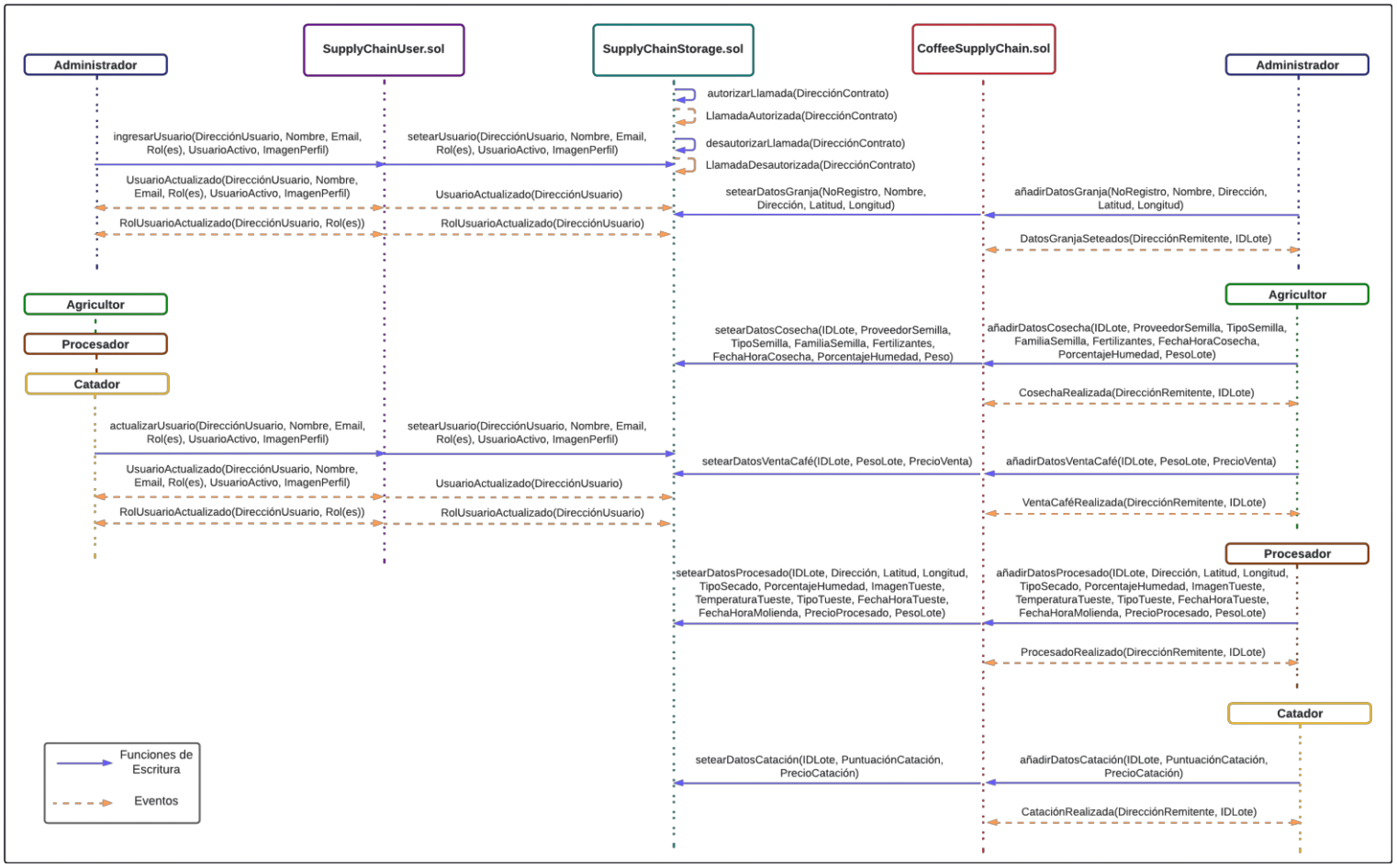


Figura 5.12. Diagrama de secuencia para las funciones de escritura de las etapas desde el registro de la granja hasta la venta de café

ANEXO X

Diagrama de secuencia que representa la interacción entre los usuarios con cada contrato desarrollado e interacción entre contratos, enfocado en las funciones de escritura y en los eventos emitidos en la blockchain para las etapas desde bodegaje hasta la comercialización en retailer.

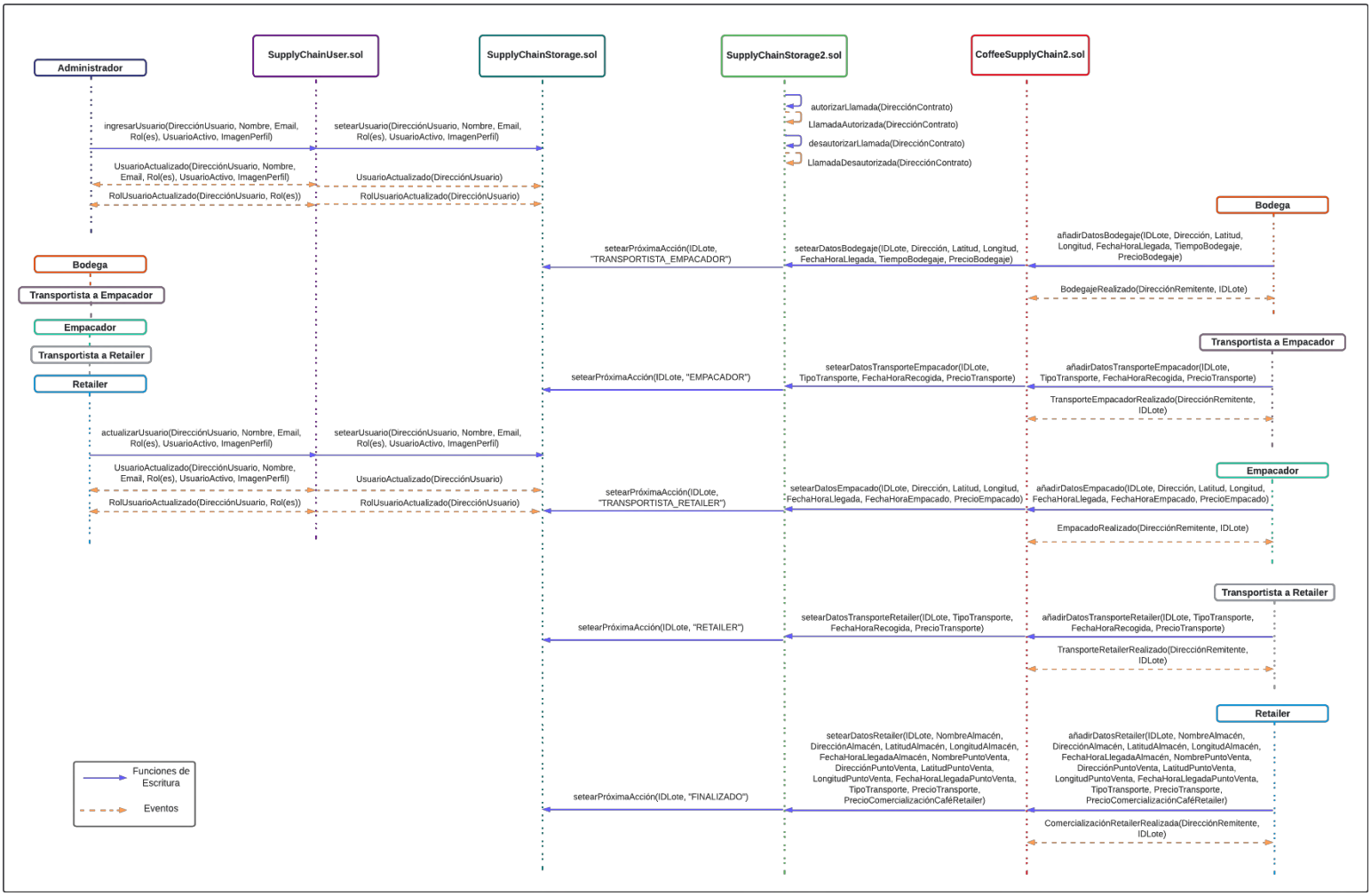


Figura 5.13. Diagrama de secuencia para las funciones de escritura de las etapas desde bodegaje hasta la comercialización en retailer

ANEXO XI

Diagrama de secuencia que representa la interacción entre los usuarios con cada contrato desarrollado e interacción entre contratos, enfocado en las funciones de lectura para todas las etapas de la cadena de suministro de café.

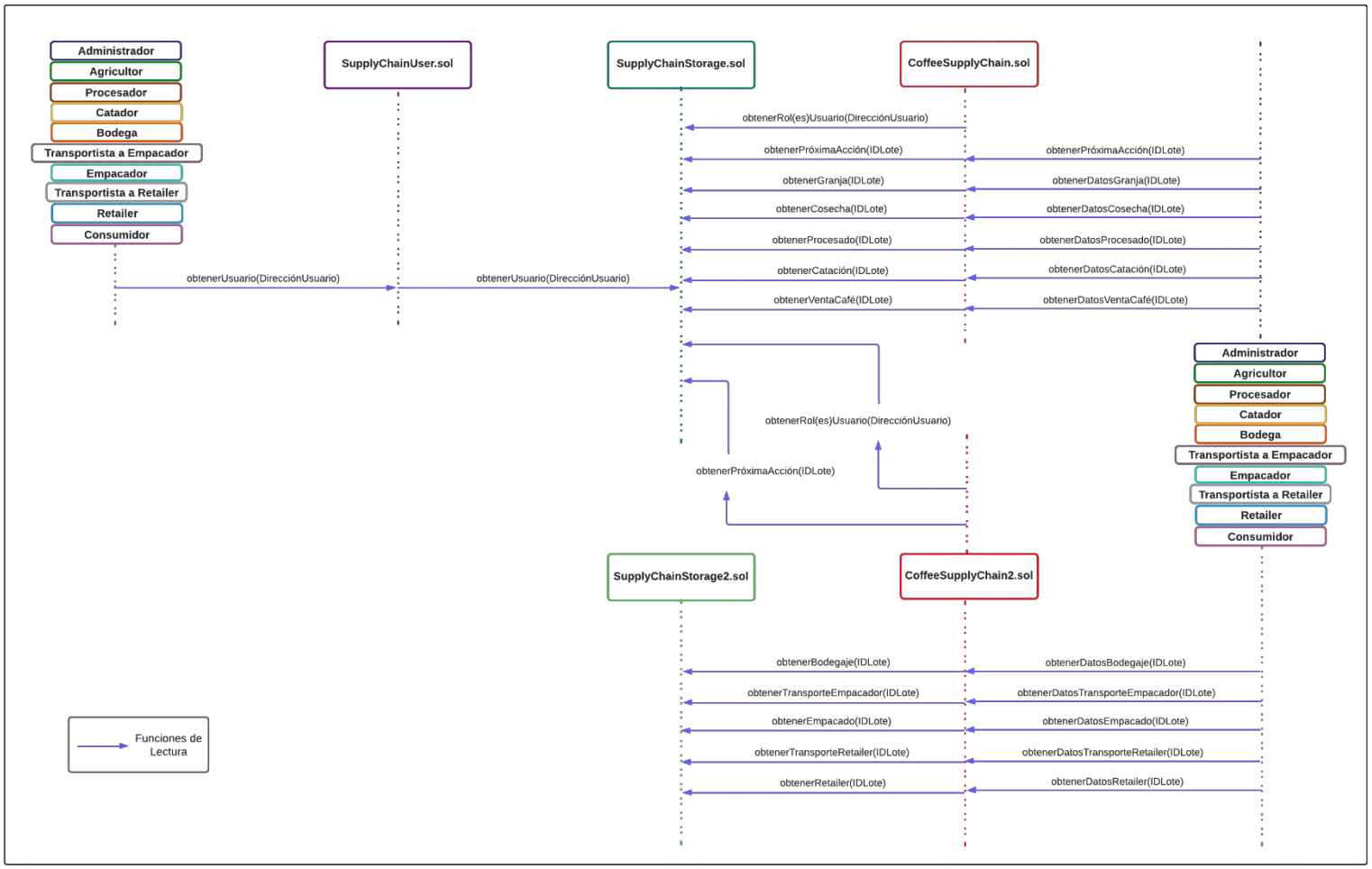


Figura 5.14. Diagrama de secuencia para las funciones de lectura

ANEXO XII

Diagrama de entidad relación del sistema que especifica las variables con su tipo de dato, las funciones, los atributos y las llamadas de función que presentan las diferentes entidades.

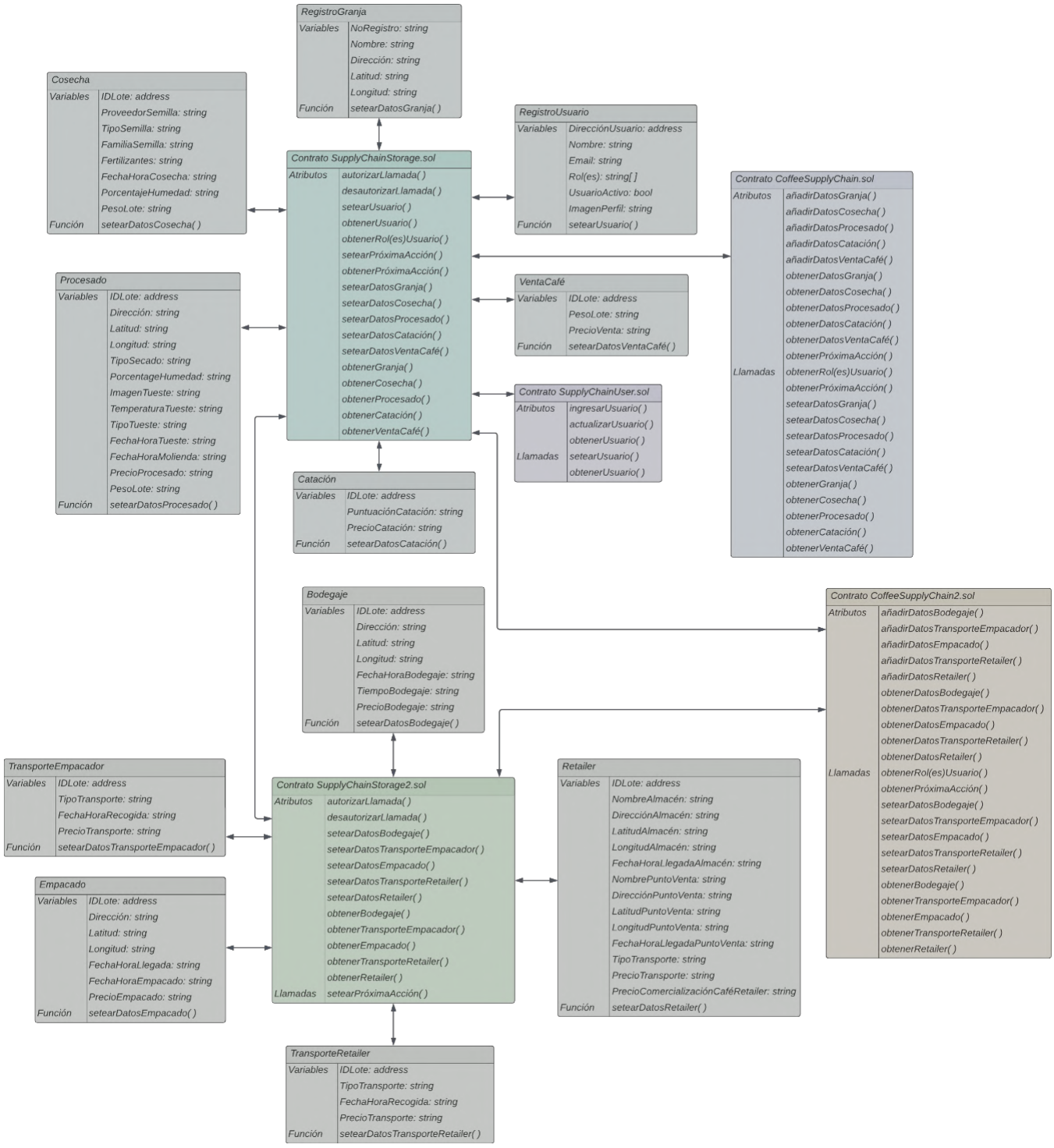


Figura 5.15. Diagrama de entidad relación del sistema

ANEXO XIII

Diagrama de flujo de la función *añadirDatosGranja* que lleva a cabo la creación del identificador único para la unidad trazable del lote de café.

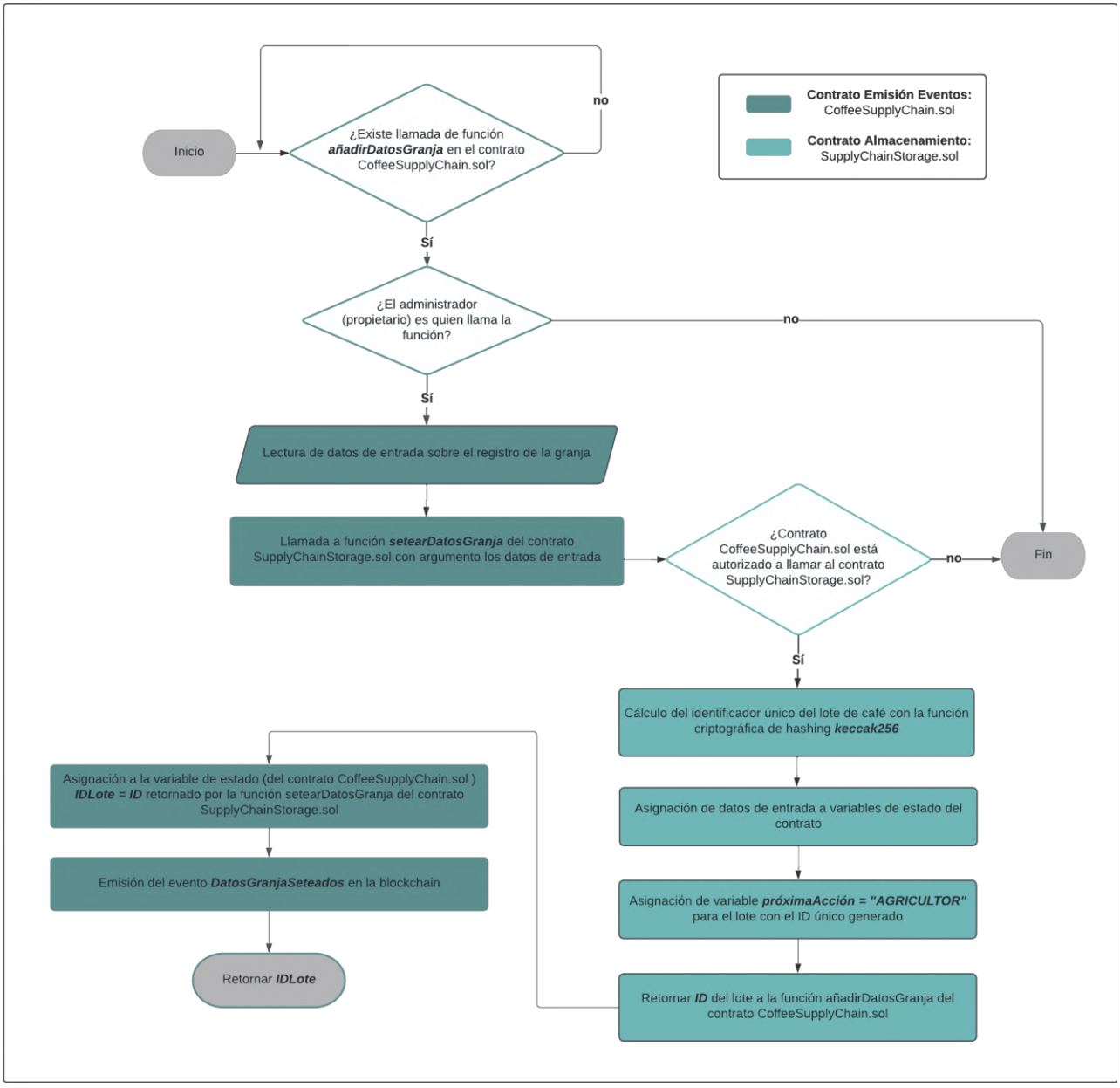


Figura 5.16. Diagrama de flujo de la función de generación del identificador único para el lote de café

ANEXO XIV

Diagrama de flujo del proceso de ingreso de usuarios en el sistema, realizado por el administrador.

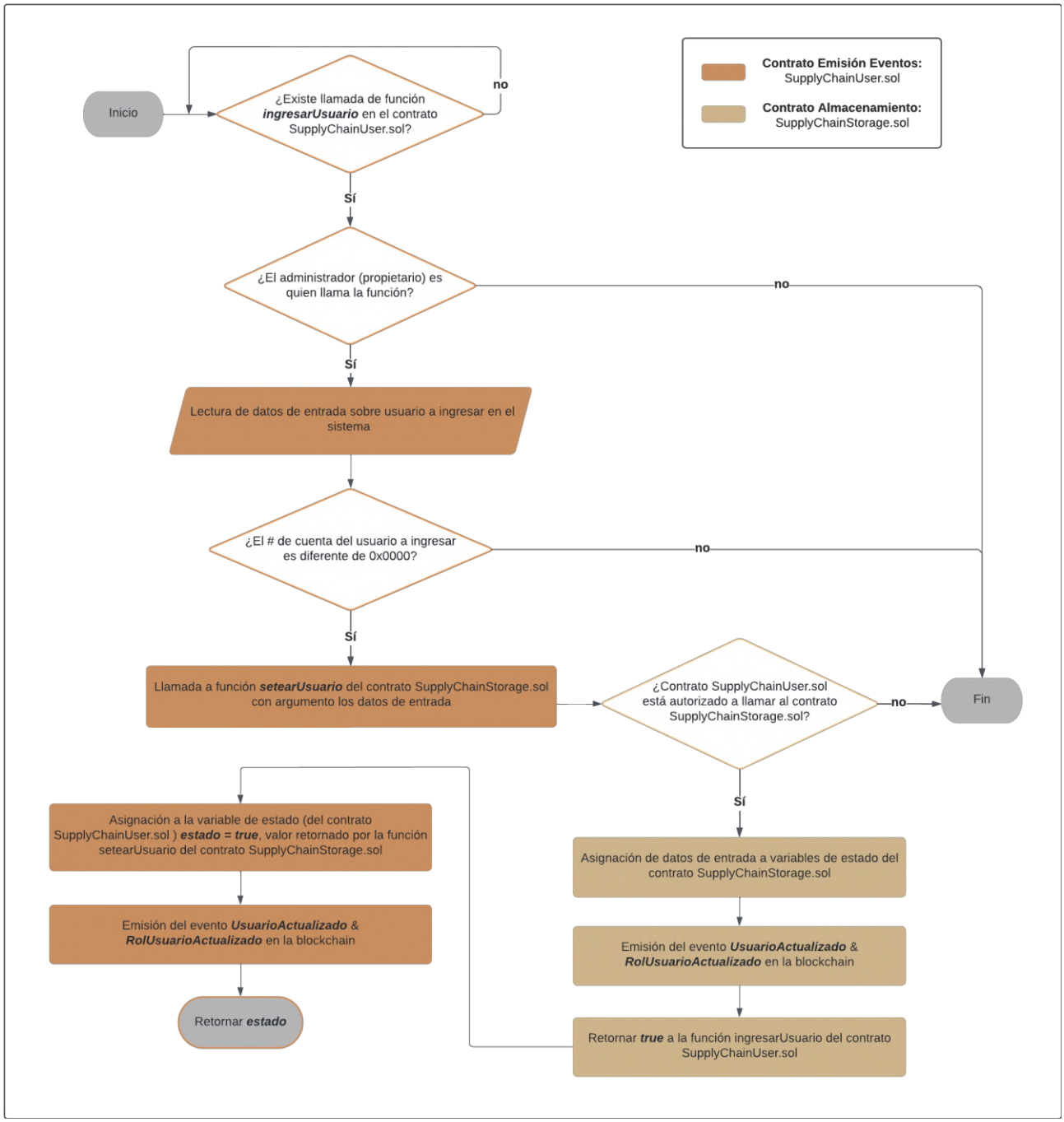


Figura 5.17. Diagrama de flujo de la función para el ingreso de usuarios en el sistema

ANEXO XV

Diagrama de flujo de la función *añadirDatosCosecha* que realiza el proceso de ingreso de información con respecto a la cosecha del lote de café.

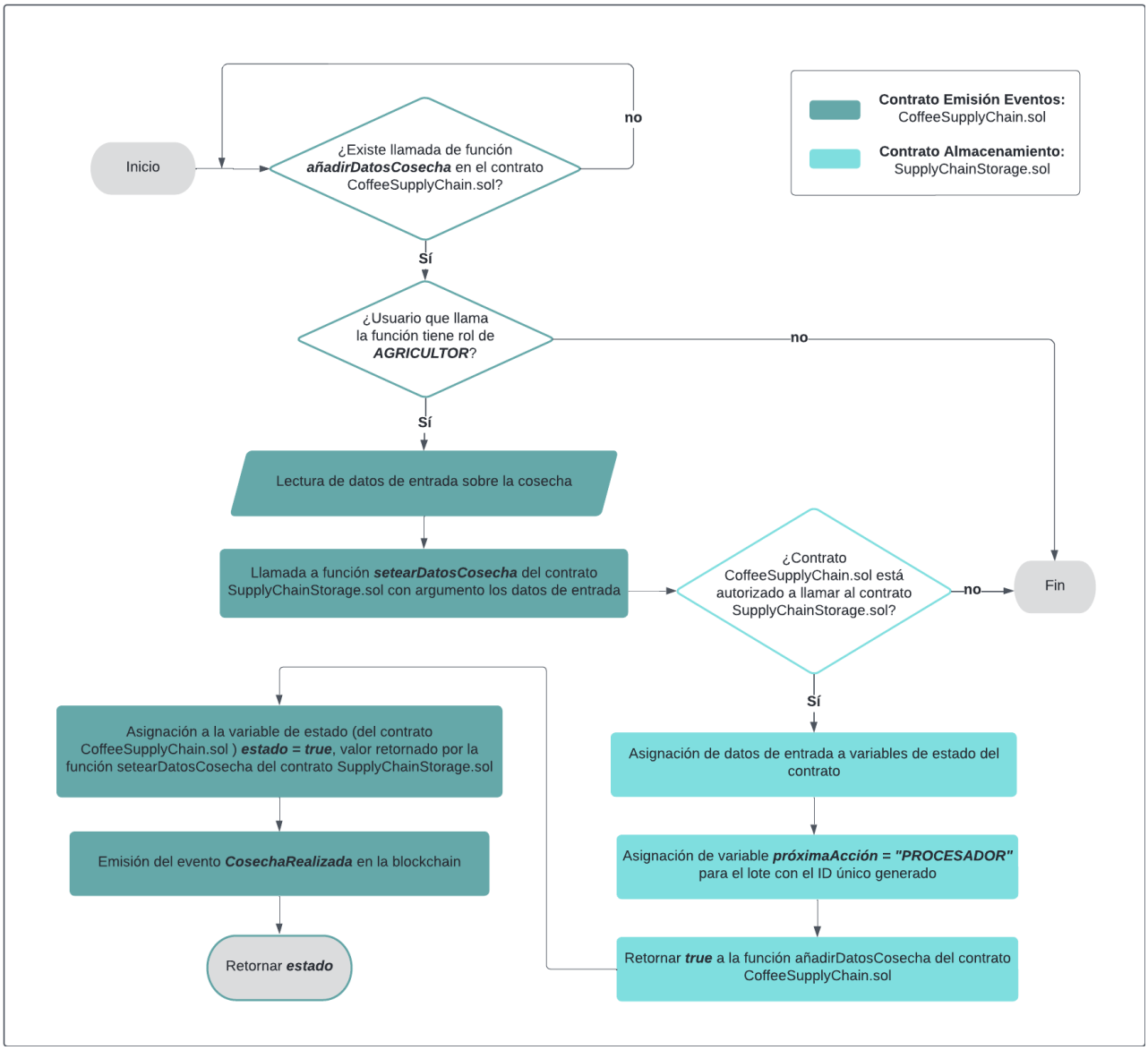


Figura 5.18. Diagrama de flujo de la función para el ingreso de información referente a la cosecha del lote de café

ANEXO XVI

Diagrama de flujo de la función *actualizarUsuario* que lleva a cabo la actualización de información de perfil por parte de los diferentes usuarios.

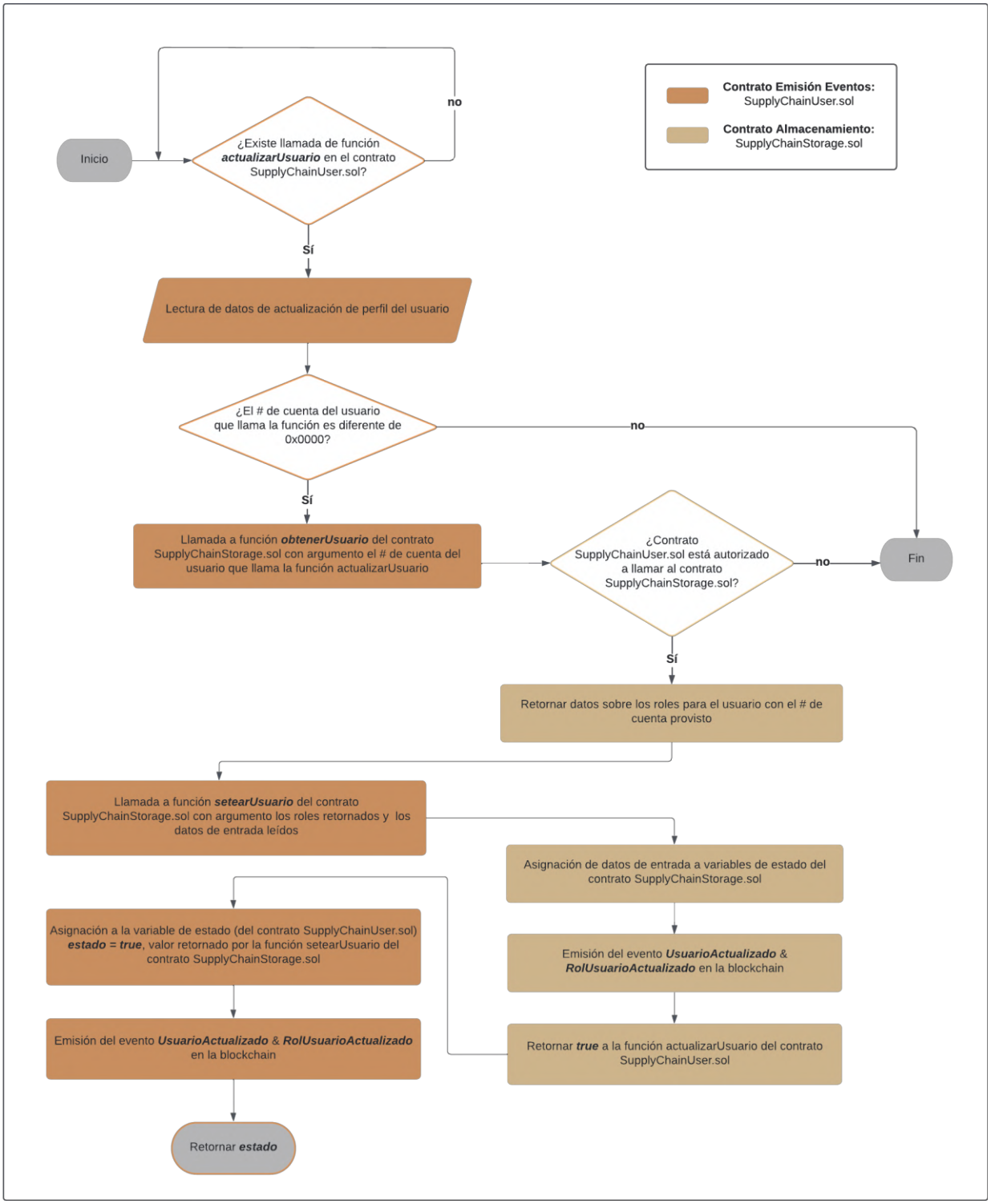


Figura 5.19. Diagrama de flujo de la función para modificación de los datos de perfil por parte de los usuarios

ANEXO XVII

Diagrama de flujo de la función *obtenerDatosProcesado* que lleva a cabo la lectura de información con respecto a la etapa de procesado del lote de café.

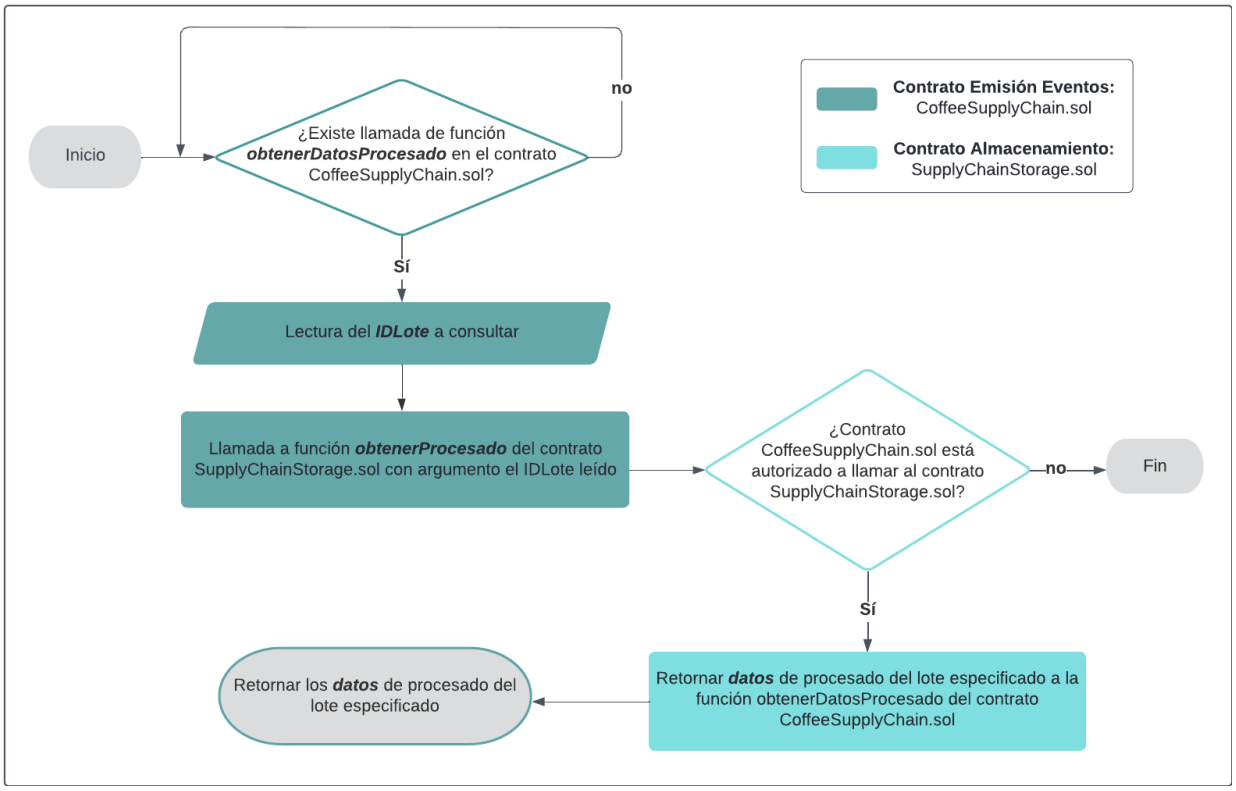


Figura 5.20. Diagrama de flujo de la función para leer los datos de procesado de un lote de café

ANEXO XVIII

Diagrama de flujo de la función *obtenerUsuario* que lleva a cabo la lectura de información de perfil de los usuarios.

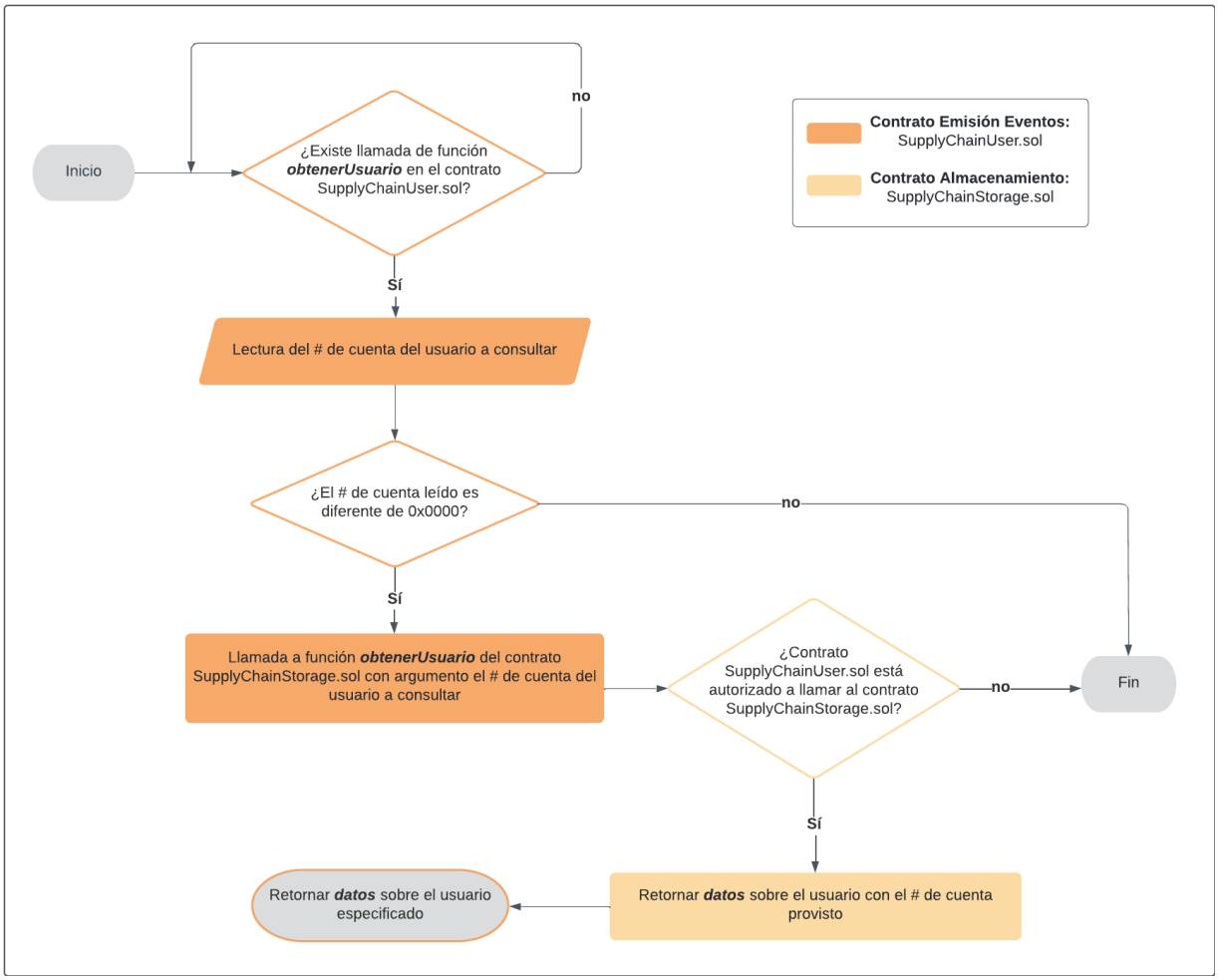


Figura 5.21. Diagrama de flujo de la función para leer los datos de perfil de los usuarios

ANEXO XIX

Capturas de pantalla correspondientes a los pasos para configurar la cadena de bloques local basada en Ethereum con Ganache.

Figura 5.22. Paso 1. Selección del botón New Workspace Ethereum para crear un nuevo entorno de cadena de bloques con Ganache

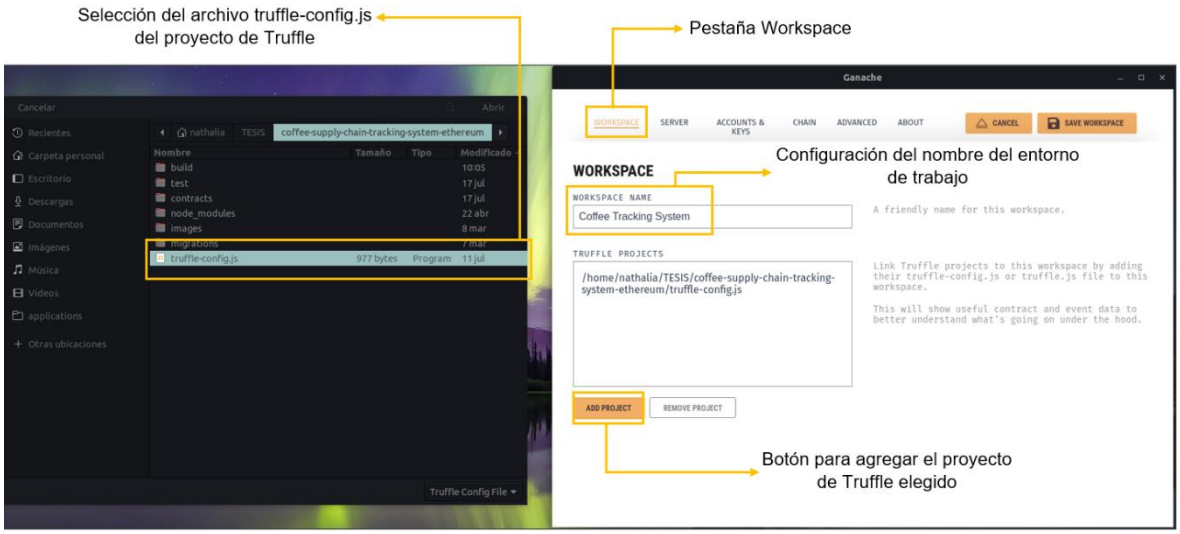


Figura 5.23. Paso 2. Configuración del nombre del entorno y selección del proyecto de Truffle a ejecutar

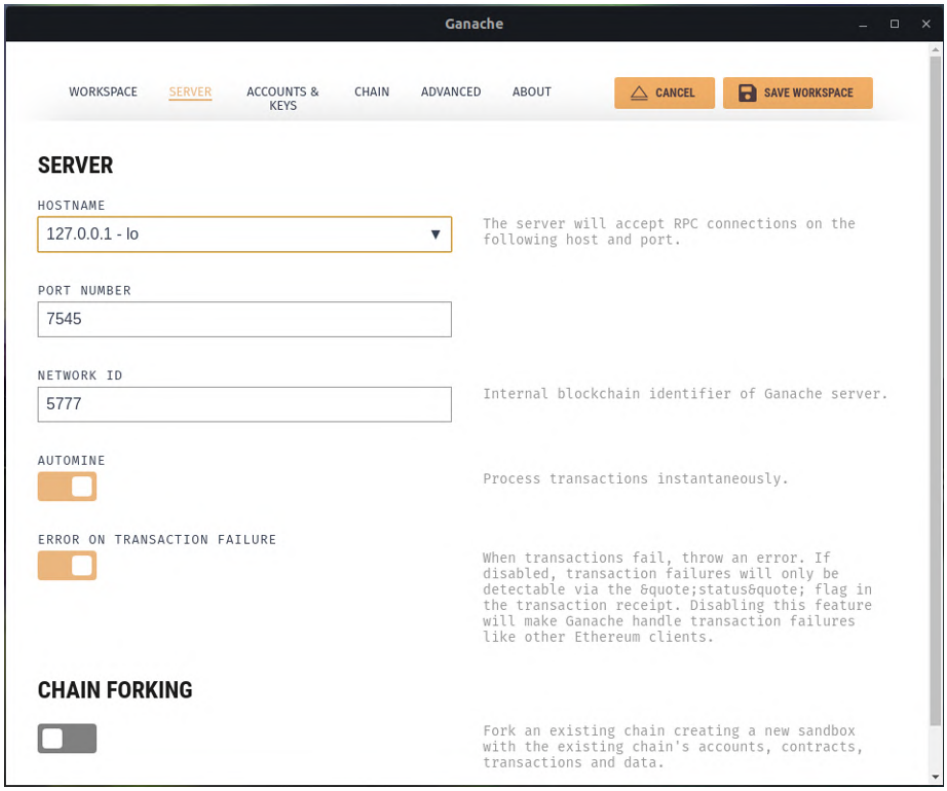


Figura 5.24. Paso 3. Configuración del Hostname y puerto para las conexiones RPC con un nodo de servidor en la red de cadena de bloques de Ganache

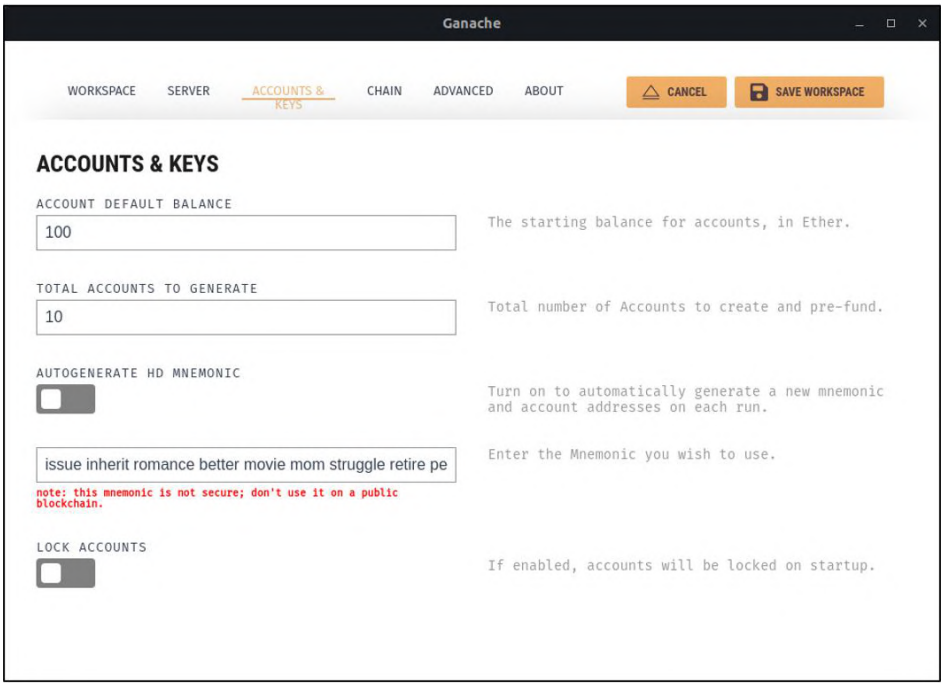


Figura 5.25. Paso 4. Configuración de las cuentas y su balance inicial de ethers, las cuales interactuarán con la cadena de bloques creada

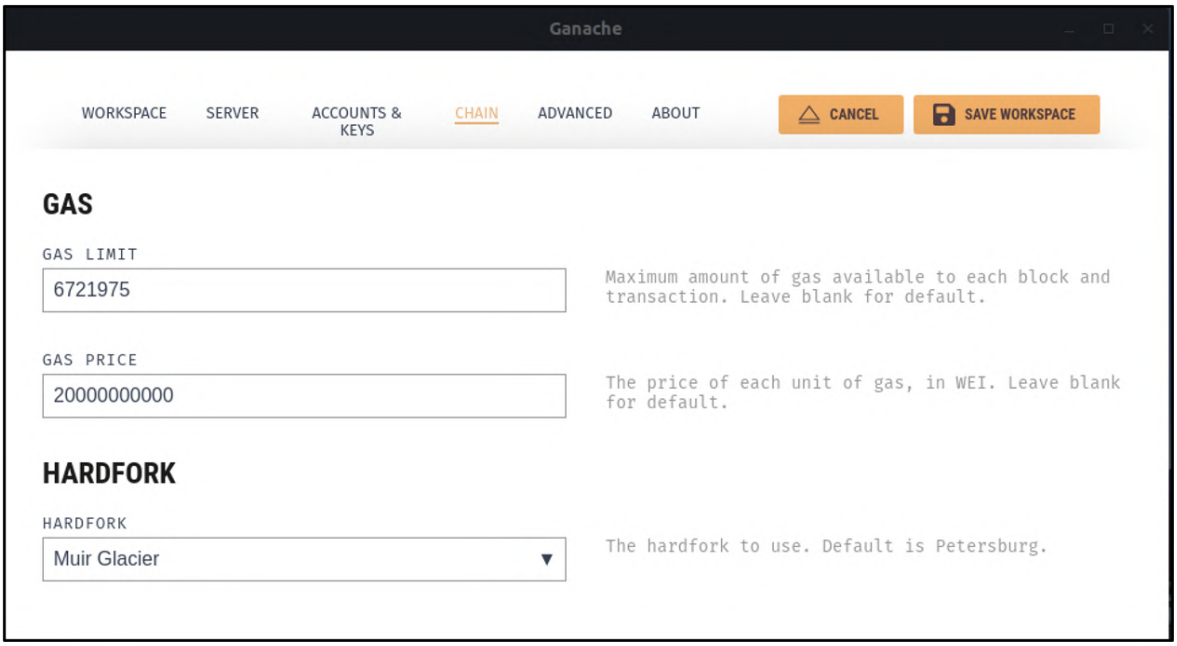


Figura 5.26. Paso 5. Configuración de los parámetros para el precio de la tarifa de transacción y la bifurcación de la red Ethereum que se utilizará para el entorno a crear

Figura 5.27. Paso 6. Guardado de la configuración del entorno

A continuación, se puede observar la red local creada con los parámetros antes definidos. En la pestaña de cuentas se tienen 10 direcciones para interactuar con la blockchain cada una con 100 ethers. En las pestañas de bloques, transacciones y eventos no existe información debido a que aún no se ha realizado ninguna transacción que conlleve algún evento y por ende tampoco hay bloques creados en la red local. En la pestaña de contratos, se observa todos los contratos codificados en el proyecto que han sido detectados a partir

del archivo de configuración del proyecto de Truffle, como ninguno de estos contratos ha sido desplegado aún en la red, todos muestran la marca de no desplegados.

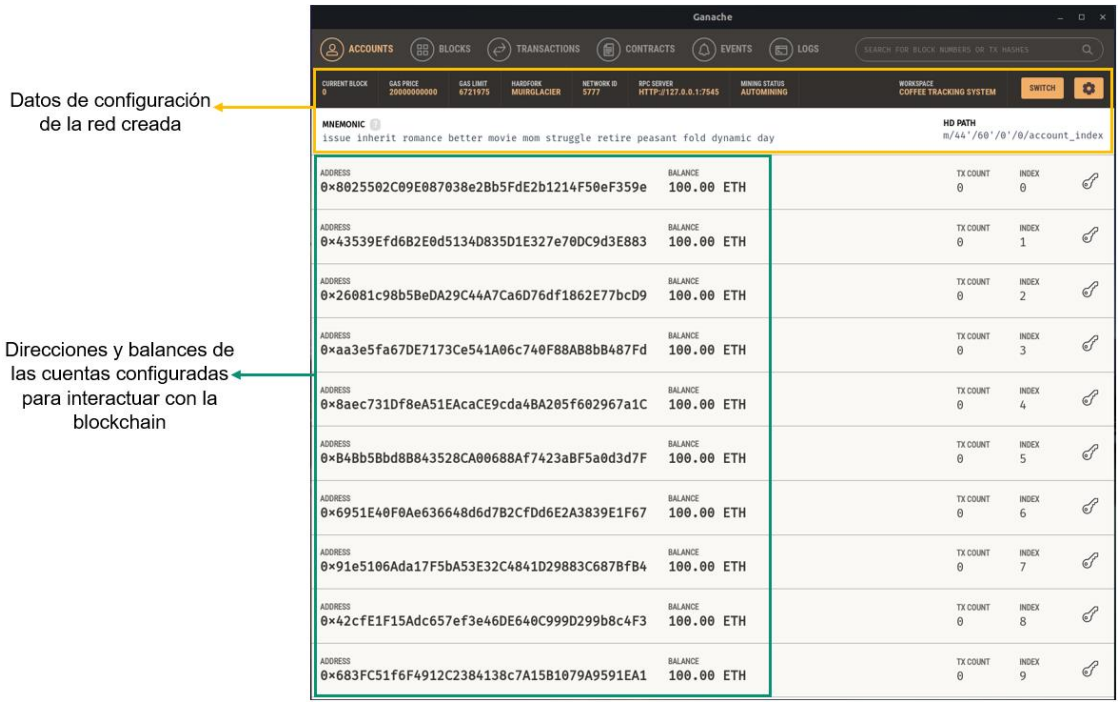


Figura 5.28. Red local creada a partir de la configuración establecida

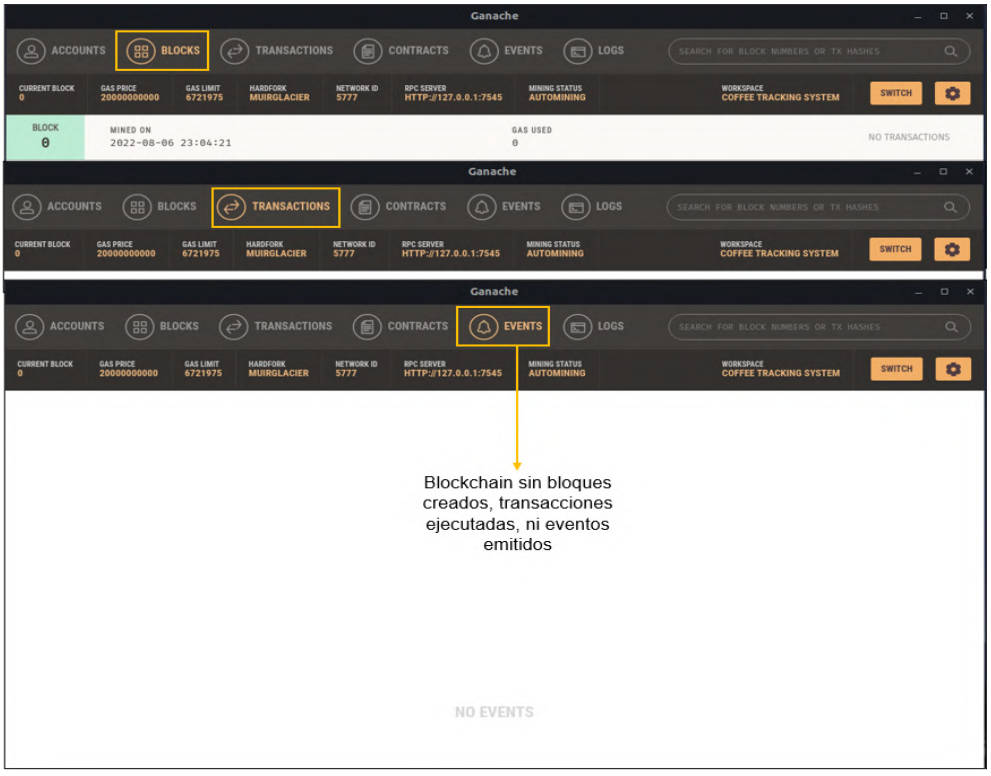


Figura 5.29. Pestañas de Bloques, Transacciones y Eventos sin información, ya que aún no se ha llevado a cabo la ejecución de transacciones

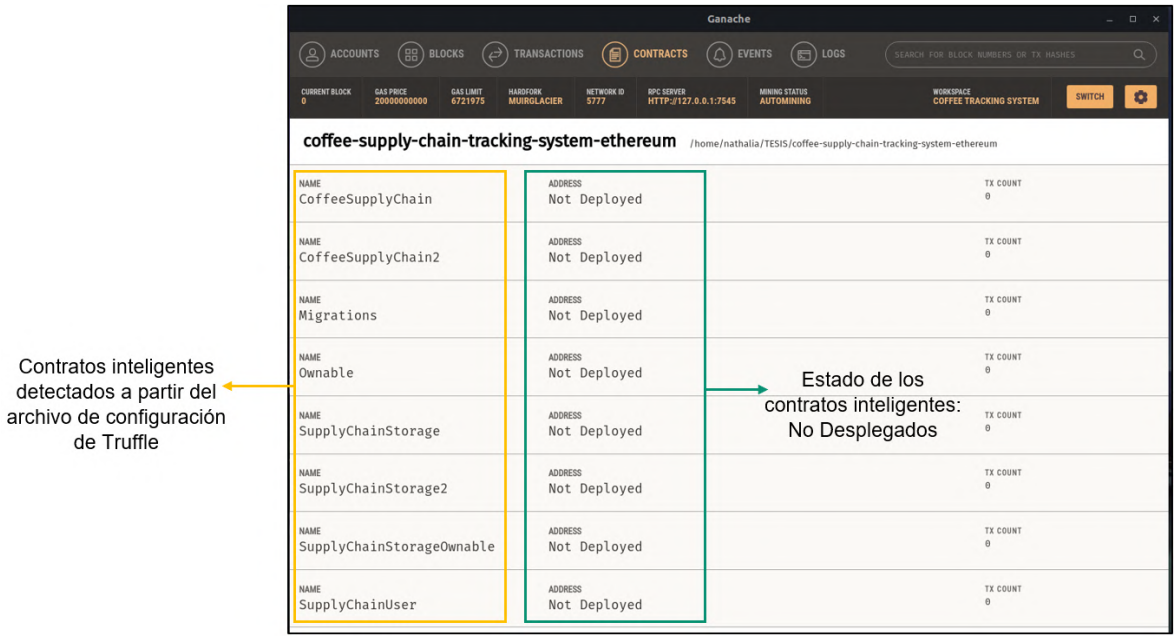


Figura 5.30. Contratos inteligentes detectados a partir del archivo de configuración del proyecto de Truffle, con estado de No Desplegados

ANEXO XX

Definición de los parámetros empleados en la configuración de las redes del ecosistema de Ethereum en el script truffle-config.js.

Tabla 5.4. Definición de los parámetros empleados en la configuración de las redes del archivo truffle-config.js

Parámetro	Significado
host	Dirección IP del nodo de conexión. Si se utiliza un proveedor ya no se debe definir este parámetro ni el puerto.
port	Puerto del nodo de conexión.
gas	Límite de unidades de gas de la tarifa de transacción empleado durante las migraciones de los contratos en la blockchain.
network_id	Número identificador de red.
provider	Instancia del proveedor de nodo que emplea Truffle para conectarse con una cadena de bloques del ecosistema de Ethereum.
from	Dirección de cuenta de la billetera digital empleada en las transacciones que Truffle realiza para ejecutar las migraciones de los contratos en la cadena de bloques.
skyDryRun	Establece si no se desea probar la ejecución del despliegue localmente antes de realizar la migración real.
timeoutBlocks	Número de bloques que se espera si la transacción no es verificada en el bloque actual.
confirmations	Número de confirmaciones a esperar entre implementaciones.

ANEXO XXI

Pasos para crear un proyecto en Infura y obtener un nodo de conexión con una blockchain del ecosistema de Ethereum.

1. Registrarse en la página web de https://infura.io/.

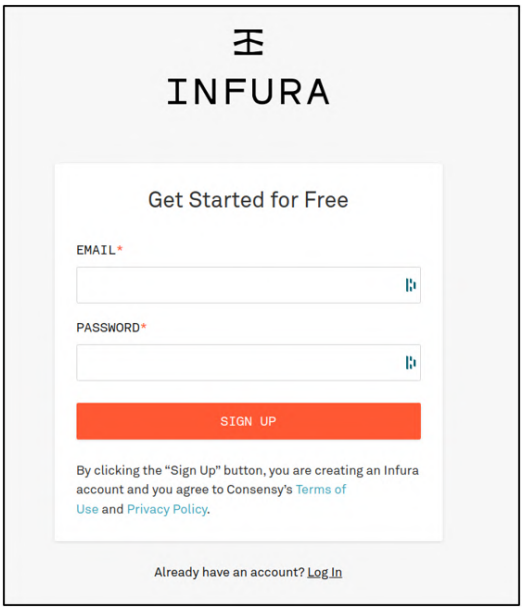


Figura 5.31. Paso 1. Registro en la página web de https://infura.io/

2. En el panel de control elegir la opción CREATE NEW KEY.

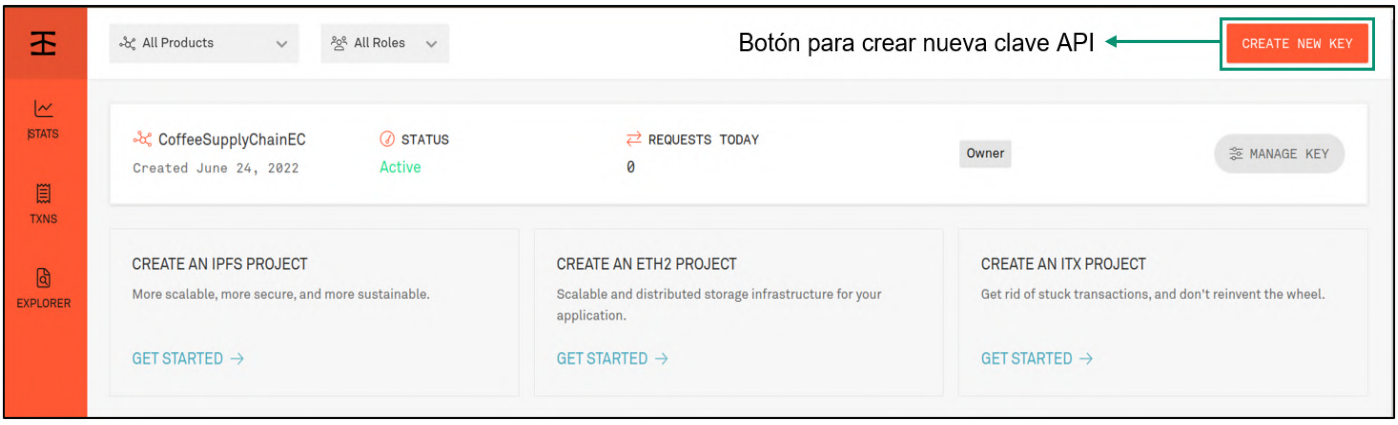


Figura 5.32. Paso 2. Elección de la opción de crear nueva clave API

3. En la ventana emergente, elegir en el campo NETWORK, a la opción Web3 API(Formerly Ethereum) y definir el nombre del proyecto en el campo NAME.

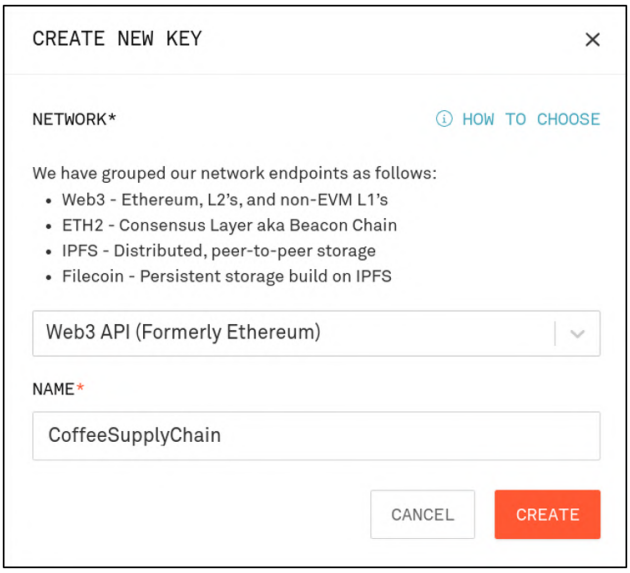


Figura 5.33. Paso 3. Configuración del proyecto: opción de red Web3 API y definición del nombre del proyecto

4. Con el proyecto creado, ir a la opción MANAGE KEY.

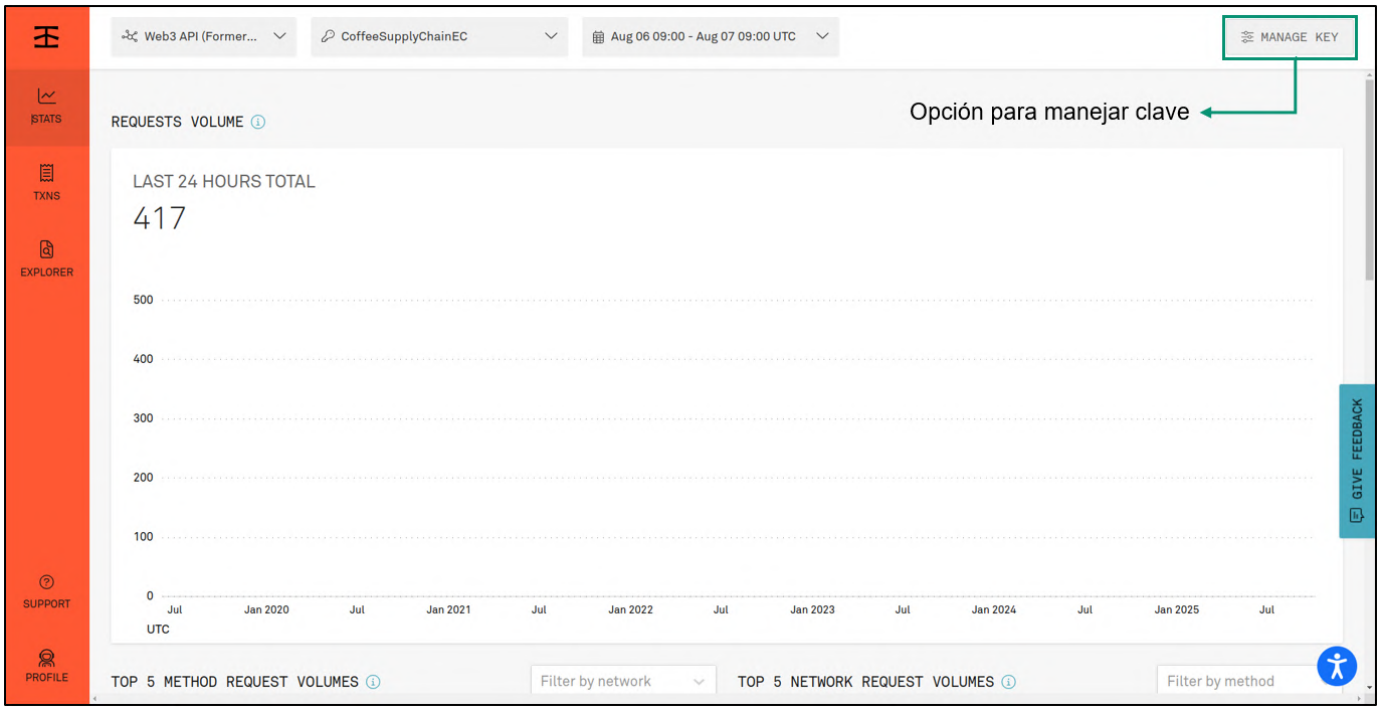


Figura 5.34. Paso 4. Selección del botón de manejar clave

5. Elegir la red a la que se conectará el nodo en la sección de NETWORK ENDPOINTS, en este caso la testnet de Rinkeby. La URL que se despliega es la requerida para la configuración de la red en Truffle.

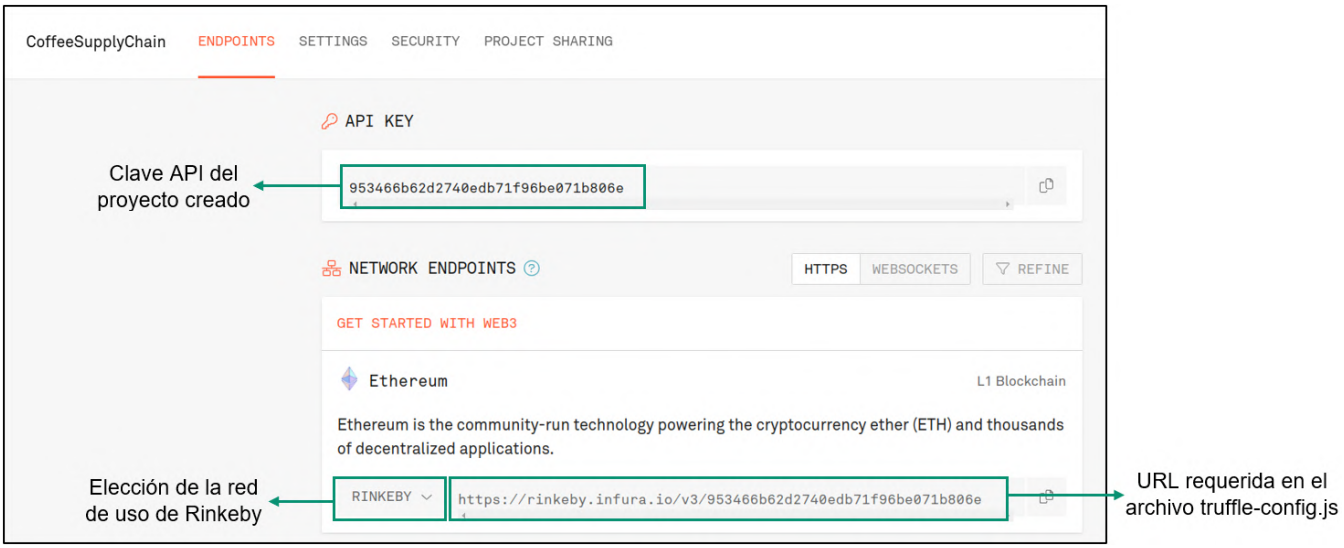


Figura 5.35. Paso 5. Elección de la red Rinkeby en la que se empleará el nodo, copiar la URL desplegada para usarla en el archivo de configuración de Truffle

ANEXO XXII

Pasos para añadir el complemento de verificación y generar una clave API en Etherscan empleada en la verificación de los contratos inteligentes desplegados en la blockchain.

1. Instalar el paquete de verificación en el directorio del proyecto: *npm i -D truffle-plugin-verify.*

Figura 5.36. Paso 1. Instalación del paquete de verificación en el directorio del proyecto a través de la terminal de comandos

2. Generar una clave API en una cuenta creada en el sitio web de Etherscan, para esto se realiza lo siguiente:

   2.1. Ingresar a URL https://etherscan.io/ y registrarse. A continuación, ingresar en la cuenta creada.

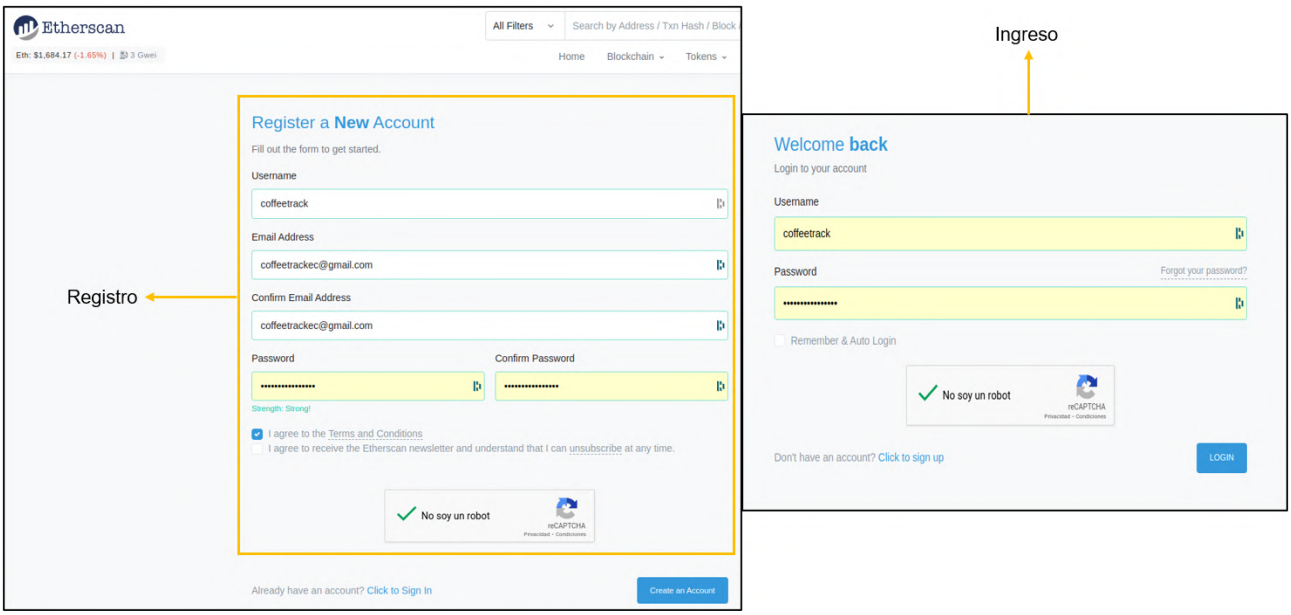


Figura 5.37. Paso 2.1. Generación de una clave API: registro e ingreso en cuenta de Etherscan

2.2. En el panel de control de usuario, elegir la opción API Keys. En la nueva vista cargada, seleccionar la opción Add y en la ventana desplegada Create API Key, ingresar un nombre para el proyecto.

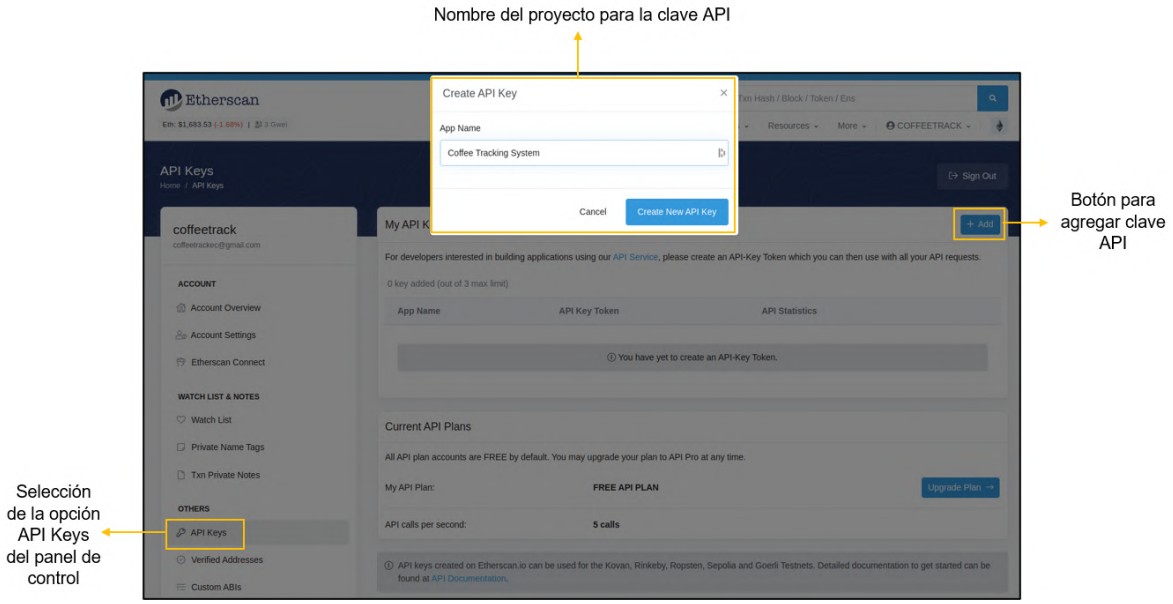


Figura 5.38. Paso 2.2. Generación de una clave API: elección de la opción API Keys en el panel de control, selección de la opción Add y definición del nombre del proyecto

A continuación, se obtiene la clave API generada. Esta debe ser copiada y agregada en el archivo de variables de entorno del proyecto de Truffle para ser llamada en el archivo de configuración truffle-config.js.

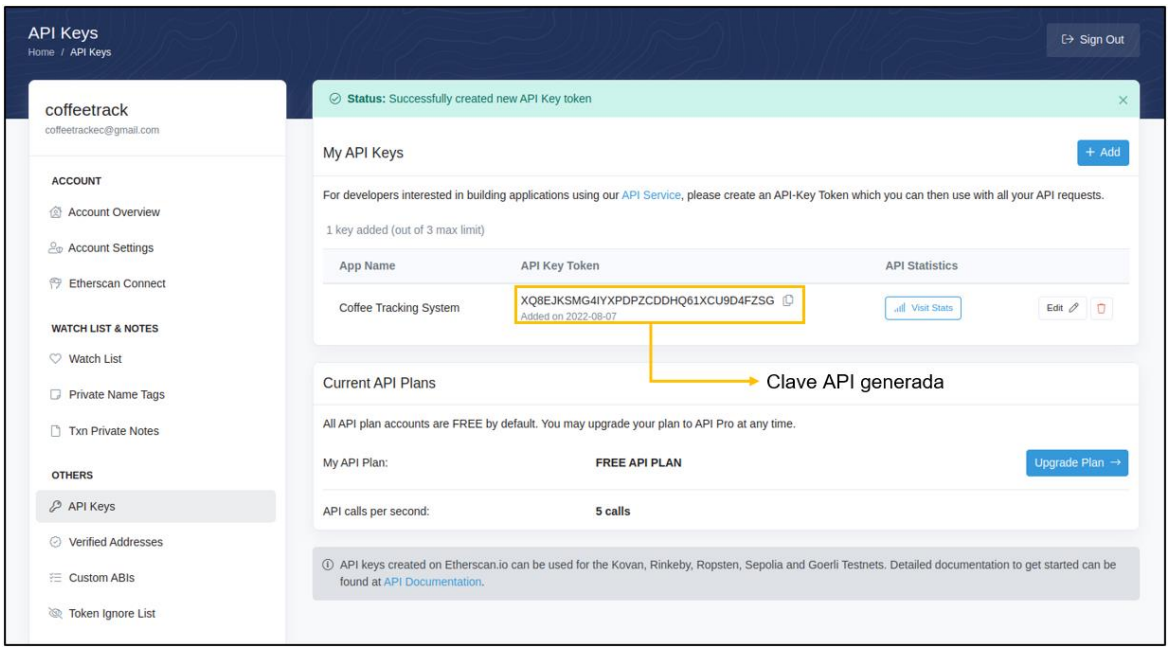


Figura 5.39. Paso 2. Clave API generada

3. Agregar el complemento instalado y la clave API generada en el archivo de configuración.

```
// Permite el acceso a la variable de entorno de la clave API de Etherscan
const { env } = require("process");
require("dotenv").config();

module.exports = {
    // Complemento para la verificación de los contratos
    networks: {
        plugins: ["truffle-plugin-verify"],
        api_keys: {
            etherscan: `${env.MY_API_KEY}`,
        },
    },
};
```

Figura 5.40. Paso 3. Definición del complemento instalado y la clave API generada en el archivo de configuración

ANEXO XXIII

Script de configuración de Truffle (truffle-config.js) en donde se definen las redes de conexión (local con Ganache, testnet con Rinkeby y real con Polygon), la versión de compilación de los contratos y el complemento de verificación.

```
// Definición del paquete HDWalletProvider
// Habilita a Truffle para firmar transacciones
const HDWalletProvider = require("@truffle/hdwallet-provider");
// Permite el acceso a las variables de entorno:
// para el mnemónico y las claves API de Etherscan e Infura
// y la dirección pública de la cuenta MetaMask para el despliegue
const { env } = require("process");
require("dotenv").config();

module.exports = {
    networks: {
        // Definición red local con Ganache
        ganache: {
            host: "localhost",
            port: 7545,
            gas: 6000000,
            network_id: "*",
        },
        // Definición red testnet Rinkeby
        rinkeby: {
            provider: () =>
                new HDWalletProvider(
                    env.MNEMONIC_2,
                    `wss://rinkeby.infura.io/ws/v3/${env.PROJECT_ID}`
                ),
            network_id: 4,
            gas: 6500000,
            confirmations: 2,
            timeoutBlocks: 200,
            networkCheckTimeoutnetworkCheckTimeout: 10000,
            skipDryRun: true,
        // Definición red de Polygon para despliegue en una
        // blockchain real del ecosistema de Ethereum
        matic: {
            provider: () =>
                new HDWalletProvider(env.MNEMONIC, `https://rpc-mumbai.maticvigil.com`),
            network_id: 80001,
            gas: 6500000,
            confirmations: 2,
            timeoutBlocks: 200,
            skipDryRun: true,
            from: `${env.MY_ADDRESS}`,
        },
    },
    // Definición de la versión de compilación
    compilers: {
        solc: {
            version: "0.8.16",
        },
    },
    // Definición del complemento de verificación en Etherscan
    plugins: ["truffle-plugin-verify"],
    api_keys: {
        etherscan: `${env.MY_API_KEY}`,
    },
};
```

Figura 5.41. Archivo truffle-config.js

ANEXO XXIV

Archivo 2_deploy_contracts.js correspondiente a la migración de los contratos inteligentes del sistema de tracking.

```
const SupplyChainStorage = artifacts.require(
    "../contracts/SupplyChainStorage.sol"
);
const SupplyChainStorage2 = artifacts.require(
    "../contracts/SupplyChainStorage2.sol"
);
const CoffeeSupplyChain = artifacts.require(
    "../contracts/CoffeeSupplyChain.sol"
);
const CoffeeSupplyChain2 = artifacts.require(
    "../contracts/CoffeeSupplyChain2.sol"
);
const SupplyChainUser = artifacts.require("../contracts/SupplyChainUser.sol");

module.exports = function (deployer) {
    deployer
        // Despliegue del contrato SupplyChainStorage
        .deploy(SupplyChainStorage)
        // Despliegue del contrato SupplyChainStorage2 con la dirección del contrato desplegado SupplyChainStorage
        .then(() => {
            return deployer.deploy(SupplyChainStorage2, SupplyChainStorage.address);
        })
        // Despliegue del contrato CoffeeSupplyChain con la dirección del contrato desplegado SupplyChainStorage
        .then(() => {
            return deployer.deploy(CoffeeSupplyChain, SupplyChainStorage.address);
        })
        // Despliegue del contrato SupplyChainStorage2 con las direcciones de los contratos
        // desplegados SupplyChainStorage y SupplyChainStorage2
        .then(() => {
            return deployer.deploy(
                CoffeeSupplyChain2,
                SupplyChainStorage.address,
                SupplyChainStorage2.address
            );
        })
        // Despliegue del contrato SupplyChainUser con la dirección del contrato desplegado SupplyChainStorage
        .then(() => {
            return deployer.deploy(SupplyChainUser, SupplyChainStorage.address);
        })
        // Una vez desplegado el contrato SupplyChainStorage, se autoriza la llamada de los otros contratos
        // empleando su dirección de contrato como argumento de la función de autorización
        .then(() => {
            return SupplyChainStorage.deployed();
        })
        .then(async function (instance) {
            await instance.authorizeCaller(SupplyChainStorage2.address);
            await instance.authorizeCaller(CoffeeSupplyChain.address);
            await instance.authorizeCaller(CoffeeSupplyChain2.address);
            await instance.authorizeCaller(SupplyChainUser.address);
            return instance;
        })
        // Una vez desplegado el contrato SupplyChainStorage2, se autoriza la llamada del contrato CoffeeSupplyChain2
        // empleando su dirección de contrato como argumento de la función de autorización
        .then(() => {
            return SupplyChainStorage2.deployed();
        })
        .then(async function (instance) {
            await instance.authorizeCaller(CoffeeSupplyChain2.address);
        })
        .catch(function (error) {
            console.log(error);
        });
};
```

Figura 5.42. Archivo 2_deploy_contracts.js correspondiente a la migración de los contratos inteligentes del sistema de tracking

ANEXO XXV

Especificación de los casos de pruebas codificados para los contratos inteligentes.

Tabla 5.5. Casos de prueba de los contratos inteligentes

Contrato Inteligente	Caso de Prueba	Funciones Empleadas
SupplyChainStorage	“should Authorize”	autorizarLlamada
	“should DeAuthorize”	desautorizarLlamada
	“should Add New User”	setearUsuario
SupplyChainStorage2	“should Authorize”	autorizarLlamada
	“should DeAuthorize”	desautorizarLlamada
SupplyChainUser	“should Update New User”	ingresarUsuario
	“should Add New User from Admin”	actualizarUsuario
	“should all Get User Details”	obtenerUsuario
CoffeeSupplyChain	“should add farm details”	añadirDatosGranja, setearDatosGranja, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get farm details”	obtenerDatosGranja y obtenerGranja
	“should add harvest data”	añadirDatosCosecha, setearDatosCosecha, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get harvest data ''	obtenerDatosCosecha y obtenerCosecha
	“should add process data”	añadirDatosProcesado, setearDatosProcesado, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get process data”	obtenerDatosProcesado y obtenerProcesado
	“should add tasting data”	añadirDatosCatación, setearDatosCatación, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get tasting data”	obtenerDatosCatación y obtenerCatación
	“should add coffee selling data”	añadirDatosVentaCafé, setearDatosVentaCafé, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get coffee selling data”	obtenerDatosVentaCafé y obtenerVentaCafé

CoffeeSupplyChain2	“should add warehouse data”	añadirDatosBodegaje, setearDatosBodegaje, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get warehouse data”	obtenerDatosBodegaje y obtenerBodegaje
	“should add shipping to packer data”	añadirDatosTransporteEmpacador, setearDatosTransporteEmpacador, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get shipping to packer data”	obtenerDatosTransporteEmpacador y obtenerTransporteEmpacador
	“should add shipping to packer data”	añadirDatosEmpacado, setearDatosEmpacado, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get shipping to packer data”	obtenerDatosEmpacado y obtenerEmpacado
	“should add shipping to retailer data”	añadirDatosTransporteRetailer, setearDatosTransporteRetailer, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get shipping to retailer data”	obtenerDatosTransporteRetailer y obtenerTransporteRetailer
	“should add retailer data”	añadirDatosRetailer, setearDatosRetailer, obtenerPróximaAcción y obtenerRol(es)Usuario
	“should get retailer data”	obtenerDatosRetailer y obtenerRetailer

ANEXO XXVI

Registros de consola de todos los casos de prueba verificados al ejecutar los tests.

```
nathalisabel in ~/TESIS/coffee-supply-chain-tracking-system-ethereum
± |main U:9 ✗| →truffle compile

Compiling your contracts...
===========================
✔ Fetching solc version list from solc-bin. Attempt #1
✔ Downloading compiler. Attempt #1.
> Compiling ./contracts/CoffeeSupplyChain.sol
> Compiling ./contracts/CoffeeSupplyChain2.sol
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/Ownable.sol
> Compiling ./contracts/SupplyChainStorage.sol
> Compiling ./contracts/SupplyChainStorage2.sol
> Compiling ./contracts/SupplyChainStorageOwnable.sol
> Compiling ./contracts/SupplyChainUser.sol
> Artifacts written to /home/nathalia/TESIS/coffee-supply-chain-tracking-system-ethereum/build/contracts
> Compiled successfully using:
   - solc: 0.8.16+commit.07a7930e.Emscripten.clang
```

Figura 5.43. Compilación de los contratos inteligentes previo a la ejecución del entorno de pruebas

Tras la compilación se puede observar en el directorio del proyecto que se genera una carpeta *build* que contiene los artefactos de los códigos compilados. Un artefacto es un archivo JSON con información útil relacionada con el contrato inteligente como el ABI, el código en bytes del contrato, los detalles de implementación, versión del compilador, etc. El artefacto de un contrato es requerido para poder interactuar con el lenguaje de programación de las aplicaciones web, ya que las aplicaciones web no se codifican en Solidity y estas tampoco entienden el código en bytes de la EVM, el artefacto contiene un objeto de información denominado ABI que contiene todas las funcionalidades del contrato inteligente, de esta manera el lenguaje de alto nivel de las aplicaciones web pueden interpretar el archivo JSON y usar las funciones del contrato inteligente.

Figura 5.44. Carpeta *build* creada tras la compilación de los contratos inteligentes

Se observa que los 28 casos de prueba fueron resueltos exitosamente en el entorno de pruebas de JavaScript.

Figura 5.45. Casos de prueba ejecutados exitosamente

Debido a que el entorno de prueba de Truffle utiliza la red local de Ganache, se puede observar en la aplicación de Ganache que se ha llevado a cabo interacción con la red ya que el balance de la primera cuenta es inferior y el número de bloques es superior al inicial.

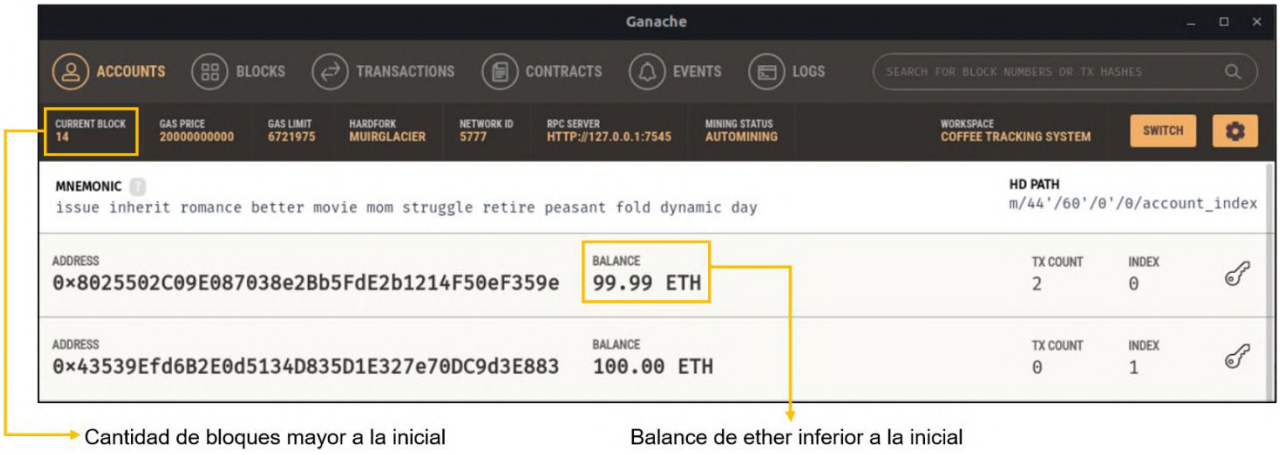


Figura 5.46. Blockchain local de Ganache después de la ejecución del entorno de pruebas con Truffle

ANEXO XXVII

Registros de consola de los contratos desplegados en la red local con Ganache y en la red testnet Rinkeby.

En los registros de consola, se observa que el primer contrato que se despliega corresponde a Migrations.sol, este contrato viene por defecto al iniciar un proyecto en Truffle y permite dar seguimiento al despliegue de los demás contratos codificados. Como resultado del despliegue se obtiene información referente a: la dirección del contrato en la blockchain, el bloque en el que se incluyó la transacción, la marca de tiempo del bloque, la cuenta que llevo a cabo el despliegue, el costo total del despliegue. En el caso de la testnet de Rinkeby, los contratos creados se pueden consultar a través de su dirección de contrato empleando el explorador Etherscan.

a) Despliegue en la red local con Ganache

```
nathalisabel in ~/TESIS/coffee-supply-chain-tracking-system-ethereum
± |dev-nath U:2 ✗| →truffle migrate --network ganache

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'ganache'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > Blocks: 0            Seconds: 0   > transaction hash:    0xa37bd19460fe0389cb62a6b5f3d6caba95b3995ff67dc8963b6ca83e05b091f3
   > Blocks: 0            Seconds: 0
   > contract address:    0x050ce63C5bD1999d9843b0D729E4c619357ae7cf
   > block number:        26
   > block timestamp:     1659949461
   > account:             0x8025502C09E087038e2Bb5FdE2b1214F50eF359e
   > balance:             99.38746786
   > gas used:            248854 (0x3cc16)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00497708 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00497708 ETH
```

Figura 5.47. Primer contrato desplegado (*Migrations.sol*) en la red local

Figura 5.48. Implementación de los contratos *SupplyChainStorage.sol* y *SupplyChainStorage2.sol* en la red local

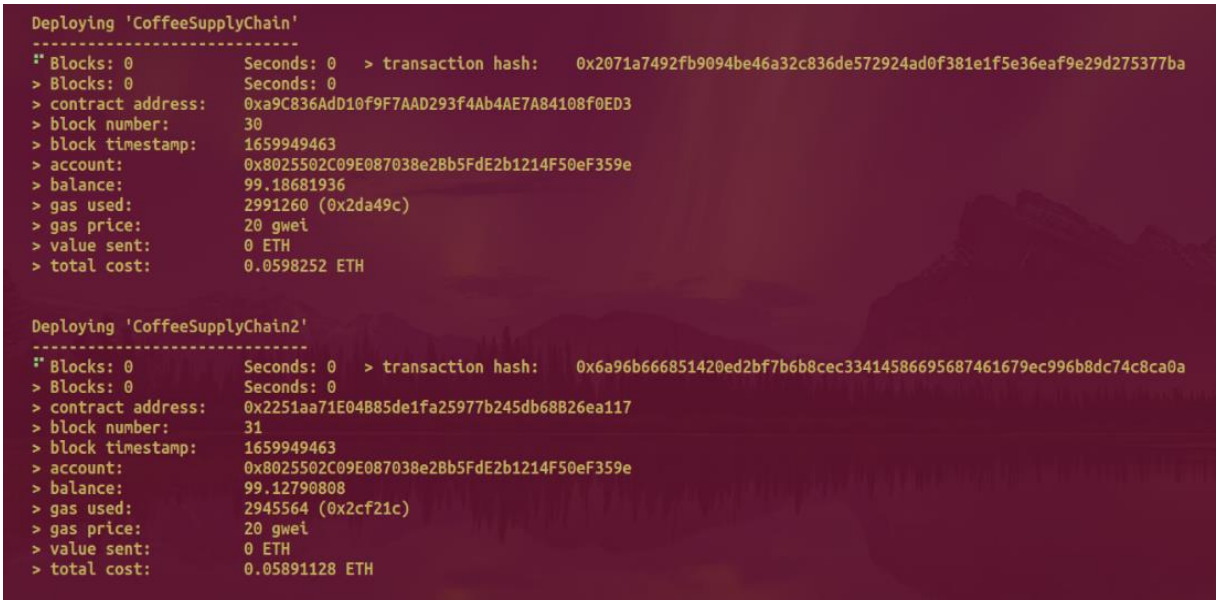


Figura 5.49. Implementación de los contratos *CoffeeSupplyChain.sol* y *CoffeeSupplyChain2.sol* en la red local

Una vez desplegados todos los contratos, se obtiene un resumen del despliegue en el que se especifican la cantidad de implementaciones llevadas a cabo y el costo total.

```
Deploying 'SupplyChainUser'
---------------------------
> Blocks: 0            Seconds: 0   > transaction hash:    0xf193241291dc3a385a8d3fd490140f3ea83d37c57ba6b61721601d2dbd8dafed
> Blocks: 0            Seconds: 0
> contract address:    0x42dAFd41618f78922969299c27287C400468A039
> block number:        32
> block timestamp:     1659949463
> account:             0x8025502C09E087038e2Bb5FdE2b1214F50eF359e
> balance:             99.10478954
> gas used:            1155927 (0x11a357)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.02311854 ETH


> Saving migration to chain.
> Saving artifacts
-------------------------------------
> Total cost:          0.28182806 ETH

Summary
=======
> Total deployments:   6
> Final cost:          0.28680514 ETH
```

Figura 5.50. Implementación del contrato *SupplyChainUser.sol* en la red local y el resumen de las implementaciones ejecutadas

En la blockchain local de Ganache se evidencia las migraciones ejecutadas tanto en la cantidad de bloques creados como en el balance de la primera cuenta.

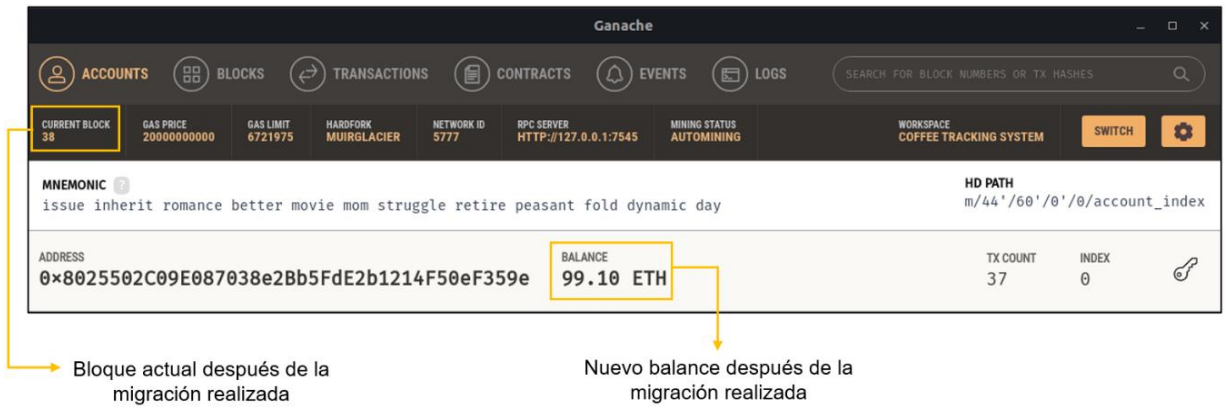


Figura 5.51. Blockchain local de Ganache tras la implementación de los contratos

Se observa en la pestaña de transacciones de la aplicación de escritorio de Ganache, la creación de los cinco contratos inteligentes.

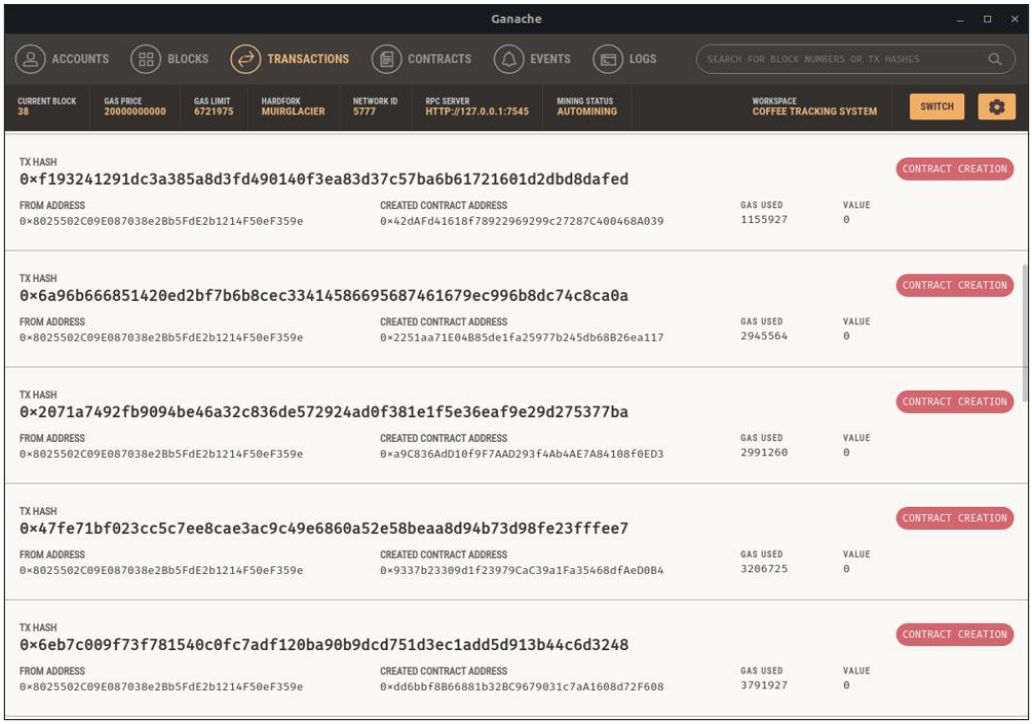


Figura 5.52. Transacciones correspondientes a la creación de los contratos inteligentes

Así también, se observa las transacciones correspondientes a las llamadas de autorización llevadas a cabo por los contratos SupplyChainStorage.sol y SupplyChainStorage2.sol, lo cual se definió en el archivo de migración de la Sección 2.4.7.

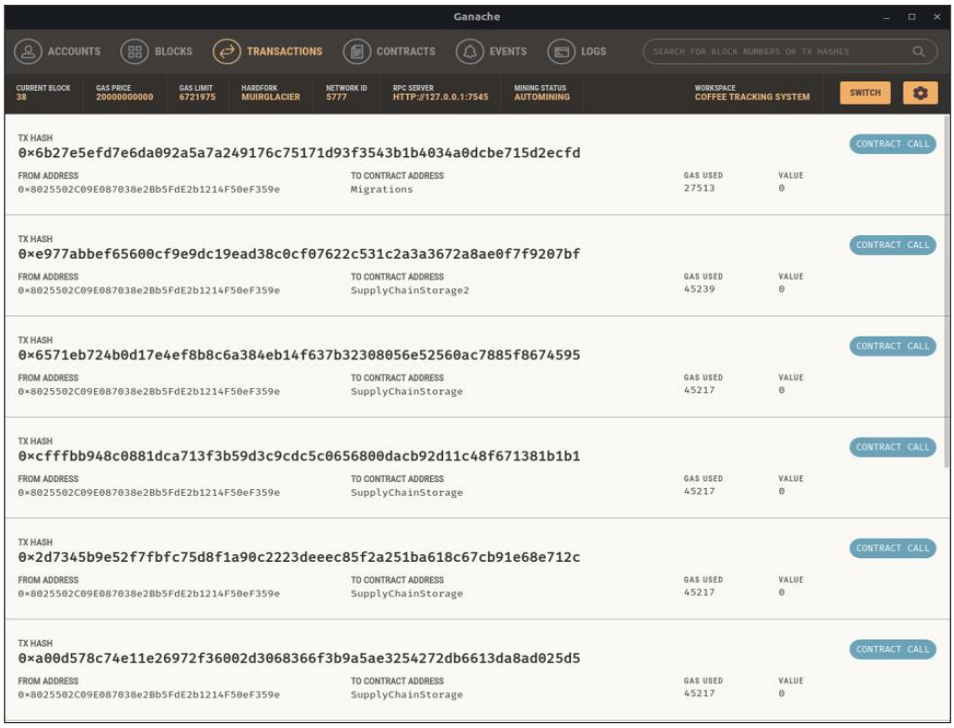


Figura 5.53. Transacciones correspondientes a las llamadas de autorización ejecutadas por los contratos SupplyChainStorage.sol y SupplyChainStorage2.sol

En la pestaña de contratos, se observa que la etiqueta de los contratos describe que ya fueron desplegados en la red local.

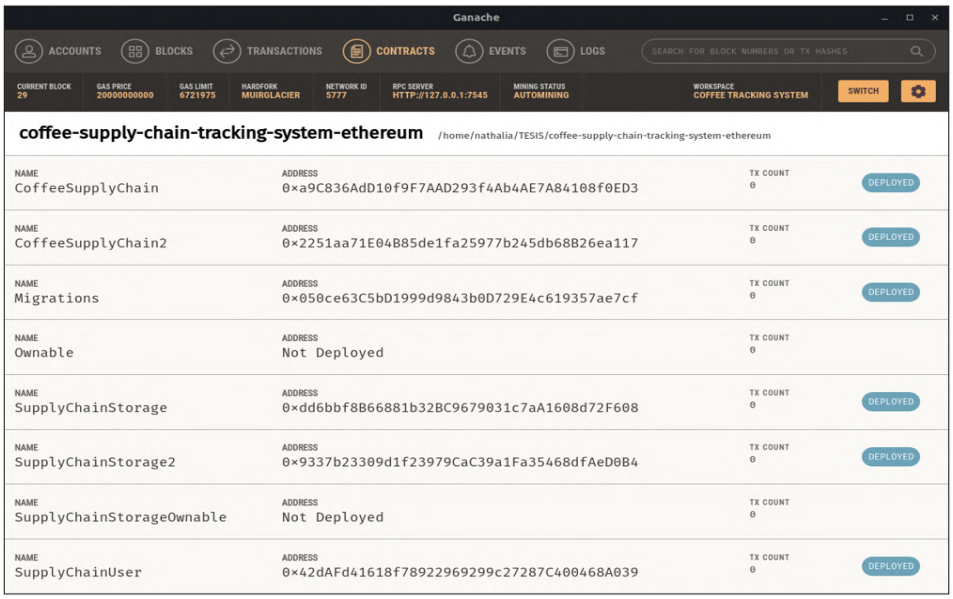


Figura 5.54. Contratos implementados en la red mostrados en Ganache

Como se muestra en los diagramas de secuencias del Anexo IX y Anexo X, las funciones de autorización emiten eventos en la cadena de bloques, esto se observa en la Figura 5.55.

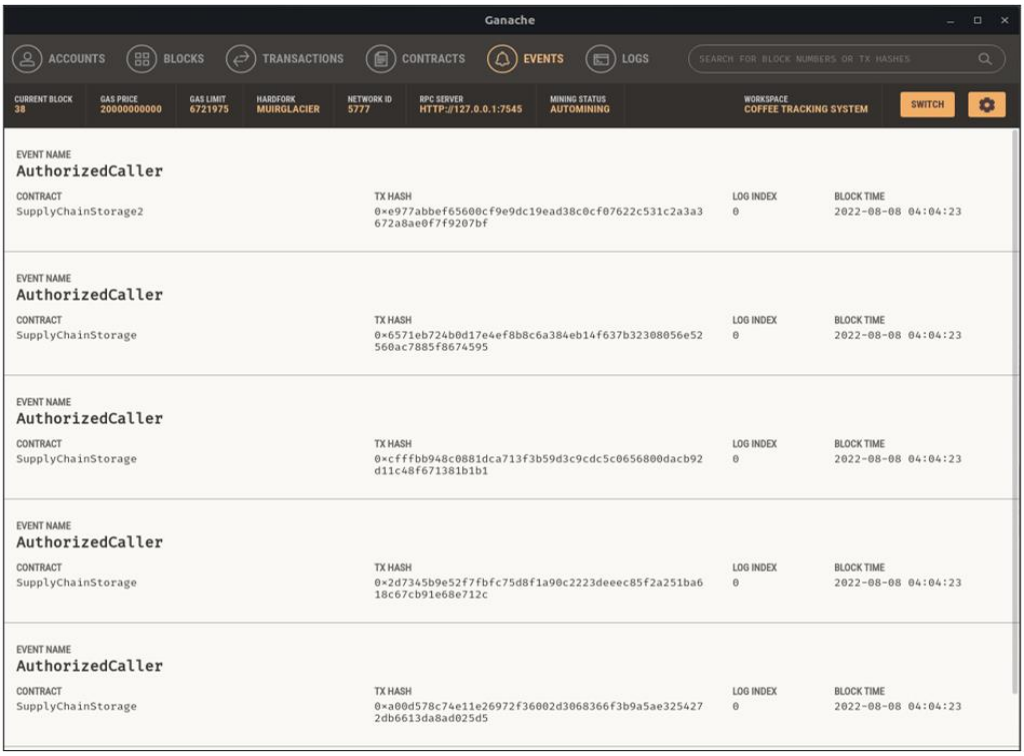


Figura 5.55. Eventos emitidos en la red local como resultado de la ejecución de las funciones de autorización

b) Despliegue en la red testnet Rinkeby

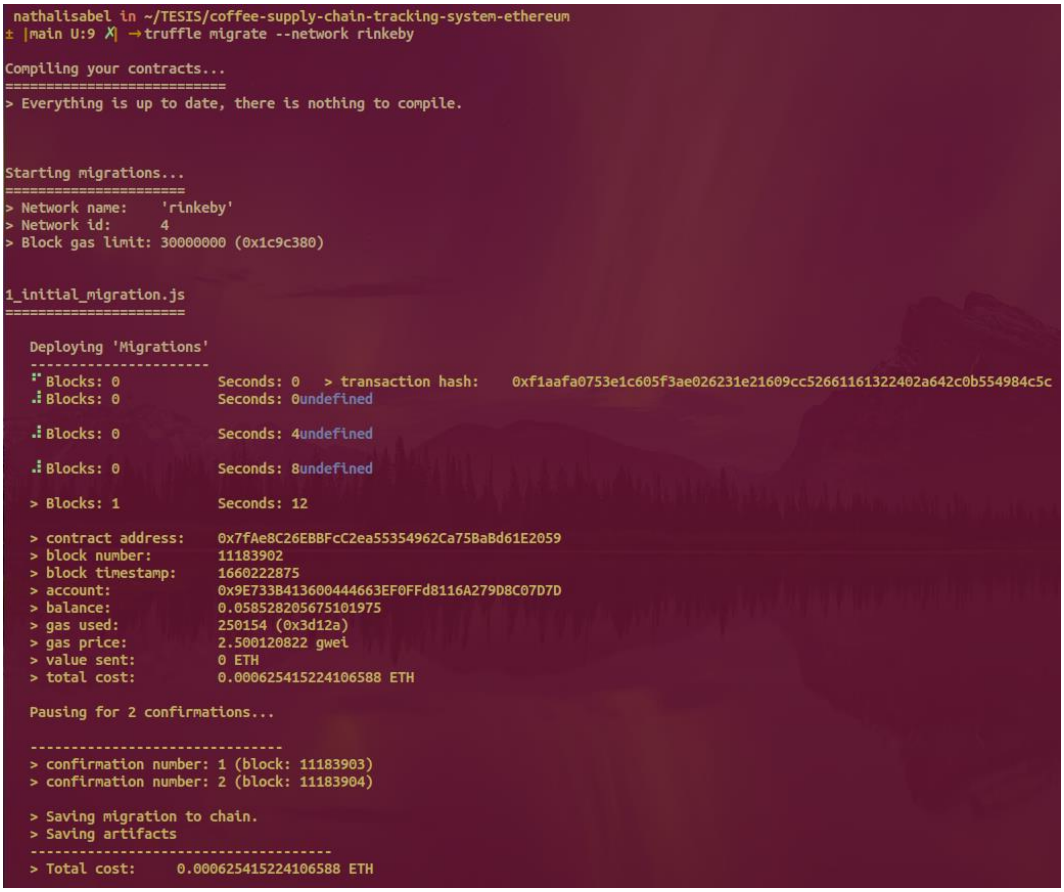


Figura 5.56. Primera migración del contrato Migrations.sol ejecutada

La migración de los contratos muestra en consola información correspondiente a la dirección del contrato desplegado en la red Rinkeby, esta dirección puede ser consultada en la página del explorador de transacciones, Etherscan (https://etherscan.io/), de esta manera se pueden seguir todas las transacciones que se lleven a cabo con este contrato, correspondiente a la ejecución de sus funciones. Otro dato que se muestra como resultado de la migración es la dirección pública de cuenta MetaMask que realizó el despliegue, en este caso esta dirección corresponde a la del administrador del sistema de tracking. De igual manera, se puede seguir todas las transacciones de la cuenta del administrador llevadas a cabo en la red de Rinkeby usando el explorador mencionado.

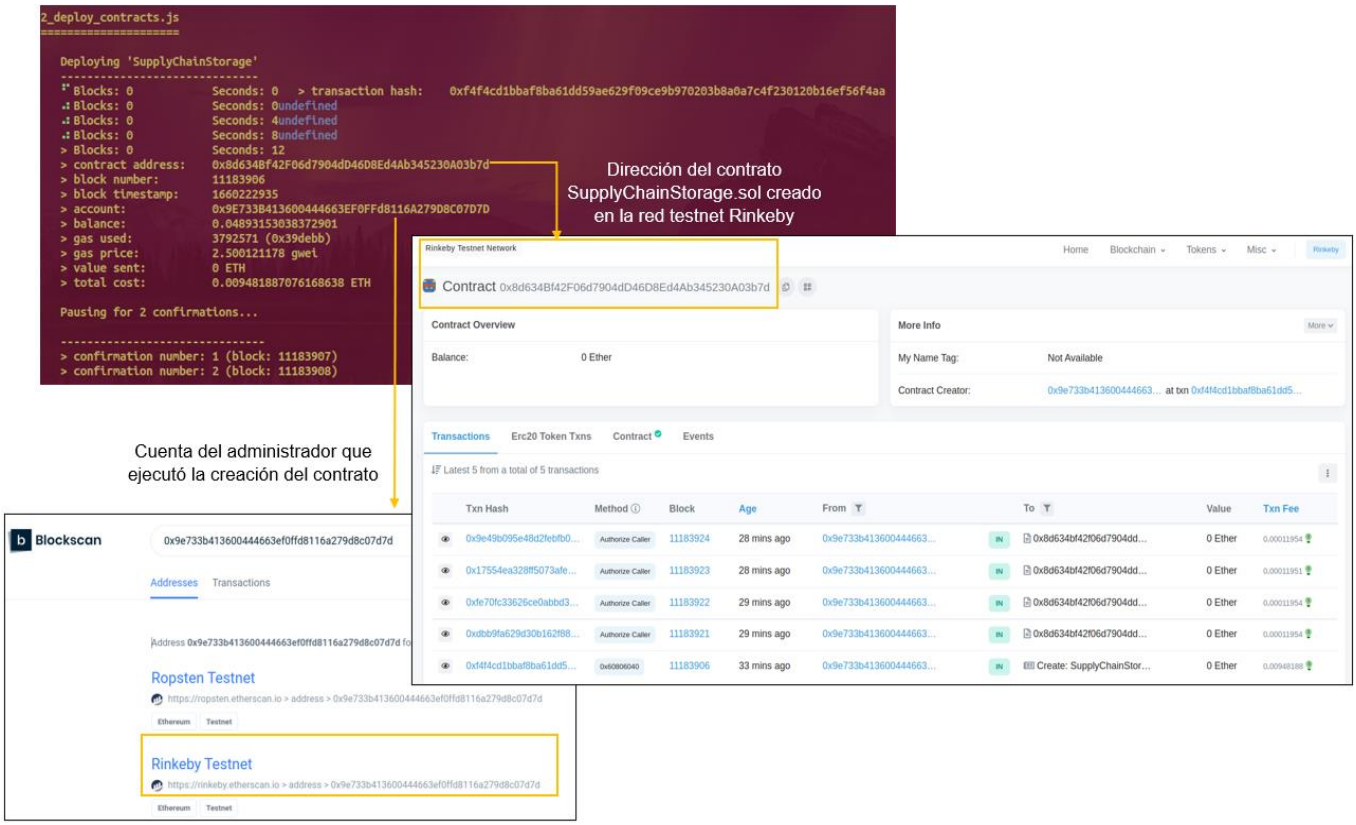


Figura 5.57. Información de migración y dirección consultada en el explorador Etherscan del contrato SupplyChainStorage.sol

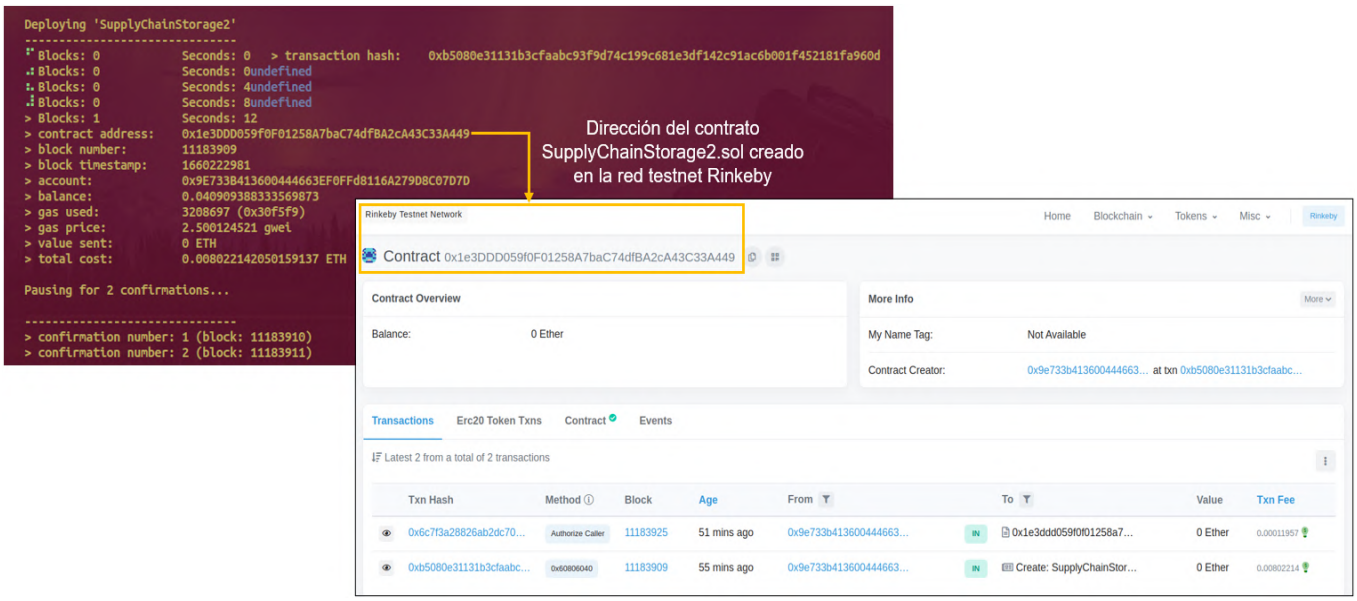


Figura 5.58. Información de migración y dirección consultada en el explorador Etherscan del contrato SupplyChainStorage2.sol

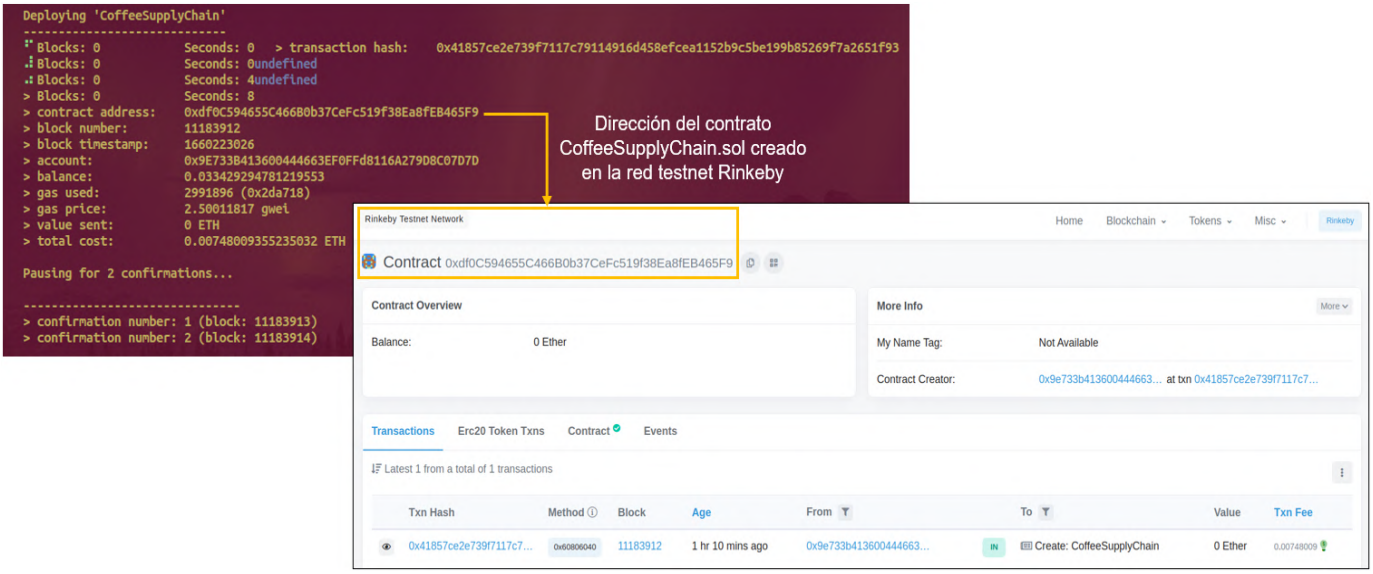


Figura 5.59. Información de migración y dirección consultada en el explorador Etherscan del contrato CoffeeSupplyChain.sol

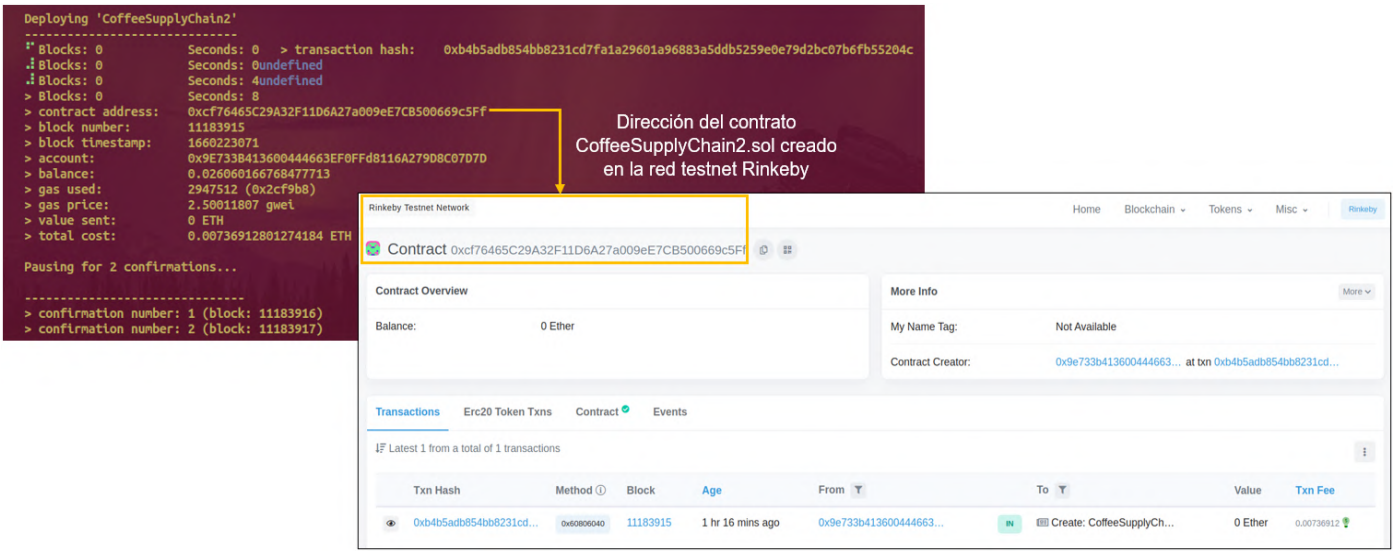


Figura 5.60. Información de migración y dirección consultada en el explorador Etherscan del contrato CoffeeSupplyChain2.sol

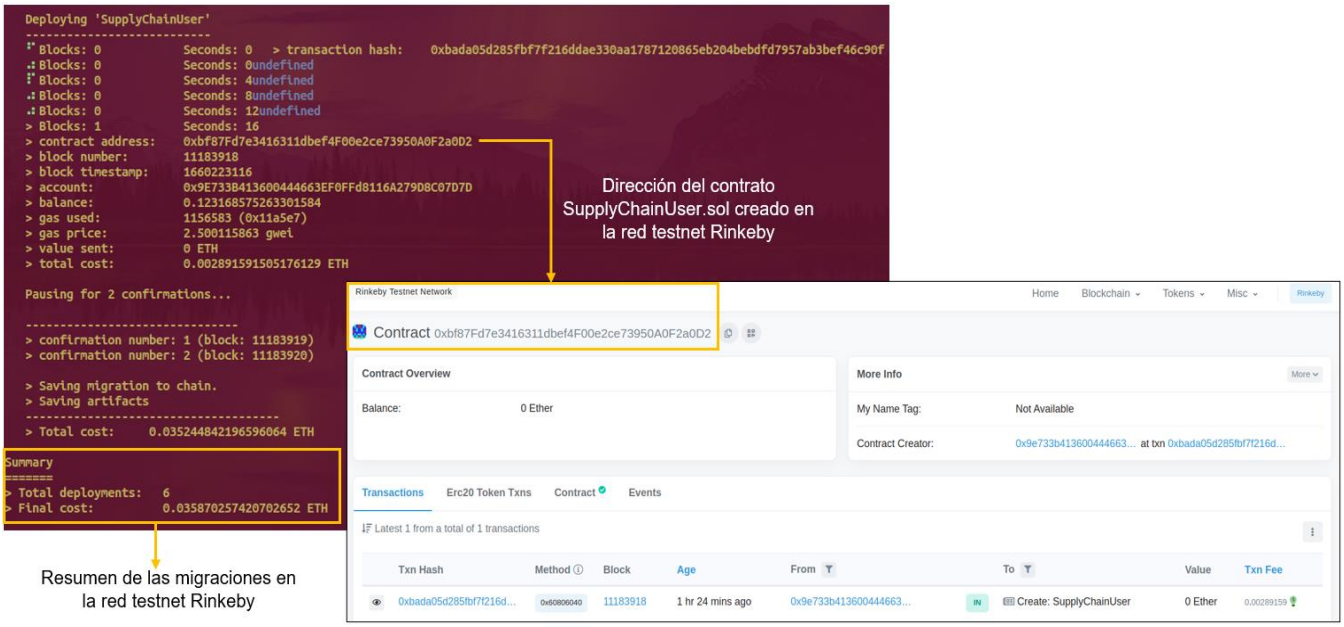


Figura 5.61. Información de migración y dirección consultada en el explorador Etherscan del contrato SupplyChainUser.sol

En la Figura 5.62, se observa la consulta de la dirección de cuenta del administrador en el explorador Etherscan en donde se evidencia las transacciones llevadas a cabo para la creación de los contratos inteligentes y las transacciones de ejecución de las funciones de autorización definidas en el script de migración codificado.

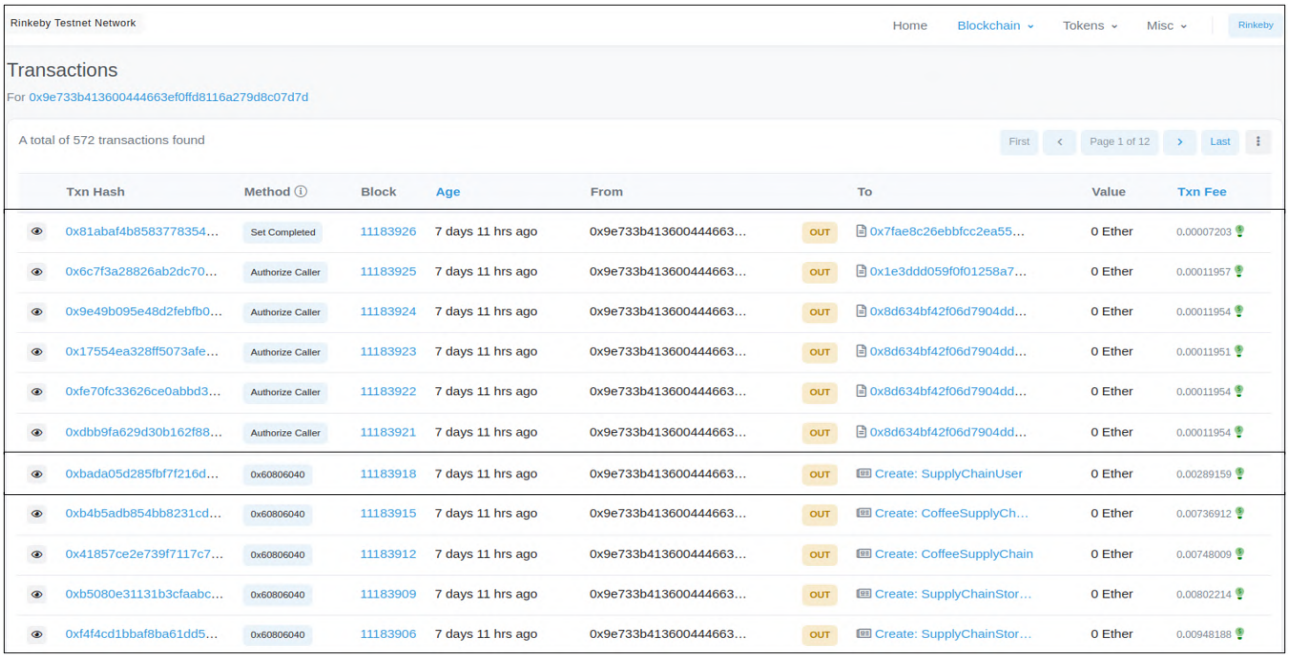


Figura 5.62. Transacciones de creación de los contratos y ejecución de las funciones de autorización por parte del administrador del sistema de tracking

ANEXO XXVIII

Capturas de pantalla de la ejecución del comando de verificación de contratos en la terminal y la consulta de los contratos verificados en el explorador Etherscan.

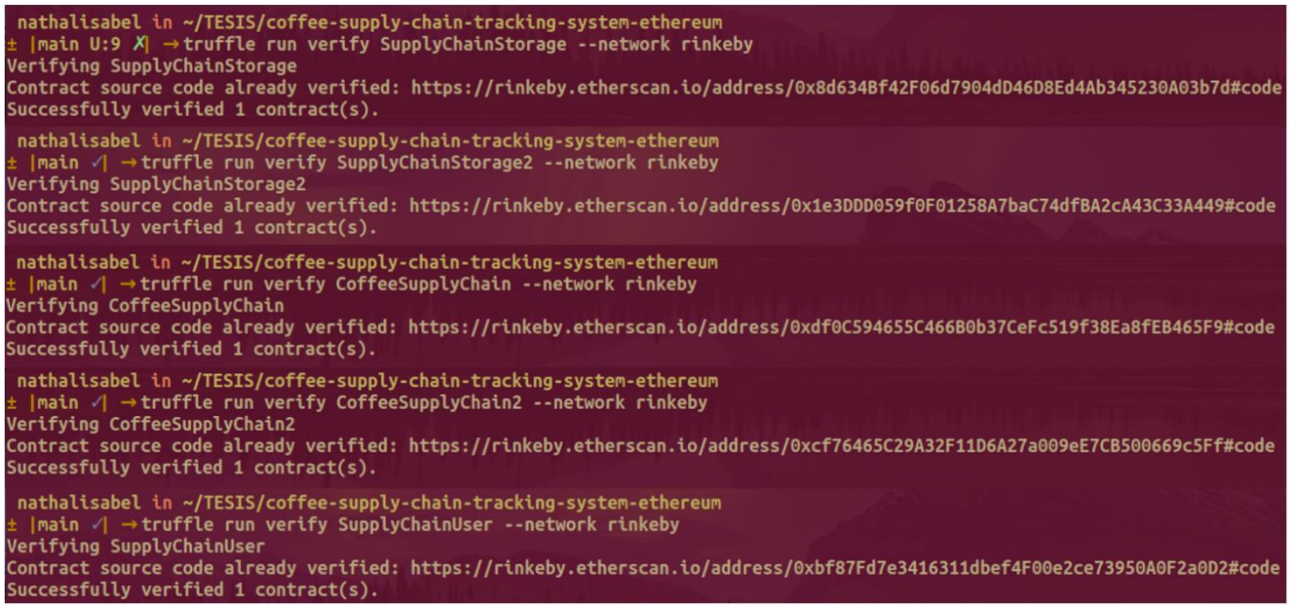


Figura 5.63. Registro de consola de la ejecución de comandos de verificación para los contratos desplegados

Cuando los contratos inteligentes son verificados, estos muestran su código fuente de Solidity en Etherscan. Debido a que los contratos durante la migración son traducidos a bytecode que es entendido por la EVM, estos al ser publicados en el explorador de blockchain, carecen de su código fuente en Solidity, por lo que la verificación es requerida para facilitar a los usuarios la lectura del contrato en lenguaje de alto nivel y transparentar el manejo de la información por parte de las aplicaciones que emplean el sistema descentralizado de blockchain.

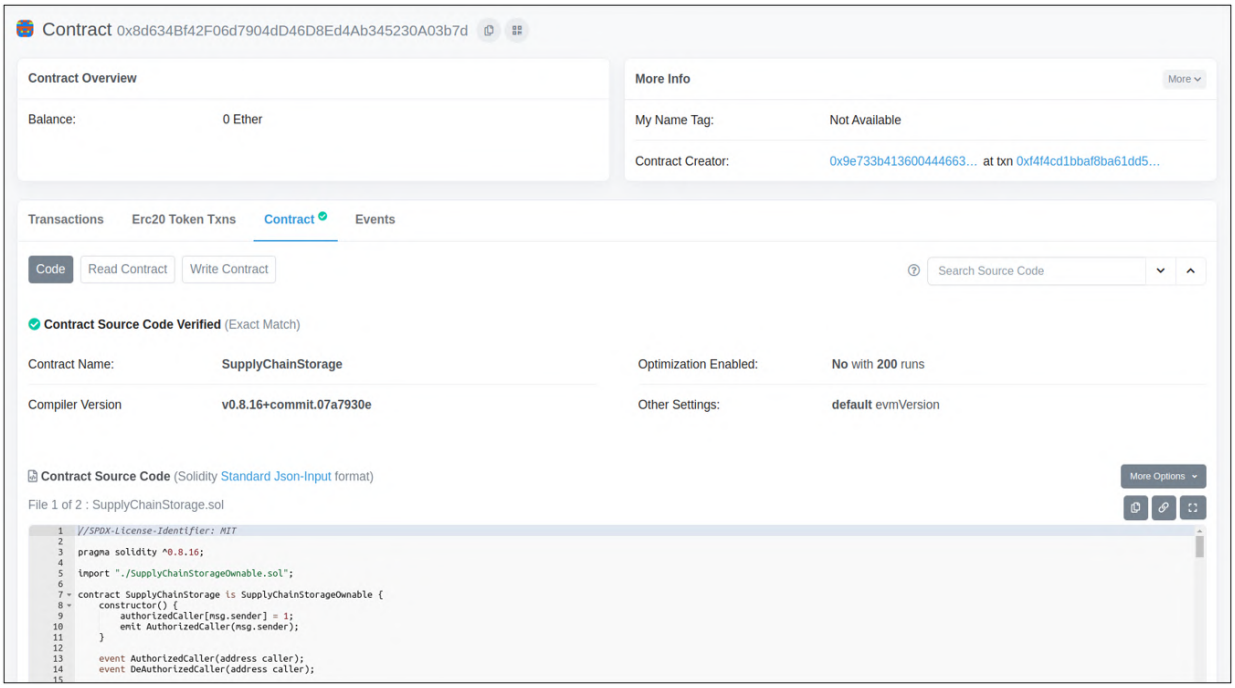


Figura 5.64. Código fuente en Solidity del contrato inteligente SupplyChainStorage.sol mostrado en Etherscan tras su verificación

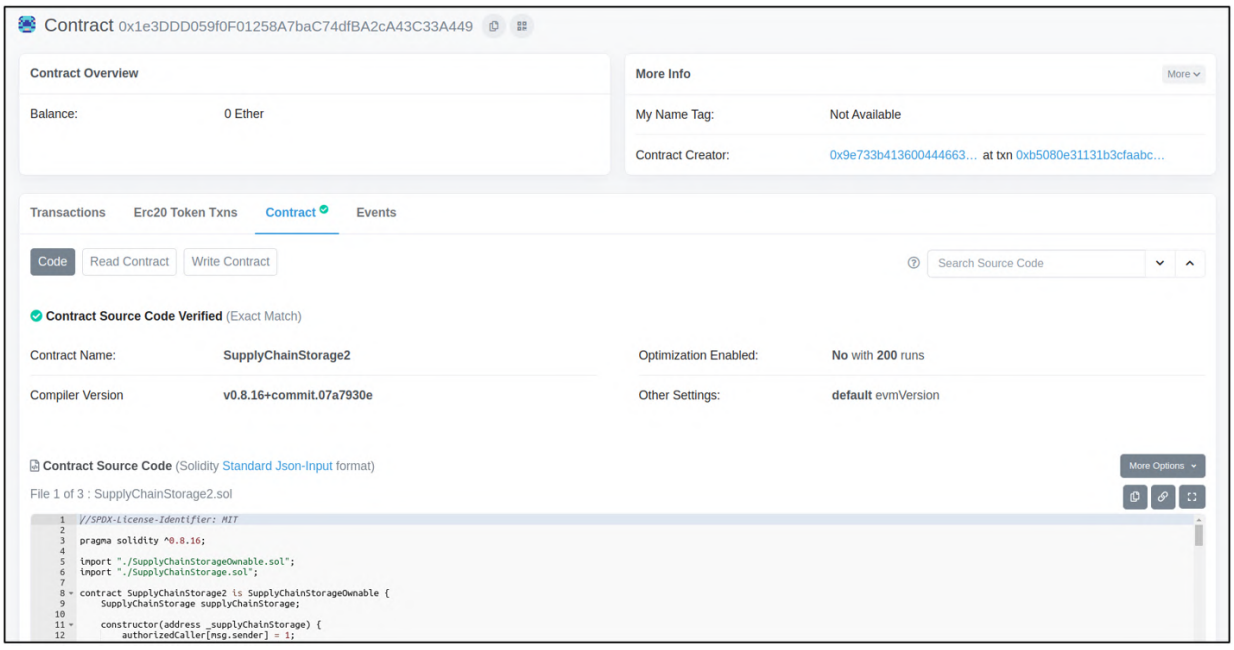


Figura 5.65. Código fuente en Solidity del contrato inteligente SupplyChainStorage2.sol mostrado en Etherscan tras su verificación

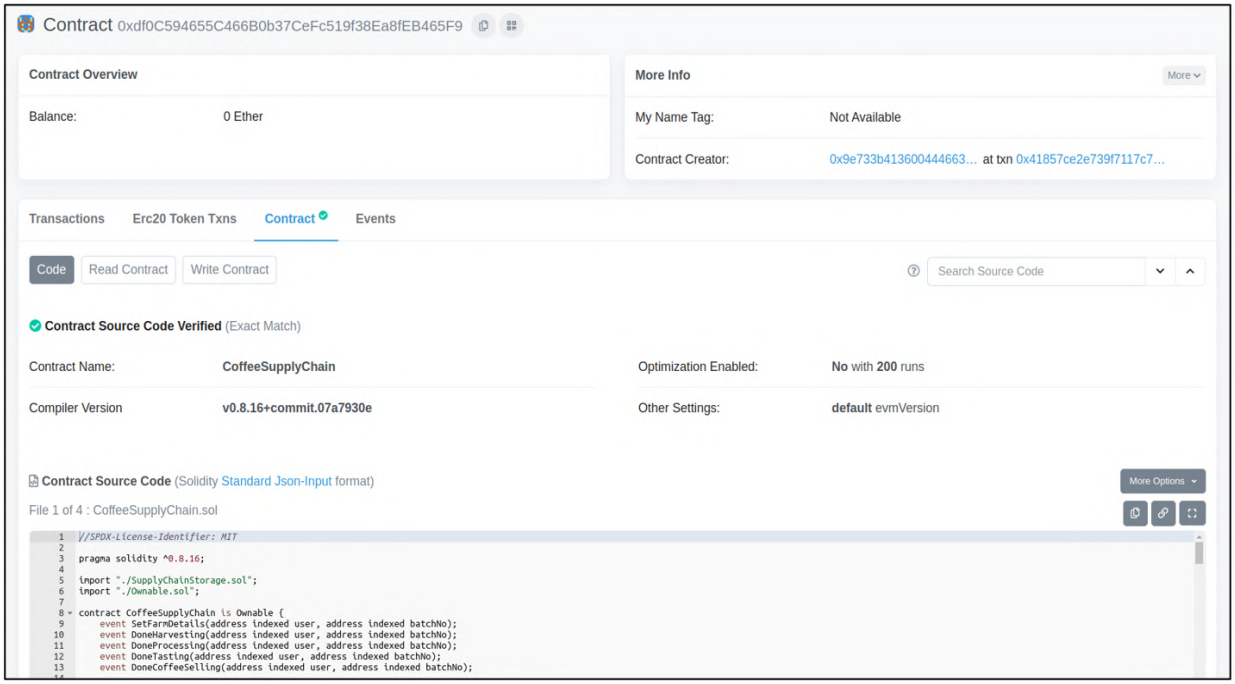


Figura 5.66. Código fuente en Solidity del contrato inteligente CoffeeSupplyChain.sol mostrado en Etherscan tras su verificación

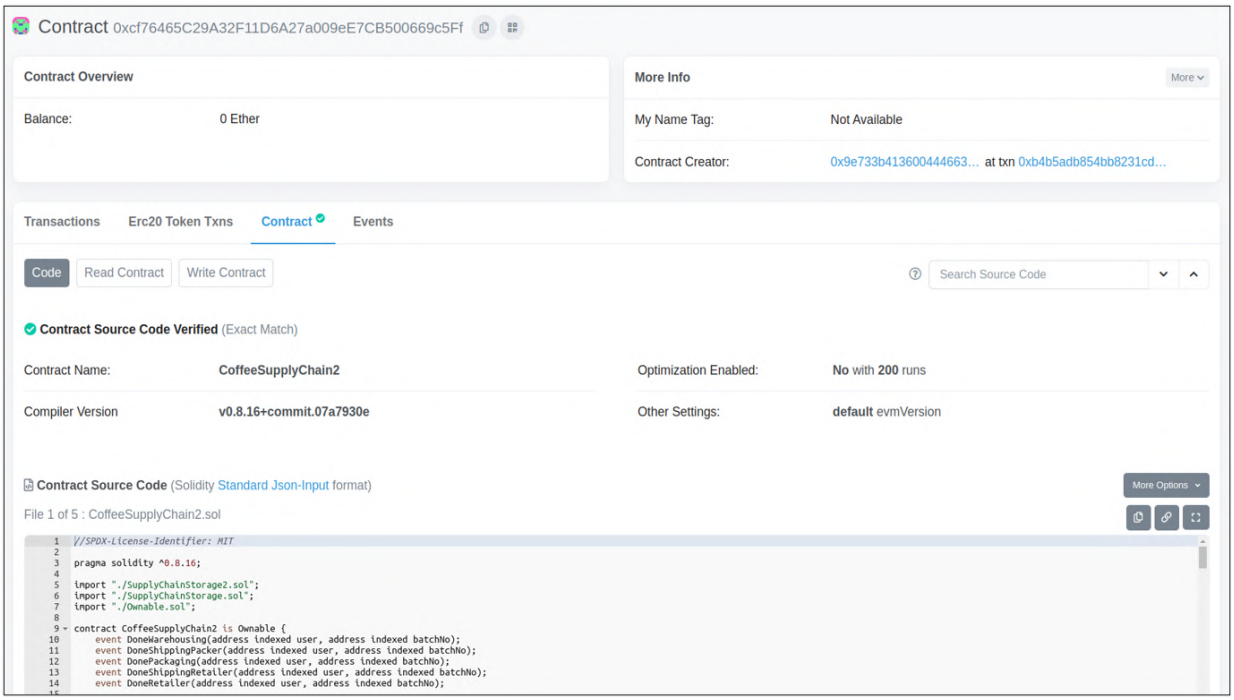


Figura 5.67. Código fuente en Solidity del contrato inteligente CoffeeSupplyChain2.sol mostrado en Etherscan tras su verificación

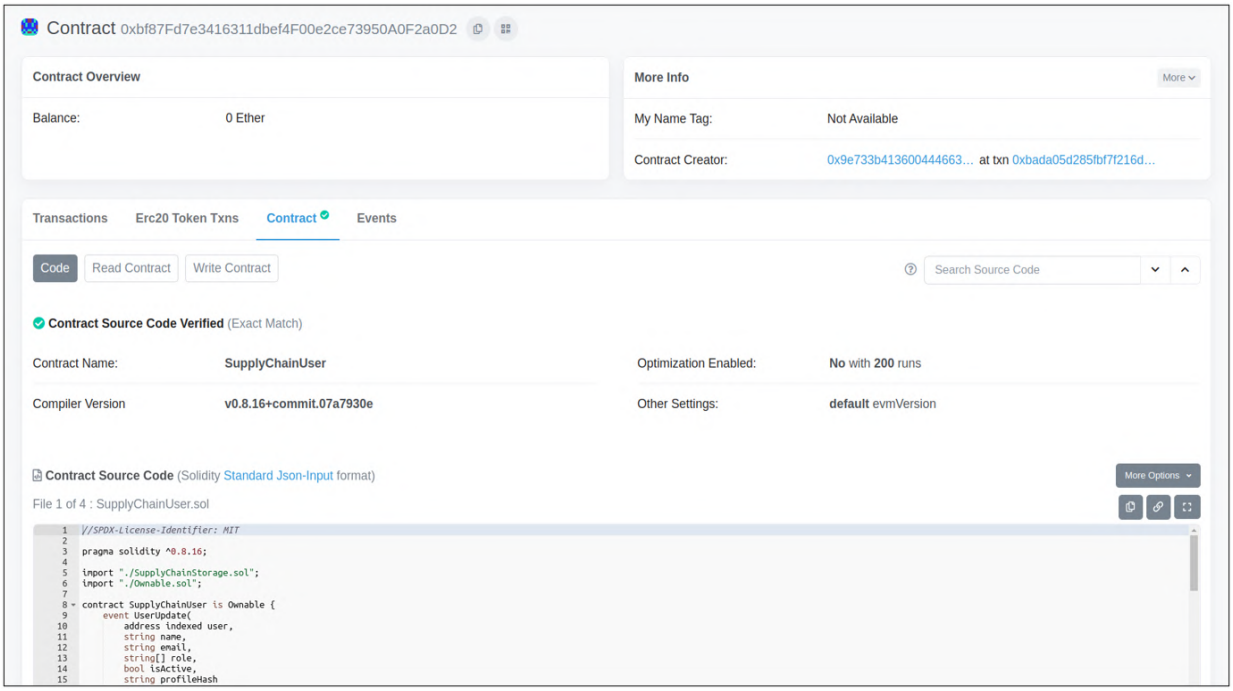


Figura 5.68. Código fuente en Solidity del contrato inteligente SupplyChainUser.sol mostrado en Etherscan tras su verificación

ANEXO XXIX

Diagrama de interacción entre la cadena de bloques Rinkeby y el navegador web por medio de Ethers.js.

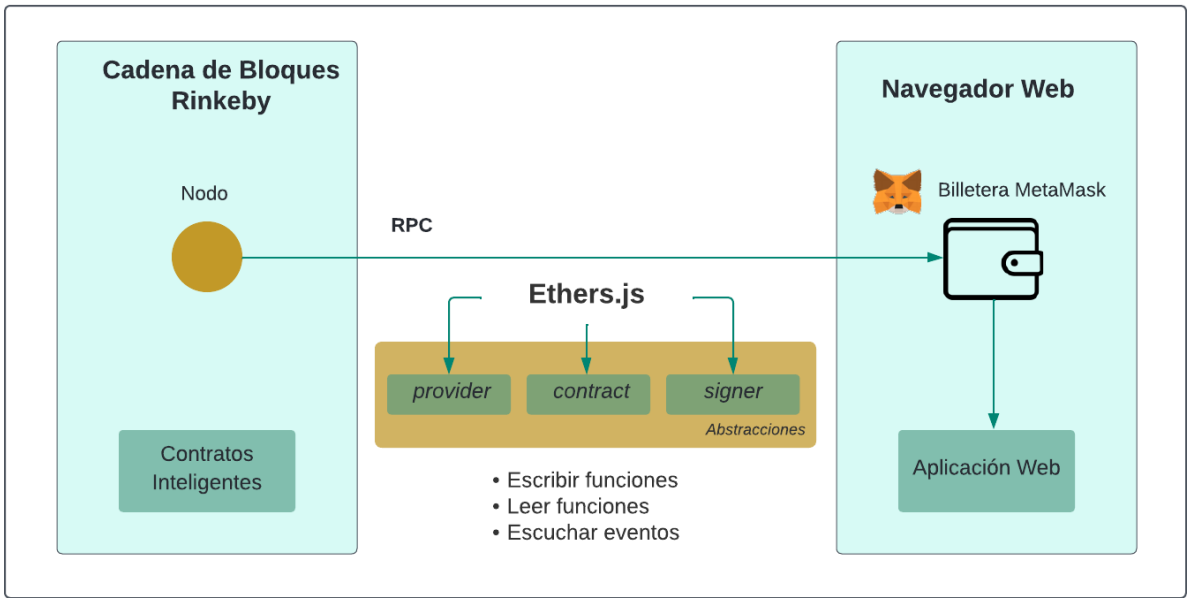


Figura 5.69. Interacción entre Ether.js, el navegador y la blockchain

Ethers.js puede definirse como la interfaz que conecta una cadena de bloques, en este caso Rinkeby, la cual aloja los contratos inteligentes desplegados (SupplyChainStorage.sol, SupplyChainStorage2.sol, CoffeeSupplyChain.sol, CoffeeSupplyChain2.sol y SupplyChainUser.sol), con un navegador web que aloja una aplicación descentralizada (Dapp) la cual requiere de la billetera digital MetaMask para iniciar sesión en la aplicación.

La billetera MetaMask internamente usa el proveedor de nodos de Infura, esto le permite conectarse a un nodo de la cadena de bloques Rinkeby por medio de una llamada de procedimiento remoto (RPC).

Ethers.js crea abstracciones de tipo *provider*, *signer* y *contract* para brindar funcionalidades de conexión con la blockchain, firma de transacciones por parte de la dirección de MetaMask de un usuario e interacción a contratos inteligentes en la blockchain. De esta manera, se pueden escribir y leer funciones de contratos inteligentes desplegados, así como escuchar eventos producidos por estos contratos en la blockchain.

ANEXO XXX

Script del código para la escucha de eventos nuevos en la blockchain con el método *.on()*.

```
import { useState, useEffect } from 'react';
import { coffeeSupplyChainReader } from '../erc20';

const PackerListener = () => {
  const [packRegistered, setPackRegistered] = useState({});

  useEffect(() => {
    if (typeof window.ethereum !== 'undefined') {
      const coffee1Contract = coffeeSupplyChainReader();
      coffee1Contract.on('DonePackaging', (user, batchNo, event) => {
        setPackRegistered({
          user,
          batchNo,
          tx: event.transactionHash,
        });
      });
      return () => {
        erc20.removeAllListeners('DonePackaging');
      };
    }
  }, []);

  return { packRegistered };
};

export default PackerListener;
```

Figura 5.70. Script del código para escuchar eventos nuevos en la blockchain

ANEXO XXXI

Script del código para el filtrado de eventos ya emitidos en la blockchain con el método *.queryFilter()*.

```
import React, { useState, useEffect } from 'react';
import { useSelector } from 'react-redux';
import { walletAddressSelector } from '../../redux/appDataSlice';
import {  coffeeSupplyChainReader } from '../../logic/erc20';
import AskNextAction from '../../logic/GetNextAction/AskNextAction';


const PackerView = () => {
  const [batchNo, setBatchNo] = useState([]);
  const [nextActions, setNextActions] = useState([]);
  const walletAddress = useSelector(walletAddressSelector);

  useEffect(() => {
    const getBatch = async () => {
      const coffee1Contract = coffeeSupplyChainReader();
      const events = await coffee1Contract.queryFilter(coffee1Contract.filters.DonePackaging(walletAddress, null));
      const batchTemp = events.map((event) => event.args.batchNo);
      const nextActionsTemp = batchTemp.map(async (item) => {
        const res = await AskNextAction({ batchNo: item });
        return res.data;
      });

      setBatchNo(batchTemp);
      setNextActions(await Promise.all(nextActionsTemp));
    };
    getBatch();
  }, []);

};

export default PackerView;
```

Figura 5.71. Script del código del filtrado de eventos ya emitidos en la blockchain

ANEXO XXXII

Configuración de un nodo de IPFS con Infura.

Para llevar a cabo la conexión a un nodo de IPFS por medio de Infura, se requiere crear un proyecto IPFS en la página de Infura ingresando la información de una tarjeta de crédito ya que el servicio es gratuito hasta los 5GB de almacenamiento, a partir de este valor en adelante, el servicio de Infura para la red IPFS tiene un costo. Para crear un proyecto IPFS en Infura se realiza lo siguiente:

1. Registrarse en la página web de https://infura.io/.
2. Seleccionar el botón de CREATE NEW KEY y en la ventana emergente, elegir IPFS como tipo de red en el campo NETWORK y colocar un nombre al proyecto en el campo NAME.

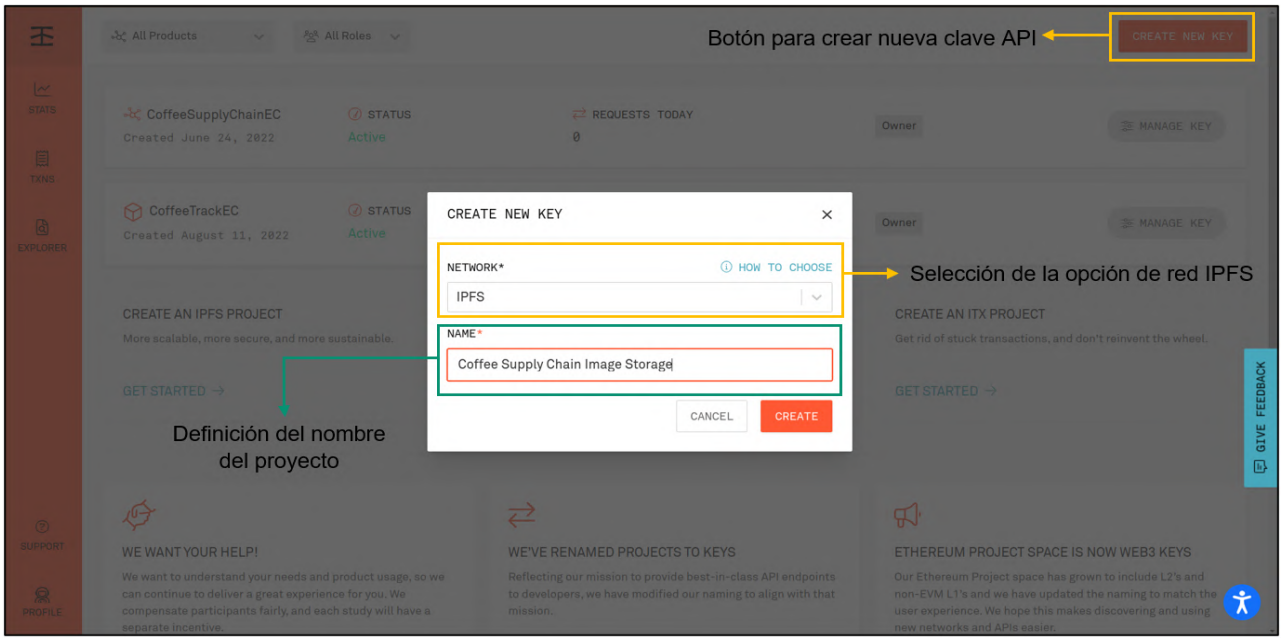


Figura 5.72. Selección del botón para crear una clave nueva y configuración del proyecto

3. A continuación, se reenviará a la página donde se incluye la información de pago. Luego, se selecciona el botón CREATE PROJECT.

Figura 5.73. Seteo de la información sobre el método de pago a utilizar

4. Habilitar el botón ENABLED del campo ENABLE DEDICATED GATEWAYS y elegir el nombre del subdominio personalizado único del campo UNIQUE SUBDOMAIN NAME, el cual es empleado para traer la información desde los nodos de la red IPFS. El proyecto creado con la información con respecto a la clave API, el ID del proyecto y el gateway dedicado habilitado, se muestran en el panel de control, estos datos son los requeridos para el uso de IPFS en la aplicación web.

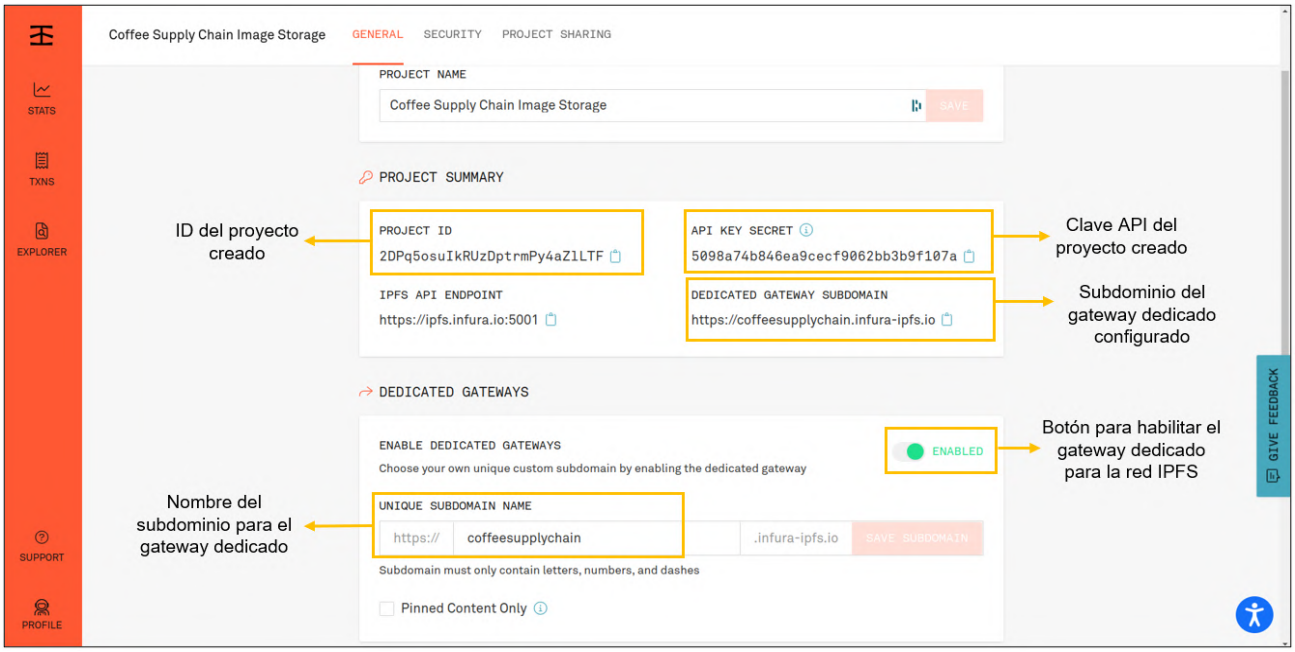


Figura 5.74. Información sobre la clave, el ID y el gateway configurado del proyecto creado visualizada en el panel de control

ANEXO XXXIII

Script del código de la conexión con el servicio de IPFS para almacenar archivos tipo imagen.

```
import { create } from 'ipfs-http-client';
import { Buffer } from 'buffer';

const projectId = process.env.REACT_APP_IPFS_PROJECT_ID;
const projectSecret = process.env.REACT_APP_IPFS_KEY;
const auth = `Basic ${Buffer.from(`${projectId}:${projectSecret}`).toString('base64')}`;

// Conexión a la red de IPFS
export const createIpfs = () => {
  return create({
    host: 'ipfs.infura.io',
    port: 5001,
    protocol: 'https',
    headers: {
      authorization: auth,
    },
  });
};

export const addFileToIpfs = async (ipfs, file) => {
  if (!ipfs) return { url: null, error: 'IPFS error' };
  try {
    // Envío del archivo a la red de IPFS
    const result = await ipfs.add(file);
    // Obtención del archivo almacenado en IPFS por medio de su CID
    return { url: `https://coffeetrack.infura-ipfs.io/ipfs/${result.path}`, error: null };
  } catch (error) {
    return { url: null, error: error.message };
  }
};
```

Figura 5.75. Script del código para usar IPFS en el almacenamiento de imágenes

ANEXO XXXIV

Script del código para el uso de las librerías *qrcode.react* para generar un código QR y *react-qr-reader* para leer un código QR.

Los scripts mostrados a continuación muestran únicamente la generación y lectura del código QR en la aplicación desarrollada, es decir que estos pertenecen a secciones más amplias de código que no se muestran en la captura de pantalla pero que pueden ser consultadas a través del repositorio de GitHub del proyecto https://github.com/NathaliaBarreiros/coffee-supply-chain-client.

```
import QRCode from 'qrcode.react';

const TableUsers = ({ batchNo, nextActions }) => {
  return (
    <Grid item xs={12} sx={{ marginTop: '50px' }}>
        {/* Generación del código QR */}
        <QRCode
        bgColor="#FFFFFF"
        id={batch}
        value={`${process.env.REACT_APP_HOST_URL}/tracking?batch=${batch}`}
        size="50"
        includeMargin
        renderAs="svg"
        />
    </Grid>
  );
};

export default TableUsers;
```

Figura 5.76. Script del código para generar un código QR

```
import React, { useState, useRef, useEffect } from 'react';
import { QrReader } from 'react-qr-reader';

const Section1 = () => {
  const [stateLectorQR, setStateLectorQR] = useState(false);
  const [dataQR, setDataQR] = useState('');
  return (
    <div id="section-1 mb-20">
        <QrReader
        onResult={(result, error) => {
            if (result) {
            setDataQR(result?.text);
            setStateLectorQR(true);
            }
        }}
        containerStyle={{
            marginTop: '0px',
            paddingTop: '0px',
            width: '450px',
        }}
        />
    </div>
  );
};

export default Section1;
```

Figura 5.77. Script del código para leer un código QR

ANEXO XXXV

Pasos para crear un clave API en el servicio de Google Maps API.

Una vez creada una cuenta en Google Maps API, se realiza lo siguiente:

1. En el panel de control, seleccionar la opción de Credenciales.

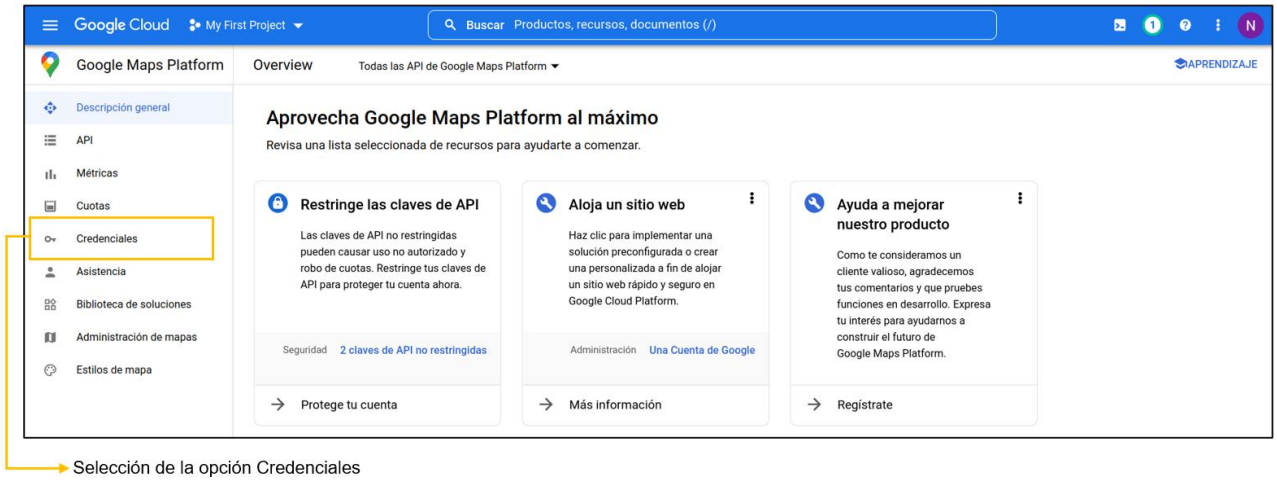


Figura 5.78. Paso 1: Selección de la opción Credenciales

2. En la nueva ventana cargada, seleccionar el botón CREATE CREDENTIALS y elegir la opción Clave de API.

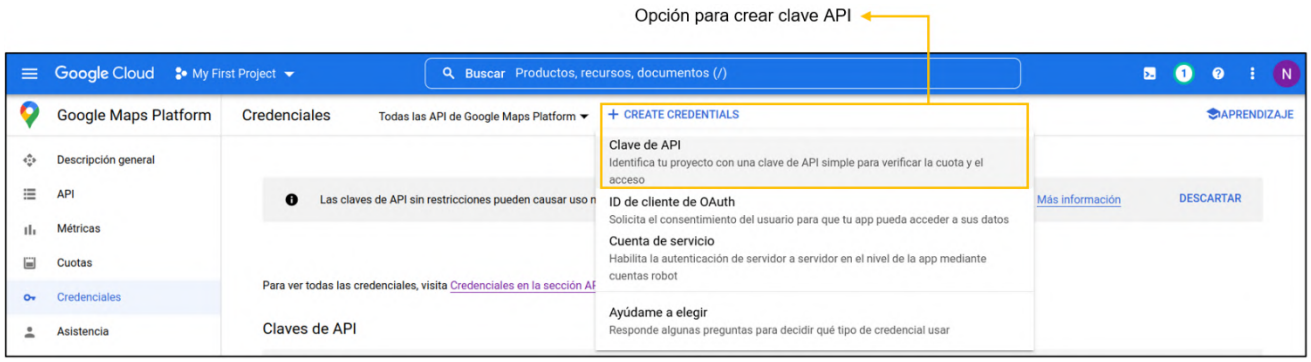


Figura 5.79. Paso 2: Selección del botón crear credenciales y de la opción clave API

A continuación, la clave API es creada y mostrada en una ventana emergente. Esta clave debe ser configura en el archivo de variables de entorno del proyecto para ser empleada.

Figura 5.80. Clave API creada

ANEXO XXXVI

Script del código para el uso de las librerías *@react-google-maps/api* para generar y usar un mapa de Google Maps API y *use-places-autocomplete* para autocompletar direcciones en una barra de búsqueda.

Los scripts mostrados a continuación pertenecen a secciones más amplias de código que no se muestran en la captura de pantalla pero que pueden ser consultadas a través del repositorio de GitHub del proyecto https://github.com/NathaliaBarreiros/coffee-supply-chain-client.

```
import React, { useState, useRef, useCallback, useEffect } from 'react';
import { GoogleMap, useLoadScript, Marker, InfoWindow } from '@react-google-maps/api';
import { Geolocation } from '../../logic/Maps/NavLocation';

const libraries = ['places'];
const mapContainerStyle = {
  height: '100%',
  width: '100%',
};
const options = {
  mapId: 'c43de04e5c01e2fc',
  scaleControl: true,
  scrollwheel: true,
  mapTypeControl: false,
  keyboardShortcuts: false,
  zoomControl: true,
};
const center = {
  lat: 0.06218317633464823,
  lng: -78.6820212451912,
};
const MapsLocation = ({ svg }) => {
  const { isLoaded, loadError } = useLoadScript({
    googleMapsApiKey: process.env.REACT_APP_GOOGLE_MAPS_API_KEY,
    libraries,
  });
  const [markers, setMarkers] = useState({});
  const [location, setLocation] = useState('');
  const [selected, setSelected] = useState(null);
  const [selectedLoc, setSelectedLoc] = useState(null);
  const onMapClick = useCallback((e) => {
    setMarkers({
      lat: e.latLng.lat(),
      lng: e.latLng.lng(),
      time: new Date(),
    });
    const locDir = Geolocation(e.latLng.lat(), e.latLng.lng());
    locDir.then((response) => {
      if (response.results && response.results[0]) {
        setLocation(response.results[0].formatted_address);
      }
    });
  }, []);
  const mapRef = useRef();
  const onMapLoad = useCallback((map) => {
    mapRef.current = map;
  }, []);
  const panTo = useCallback(({ lat, lng }) => {
    mapRef.current.panTo({ lat, lng });
    mapRef.current.setZoom(14);
  }, []);
  if (loadError) return 'Error';
  if (!isLoaded) return 'Cargando ...';
  return (
    <>
      <GoogleMap
        id="map"
        mapContainerStyle={mapContainerStyle}
        zoom={9}
        center={center}
        options={options}
        onClick={onMapClick}
        onLoad={onMapLoad}
      >
```

```
        <Search panTo={panTo} setMarker={setMarkers} setLocation={setLocation} />
        <Locate panTo={panTo} setMarker={setMarkers} setLocation={setLocation} />
        <Marker
          key={uuid()}
          position={{ lat: parseFloat(markers.lat), lng: parseFloat(markers.lng) }}
          onClick={() => {
            setSelected(markers);
            setSelectedLoc(location);
          }}
          icon={{
            url: svg,
            origin: new window.google.maps.Point(0, 0),
            anchor: new window.google.maps.Point(10, 10),
            scaledSize: new window.google.maps.Size(40, 40),
          }}
        />
        {selected && selectedLoc ? (
          <InfoWindow
            position={{ lat: selected.lat, lng: selected.lng }}
            onCloseClick={() => {
              setSelected(null);
              setSelectedLoc(null);
            }}
          >
            <Box>
              <Typography variant="subtitle2">{selectedLoc}</Typography>
              <Typography variant="body2">{formatRelative(selected.time, new Date())} </Typography>
            </Box>
          </InfoWindow>
        ) : null}
      </GoogleMap>
    </>
  );
};

export default MapsLocation;
```

Figura 5.81. Script del código para generar y usar un mapa de Google Maps API

```
import usePlacesAutocomplete, { getGeocode, getLatLng } from 'use-places-autocomplete';

function Search({ panTo, setMarker, setLocation }) {
  const {
    ready,
    value,
    suggestions: { status, data },
    setValue,
    clearSuggestions,
  } = usePlacesAutocomplete({
    requestOptions: {
      location: {
        lat: () => -0.18117269544866682,
        lng: () => -78.51212535843558,
      },
      radius: 100 * 1000,
    },
  });
  const handleInput = (e) => {
    setValue(e.target.value);
  };
  const handleSelect = async (address) => {
    setValue(address, false);
    clearSuggestions();
    const results = await getGeocode({ address });
    const { lat, lng } = await getLatLng(results[0]);
    setMarker({ lat, lng, time: new Date() });
    panTo({ lat, lng });
    const locDir = Geolocation(lat, lng);
    locDir.then((response) => {
      if (response.results[0]) {
        setLocation(response.results[0].formatted_address);
      }
    });
  };
  return (
    <Box className="search">
      <Combobox onSelect={handleSelect}>
        <ComboboxInput value={value} onChange={handleInput} disabled={!ready} placeholder="Ingresa la dirección" />
        <ComboboxPopover className="comboPopover" style={{ zIndex: 9999999999 }}>
          <ComboboxList>
            {status === 'OK' && data.map(({ id, description }) => <ComboboxOption key={uuid()} value={description} />)}
          </ComboboxList>
        </ComboboxPopover>
      </Combobox>
    </Box>
  );
}

export default Search;
```

Figura 5.82. Script del código para autocompletar direcciones en una barra de búsqueda

ANEXO XXXVII

Diagrama de la organización visual de la interfaz web desarrollada.

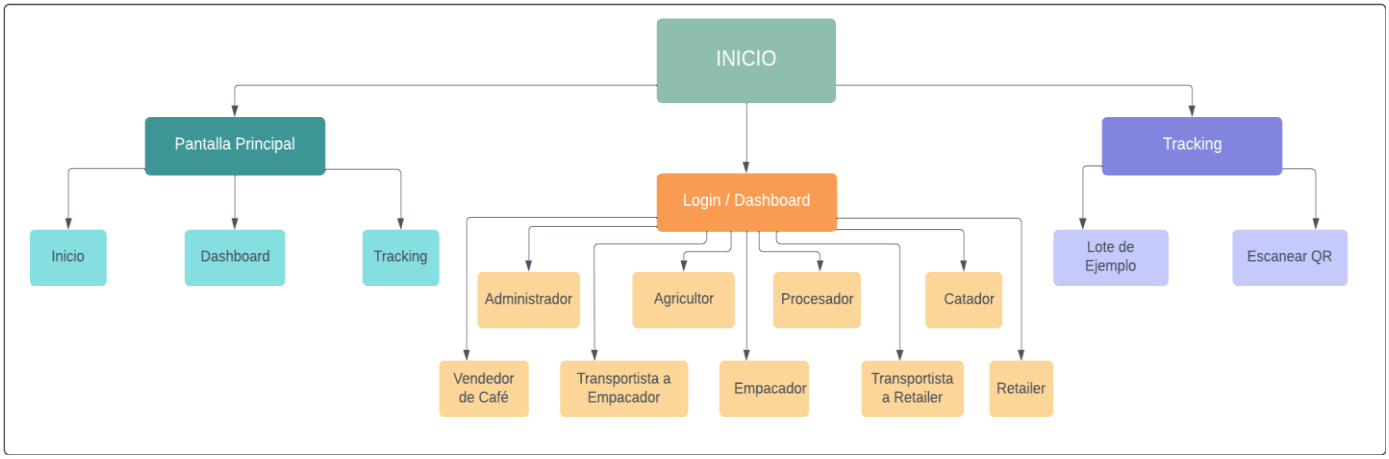


Figura 5.83. Diagrama de la organización visual de la aplicación web

ANEXO XXXVIII

Diagramas de flujo de los procesos de uso de la aplicación web.

1. Perfil del Administrador:

El administrador puede llevar a cabo los procesos de agregar usuario y registro de granja para creación de un nuevo lote de café.

Figura 5.84. Flujo de proceso para agregar un nuevo usuario al sistema

Figura 5.85. Flujo de proceso para registrar una granja de café y crear un nuevo lote

2. Perfil de los Usuarios Modificadores:

Los usuarios modificadores pueden llevar a cabo la actualización de información de su perfil, agregar su intervención ya sea con el ID de lote, con el código QR de redirección o con una URL de redirección. Para esto previamente deben iniciar sesión con su cuenta de billetera MetaMask. El proceso de inicio de sesión también aplica para el administrador.

A continuación, se mostrarán únicamente los flujos de proceso enfocados en el usuario Agricultor con su intervención de cosecha. Para los demás usuarios modificadores y el administrador, en el caso del inicio de sesión, los diagramas son similares.

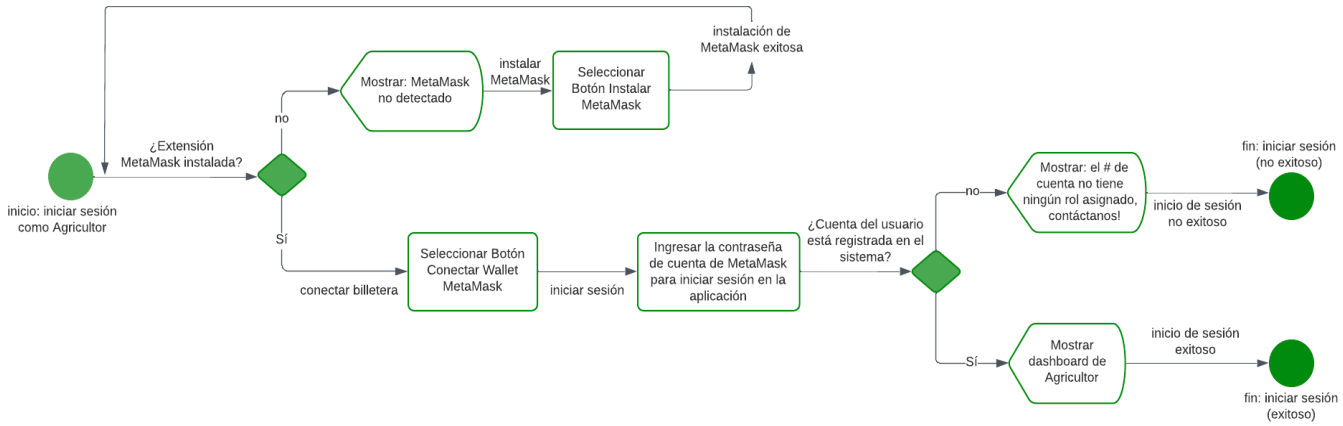


Figura 5.86. Flujo de proceso para el inicio de sesión de un usuario del sistema

Figura 5.87. Flujo de proceso para actualizar perfil por parte de un usuario modificador

Figura 5.88. Flujo de proceso para agregar intervención con el ID del lote de café por parte de un usuario modificador

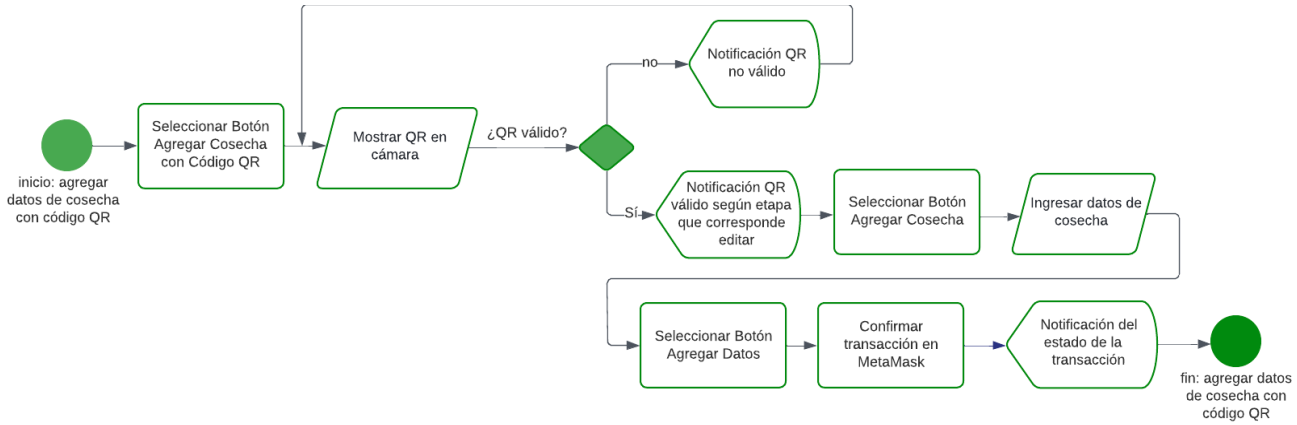


Figura 5.89. Flujo de proceso para agregar intervención con el código QR del lote de café por parte de un usuario modificador

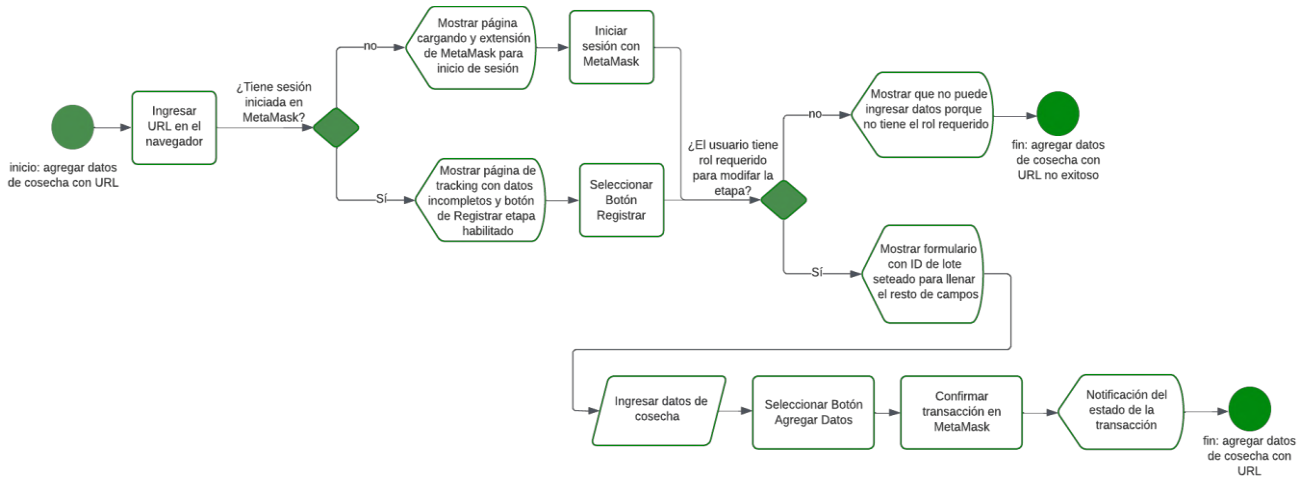


Figura 5.90. Flujo de proceso para agregar intervención con la URL de redirección del lote de café por parte de un usuario modificador

3. Perfil de los Usuarios Modificadores y del Administrador:

Tanto los usuarios modificadores como el administrador pueden ver el tracking o historial de un lote de café específico mostrado en la tabla de su panel de control, esto les redirige a la página de Tracking del lote seleccionado. Por otro lado, también pueden compartir el ID del lote.

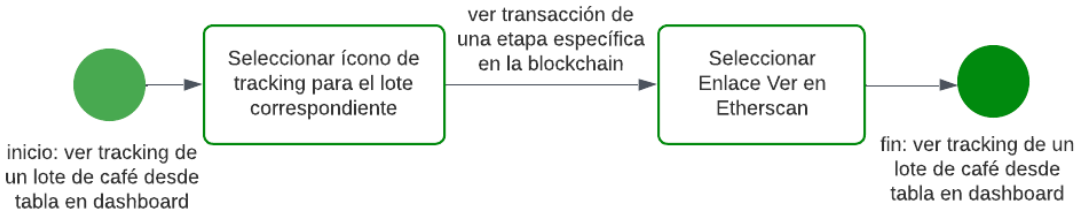


Figura 5.91. Flujo de proceso para ver el tracking de un lote de café específico desde la tabla desplegada en el panel de control de los usuarios modificadores y del administrador

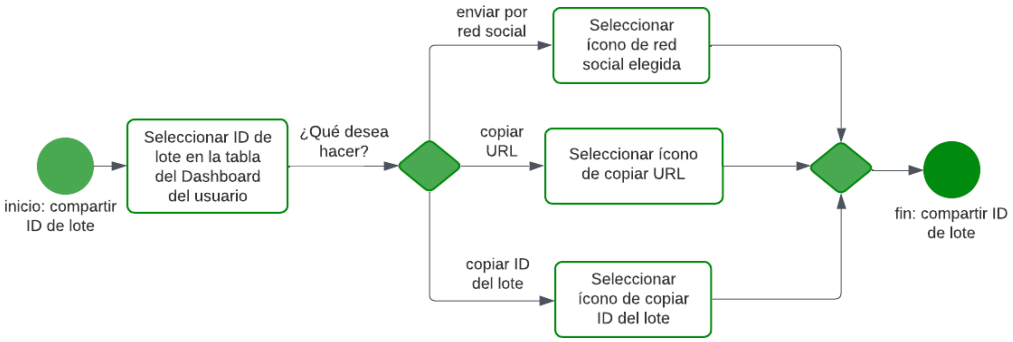


Figura 5.92. Flujo de proceso para compartir el ID de un lote de café específico desde la tabla desplegada en el panel de control de los usuarios modificadores y del administrador

4. Usuario Consumidor:

Los usuarios consumidores no tienen un panel de control, sin embargo, estos pueden usar la aplicación web para dirigirse a la ventana de Tracking y escanear el código QR de su funda de café. También pueden usar alguna aplicación de escaneo de códigos QR, esta les mostrará la URL a la cual redirigirse para consultar el historial sobre su café.

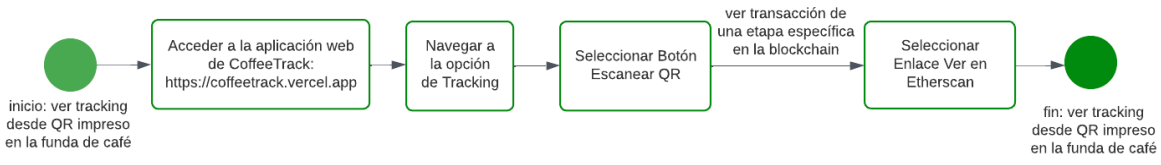


Figura 5.93. Flujo de proceso para ver el historial de tracking desde un código QR impreso en la funda de café

ANEXO XXXIX

Proceso para alojar el proyecto en el servicio de Vercel.

Se siguen los siguientes pasos:

1. Iniciar sesión en la página de Vercel https://vercel.com/ utilizando la cuenta de GitHub que contiene el repositorio de código creado, seleccionando Continue with GitHub.

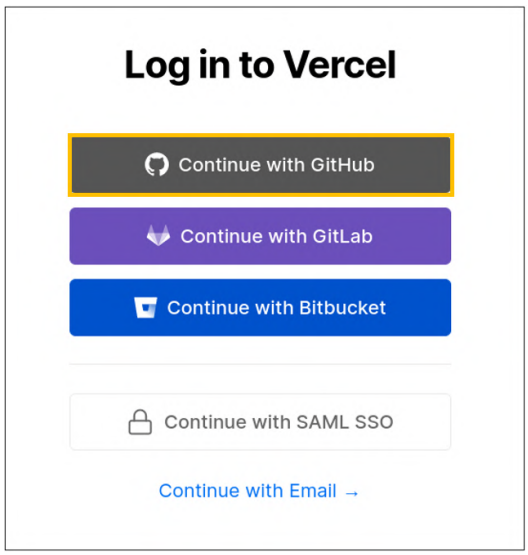


Figura 5.94. Paso 1: Ventana de inicio de sesión con una cuenta de GitHub

2. En el panel de control mostrado, elegir el botón New Project.

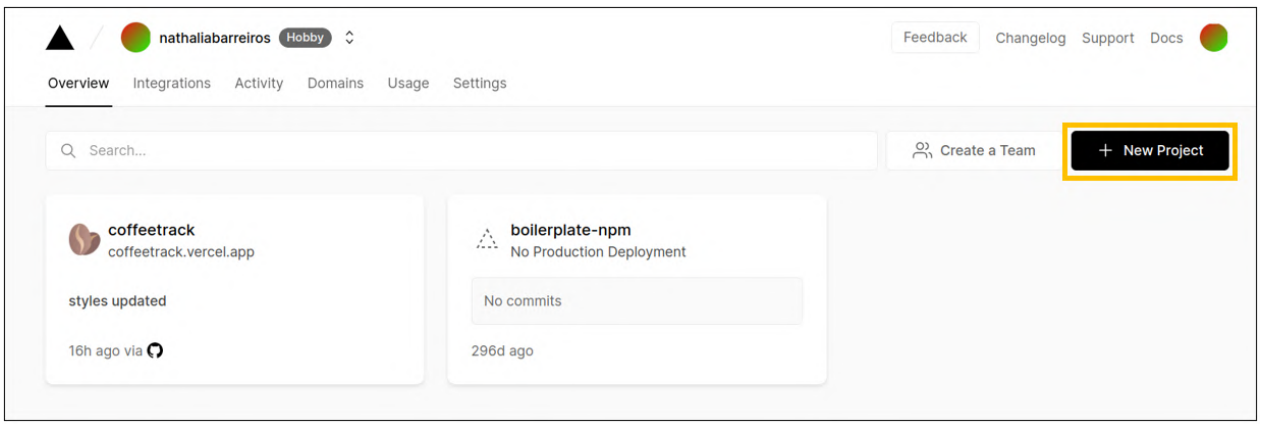


Figura 5.95. Paso 2: Selección del botón para crear un nuevo proyecto

3. En la ventana mostrada, seleccionar el repositorio del código a implementar con el botón Import.

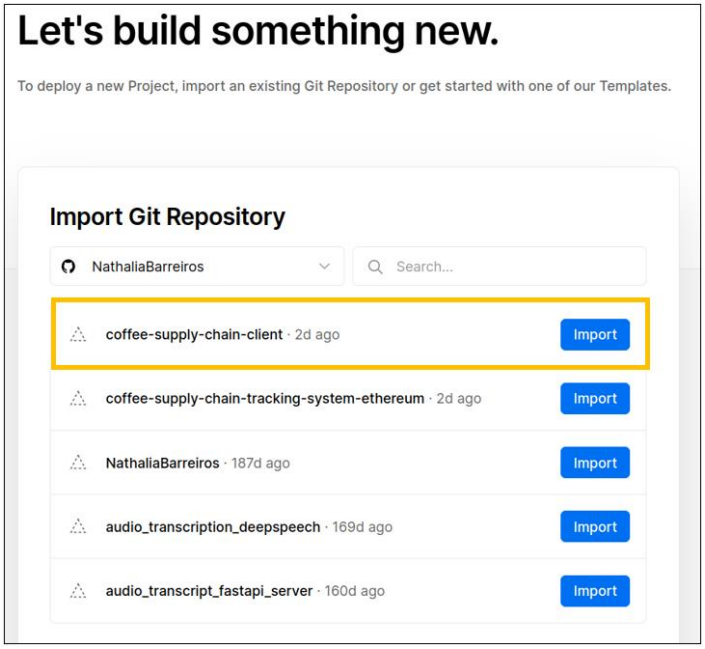


Figura 5.96. Paso 3: Selección del repositorio del código a implementar

4. Configurar el nombre que mostrará el dominio del proyecto, el framework utilizado de Create React App, elegir el directorio donde se encuentra el archivo raíz del proyecto y setear las variables de ambiente correspondientes a las claves API de los servicios utilizados en el proyecto. Seleccionar el botón Deploy para el despliegue.

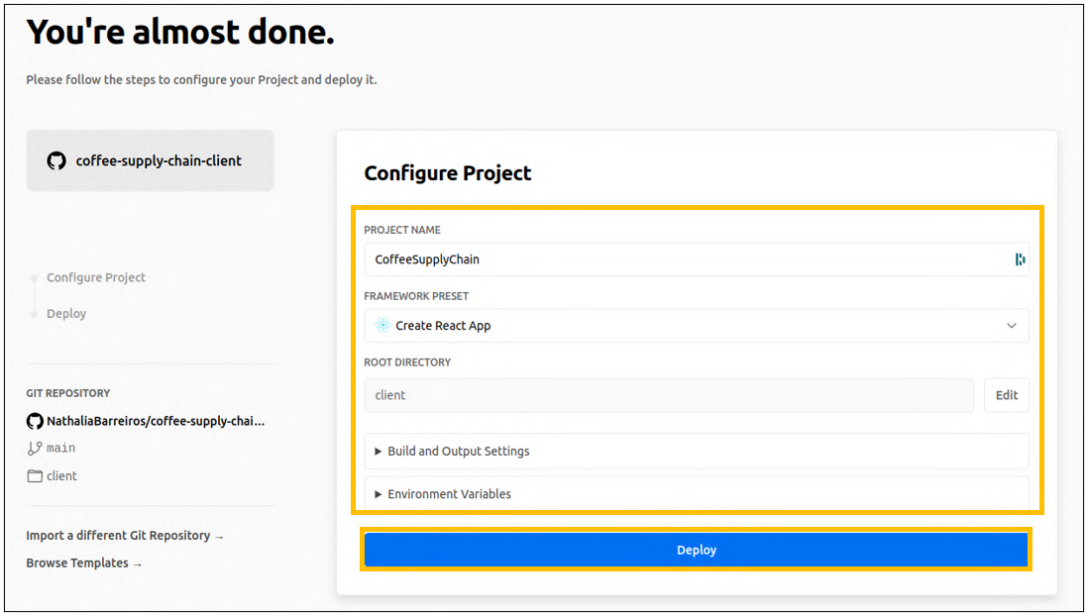


Figura 5.97. Paso 4: Configuración del proyecto y selección del botón de despliegue

Una vez configurado el proyecto, Vercel empieza la construcción de la implementación del proyecto.

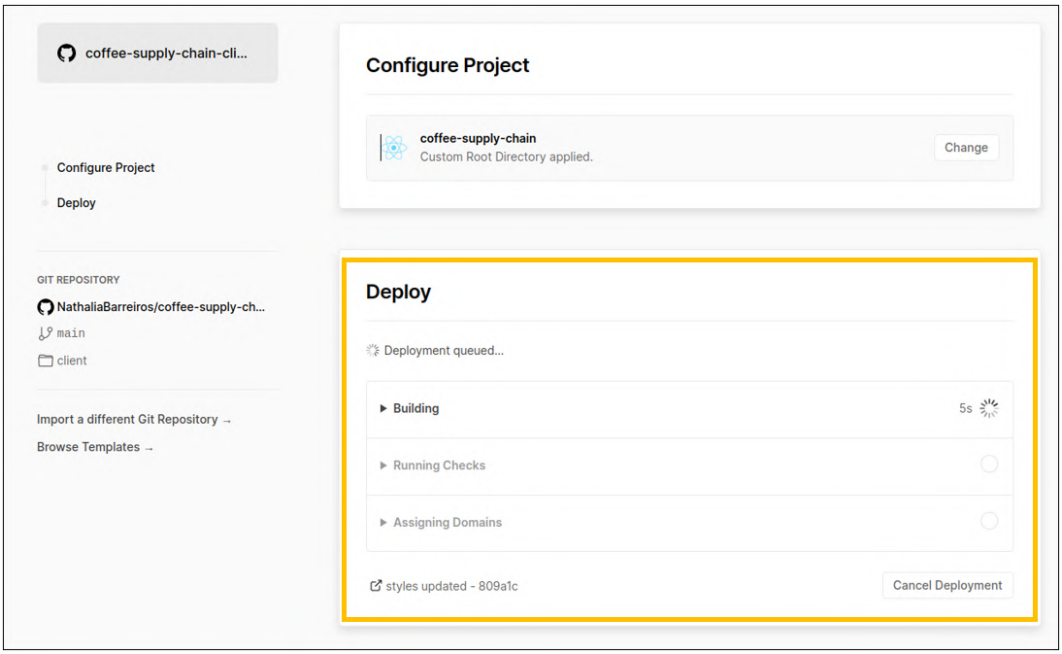


Figura 5.98. Construcción de la implementación del proyecto

5. Cuando el proceso de implementación haya terminado, se muestra una nueva ventana. Seleccionar el botón Go to Dashboard para ir al panel de control del proyecto implementado.

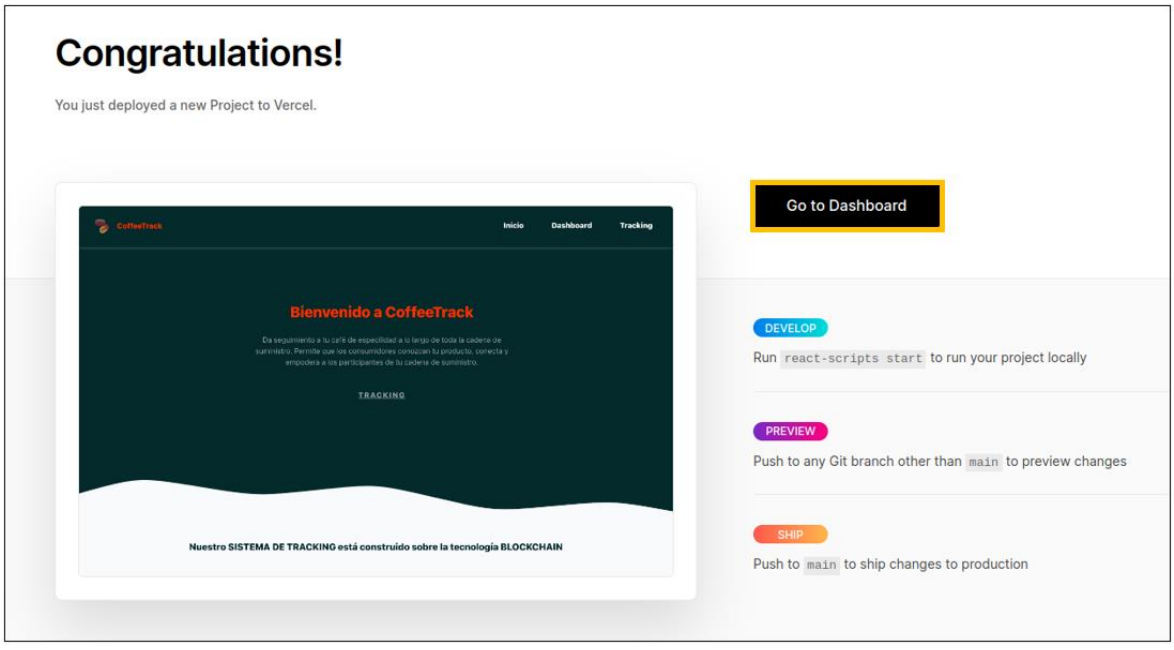


Figura 5.99. Paso 5: Selección del botón Go to Dashboard

6. En el panel de control, seleccionar el botón Visit que redirige a la URL del proyecto implementado.

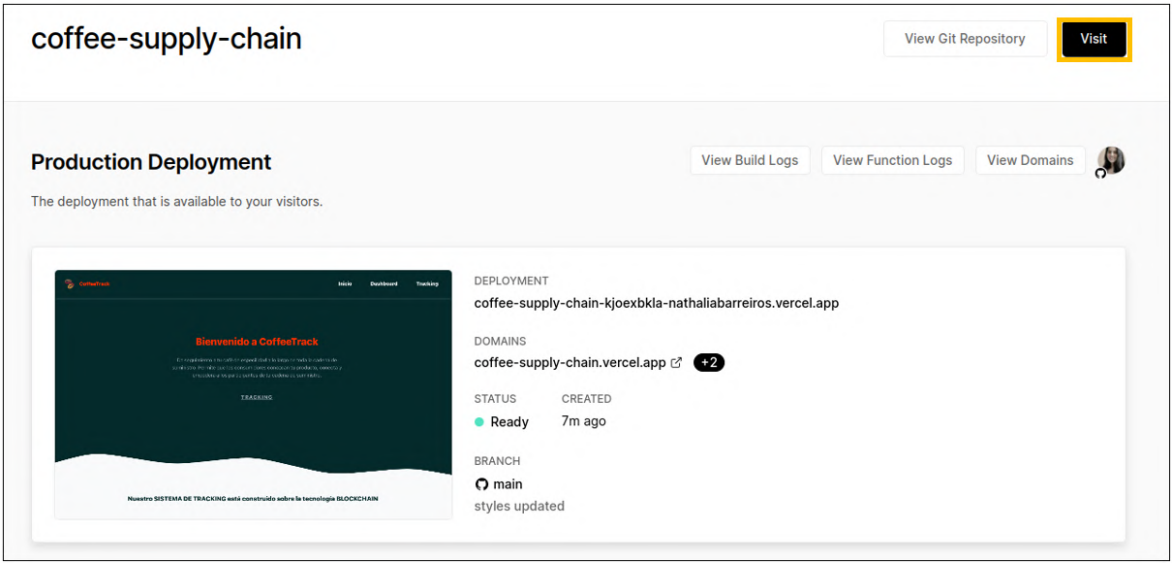


Figura 5.100. Paso 6: Selección del botón Visit para redirección a la URL del proyecto

ANEXO XL

Manual de usuario: instalación del sistema y uso de la aplicación web para interactuar con el sistema de tracking.

- **INSTALACIÓN DE LA BILLETERA METAMASK EN EL NAVEGADOR WEB Y CREACIÓN DE CUENTA EN METAMASK**

La instalación de la extensión de MetaMask se realizará en el navegador Chrome, sin embargo, el procedimiento es el mismo para cualquier navegador.

1. Acceder a la URL de MetaMask https://metamask.io/ y seleccionar la opción Download.

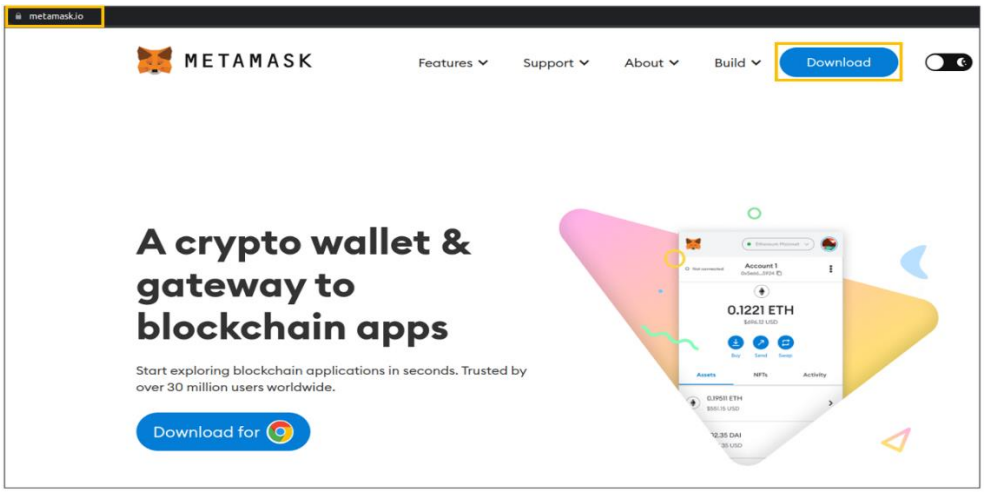


Figura 5.101. Paso 1. Seleccionar opción descargar en https://metamask.io/

2. En la página de redirección seleccionar Install MetaMask for Chrome. El nombre de este botón varía en función del navegador que se use.

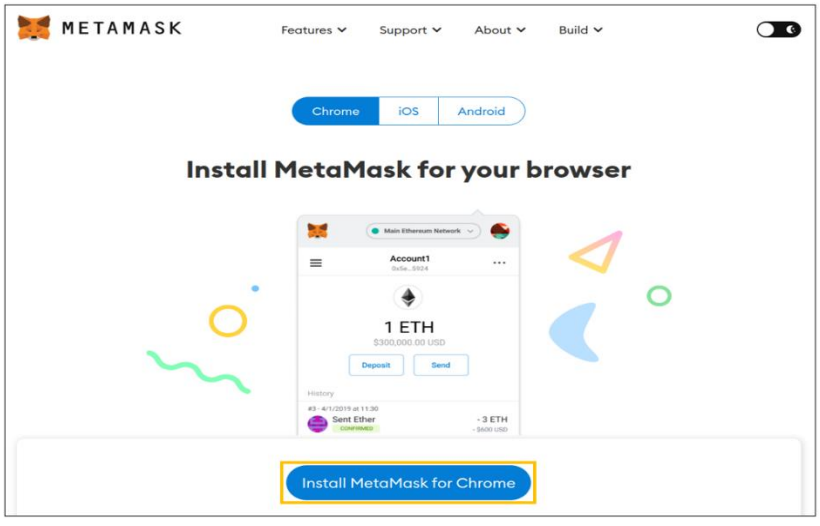


Figura 5.102. Paso 2. Seleccionar Install MetaMask for Chrome

3. En la página de redirección correspondiente a las extensiones del navegador, seleccionar Agregar a Chrome y en la ventana emergente seleccionar Agregar extensión.

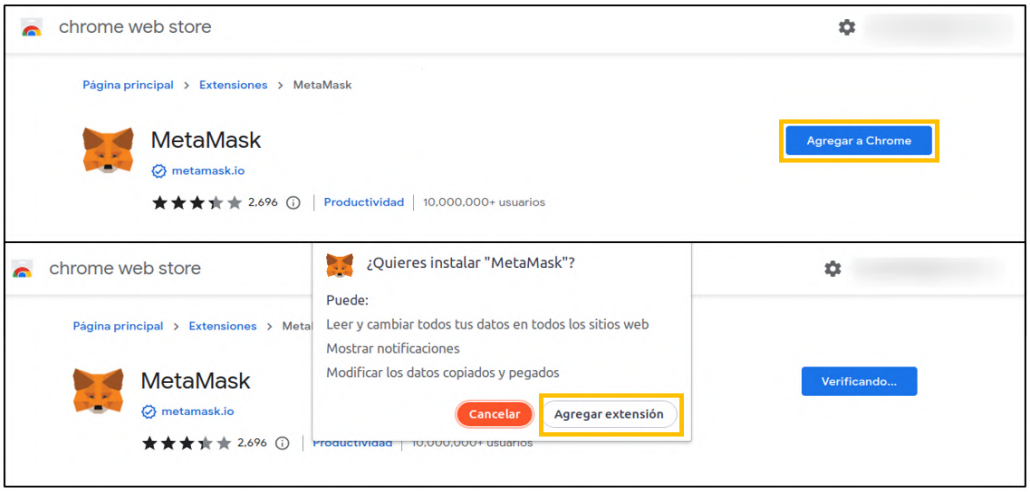


Figura 5.103. Paso 3. Seleccionar Agregar a Chrome en la página de extensiones del navegador

4. Seleccionar Empezar en la página de redirección. En la siguiente página, seleccionar la opción deseada en función de si se desea aceptar la política de privacidad de MetaMask.

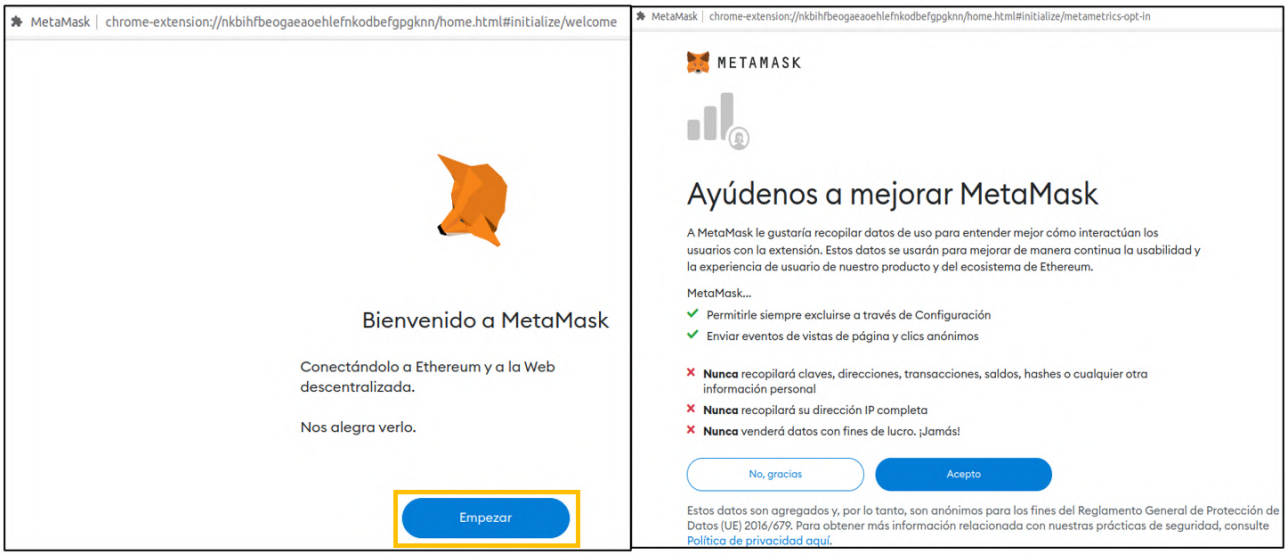


Figura 5.104. Paso 4. Empezar la configuración de cuenta en MetaMask

5. Para crear una cuenta en MetaMask, seleccionar Crear una cartera, en la página mostrada.

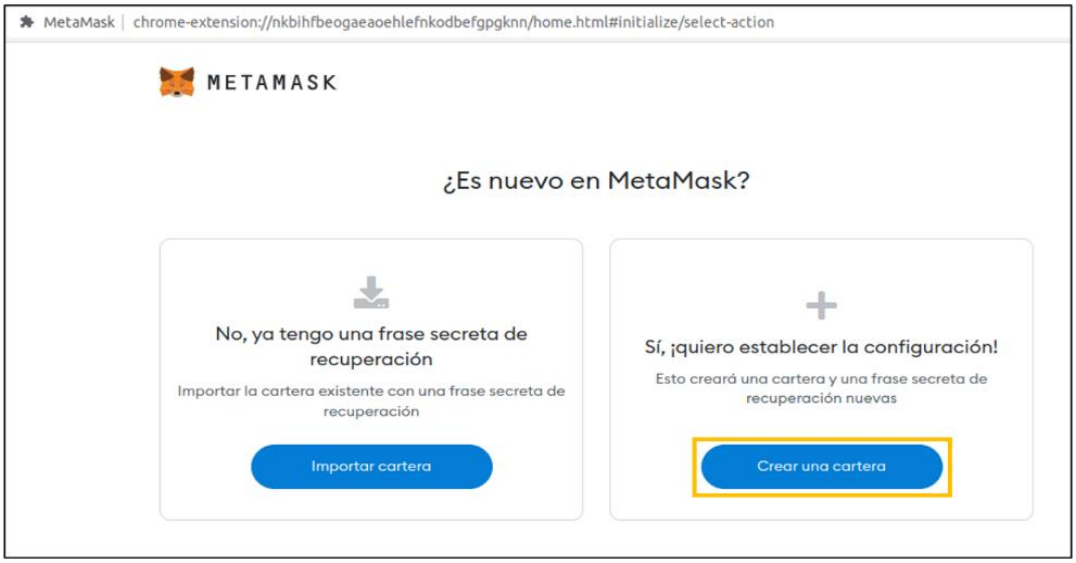


Figura 5.105. Paso 5. Seleccionar Crear una cartera

6. En la siguiente página, llenar los campos de contraseña y confirmación de contraseña, aceptar los términos de uso y luego seleccionar el botón de Crear.

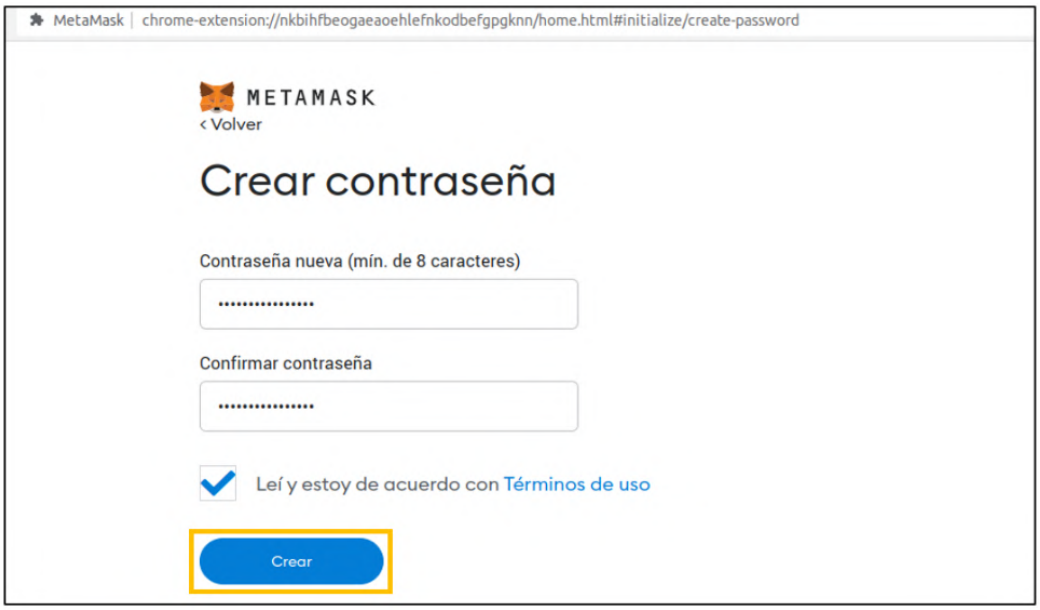


Figura 5.106. Paso 6. Definición de contraseña de cuenta MetaMask

7. Leer la información sobre la frase secreta de recuperación o mnemónico de la cartera en MetaMask y su importancia en la protección de la cuenta a crear y luego seleccionar el botón de Siguiente.

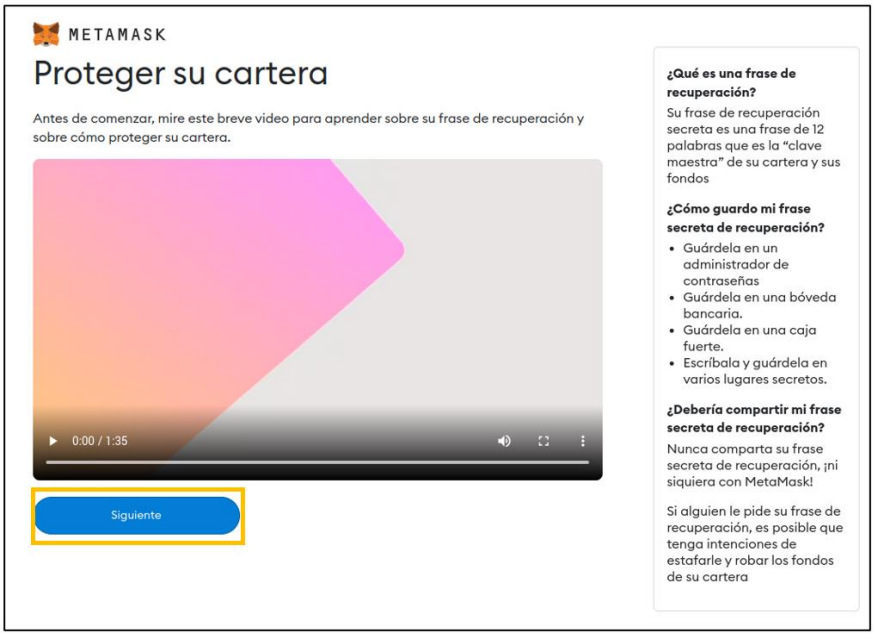


Figura 5.107. Paso 7. Leer información sobre el mnemónico en MetaMask

8. En la siguiente página, se muestra el mnemónico de la cuenta, copiar esta información en un lugar seguro y asegurarse de no compartirla con nadie, luego seleccionar Siguiente. Cabe mencionar que esta frase permite la creación de una copia de seguridad de la cuenta y su restauración, por lo que cualquier persona que tenga esta información, puede acceder a las criptomonedas que se tengan almacenadas en la billetera y tranquilamente podrían robar los fondos de esta y realizar transacciones con el número de cuenta de la billetera. También se recomienda que se acceda a la billetera únicamente a través de dispositivos personales.

Figura 5.108. Paso 8. Mostrar mnemónico generado para la cuenta MetaMask

9. En la siguiente página, confirmar la frase secreta de respaldo anteriormente mostrada y seleccionar Confirmar.

Figura 5.109. Paso 9. Confirmación de la frase secreta de recuperación de la cuenta

10. Finalmente, seleccionar Todo listo para abrir la extensión de MetaMask y mostrar la cuenta creada.

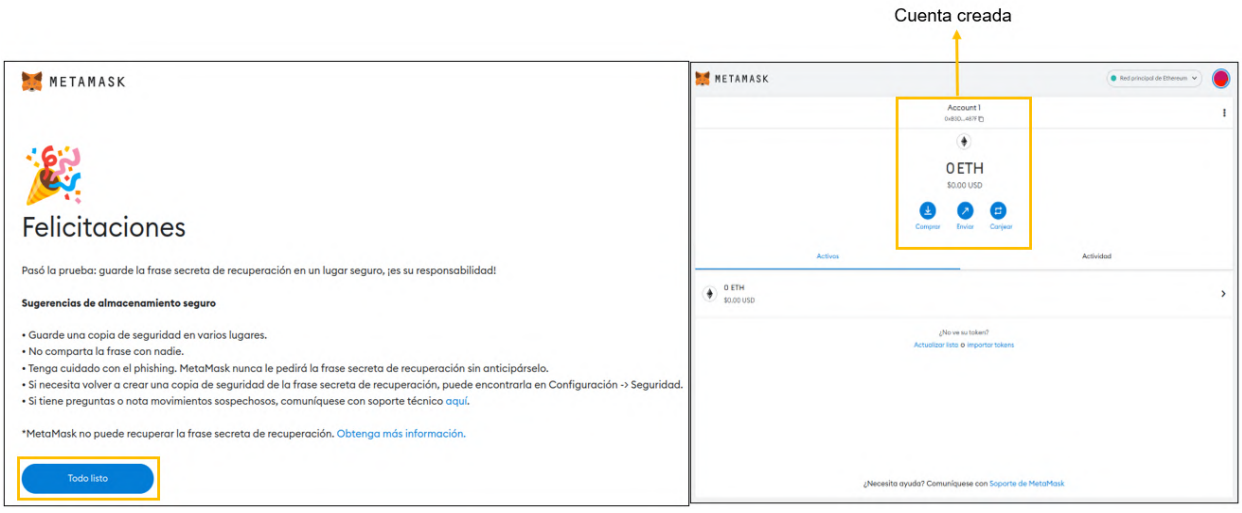


Figura 5.110. Paso 10. Mostrar cuenta en MetaMask creada

11. Para fijar la extensión en el navegador, ir al ícono de Extensiones y seleccionar el ícono Fijar. De esta manera, se muestra el ícono de MetaMask en el navegador y se puede acceder a la cartera rápidamente.

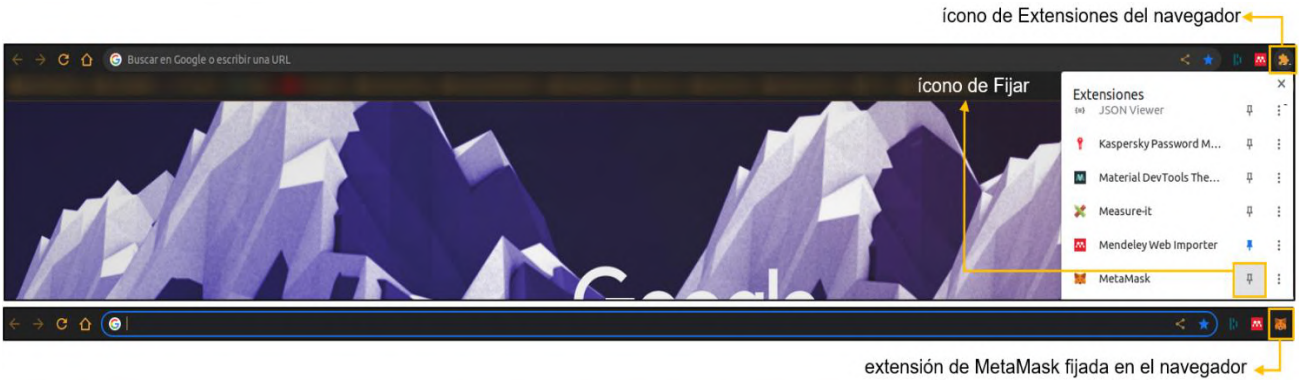


Figura 5.111. Paso 11. Mostrar ícono de MetaMask en el navegador

12. Cada que se desee acceder a la billetera creada, se abre el navegador, se selecciona el ícono de MetaMask y se ingresa la contraseña que se definió anteriormente. En el caso de no recordar esta contraseña, usar el mnemónico de la cuenta.

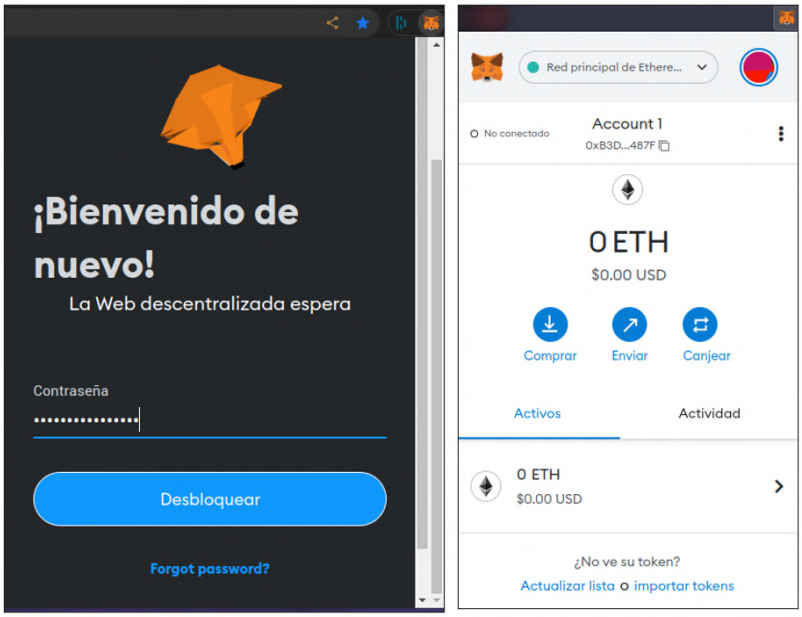


Figura 5.112. Paso 12. Inicio de sesión en la billetera MetaMask con la cuenta creada

Es importante mencionar que para emplear la aplicación web de CoffeeTrack, se requiere iniciar sesión con una cuenta MetaMask y seleccionar la red blockchain en la que se haya desplegado los contratos inteligentes, en este caso sería la red testnet Rinkeby. De lo contrario, no se podrá interactuar con la aplicación. Para agregar la testnet de Rinkeby en MetaMask, se sigue el procedimiento del Anexo IV y en caso de usar la red de Polygon para una interacción con una blockchain real, seguir el procedimiento del Anexo V.

- **DESPLIEGUE DE LOS CONTRATOS INTELIGENTES USANDO LA CUENTA DEL ADMINISTRADOR DEL SISTEMA**

  - **Requisitos previos para correr el sistema:**

Software	Versión
Sistema Operativo	Linux Ubuntu 20.04 LTS +
Nodejs	16.15.0
Truffle	5.4.32
Solidity Compiler	0.8.16
Git	2.26.2
Yarn	1.22.19

  - **Pasos de implementación:**

1. Una vez verificado la instalación de los requisitos previos, se clona el repositorio de GitHub (https://github.com/NathaliaBarreiros/coffee-supply-chain-tracking-system-ethereum) en el directorio deseado y se accede a la carpeta del proyecto. Para esto se ejecutan los siguientes comandos en una terminal del directorio:

```
git clone https://github.com/NathaliaBarreiros/coffee-supply-chain-tracking-system-ethereum.git
cd coffee-supply-chain-tracking-system-ethereum/
```

2. Para establecer las variables de entorno del proyecto, crear un archivo .env en la carpeta raíz del proyecto, copiar el contenido del archivo .env.example y configurar las variables de entorno de manera personalizada.

Cabe mencionar que para obtener las claves API tanto del nodo de Infura como del proyecto en Etherscan, valores que se requieren en el archivo .env, se siguen los procedimientos del Anexo XXI y Anexo XXII. Por otro lado, los valores del mnemónico y la dirección de la cuenta MetaMask requeridos en el archivo .env, deben corresponder a los del administrador del sistema.

3. En una terminal del proyecto, ejecutar los siguientes comandos para la instalación de algunas librerías requeridas, la compilación y el despliegue de los contratos:

```
rm -rf build/
npm install @truffle/hdwallet-provider
npm install truffle-plugin-verify
truffle compile
truffle migrate --network rinkeby reset
```

Una vez desplegados los contratos inteligentes se obtienen registros de consola similares a los mostrados en el Anexo XXVII apartado b). Las direcciones de los contratos inteligentes pueden ser consultadas en el explorador Etherscan. Las direcciones de los contratos CoffeeSupplyChain.sol, CoffeeSupplyChain2.sol y SupplyChainUser.sol deben ser guardadas para configurar la aplicación web (mostrado a continuación).

4. Para llevar a cabo la verificación de los contratos inteligentes desplegados en la testnet de Rinkeby, se ejecutan los siguientes comandos en la terminal del directorio del proyecto.

```
truffle run verify SupplyChainStorage –network rinkeby
truffle run verify SupplyChainStorage2 –network rinkeby
truffle run verify CoffeeSupplyChain –network rinkeby
truffle run verify CoffeeSupplyChain2 –network rinkeby
truffle run verify SupplyChainUser –network rinkeby
```

Los registros de consola obtenidos tras ejecutar los comandos anteriores deben ser similares a los mostrados en el Anexo XXVIII.

- **DESPLIEGUE DE LA APLICACIÓN WEB POR PARTE DEL ADMINISTRADOR DEL SISTEMA**

Se requiere hacer una bifurcación del repositorio de GitHub de la aplicación web CoffeeTrack a través de la URL https://github.com/NathaliaBarreiros/coffee-supply-chain-client, utilizando una cuenta personal de GitHub. Esto debido a que el alojamiento de la aplicación web en el servicio Vercel se lo realiza a través de un repositorio propio del proyecto.

1. Una vez accedida a la URL del proyecto, se selecciona la opción Fork, en la ventana emergente se selecciona el botón Create fork. Una vez que el proyecto se ha bifurcado correctamente, se observa el proyecto creado en la cuenta personal de GitHub.

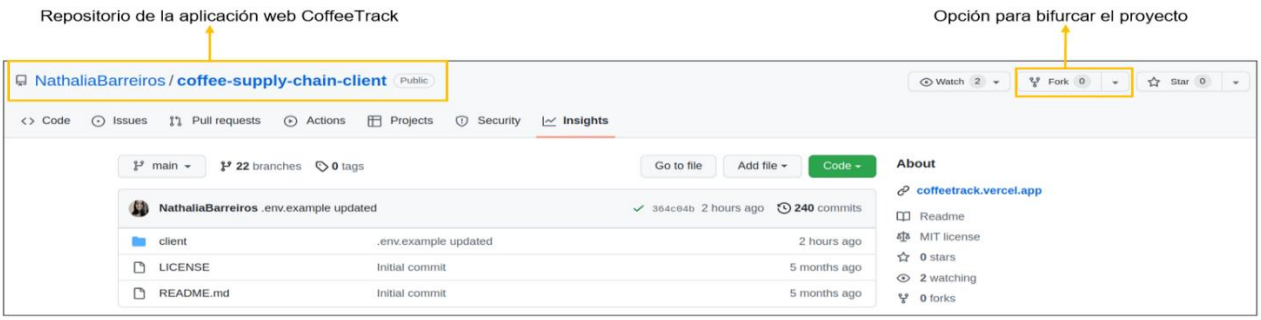


Figura 5.113. Paso 1. Selección de la opción Fork del repositorio original de la aplicación

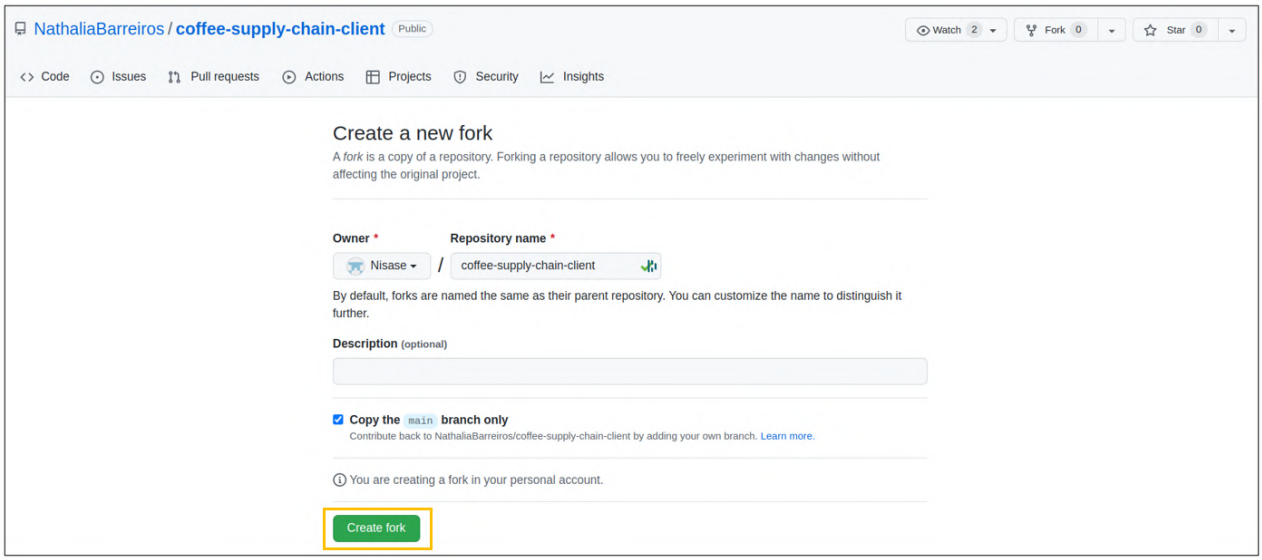


Figura 5.114. Paso 1. Selección de la opción Create fork

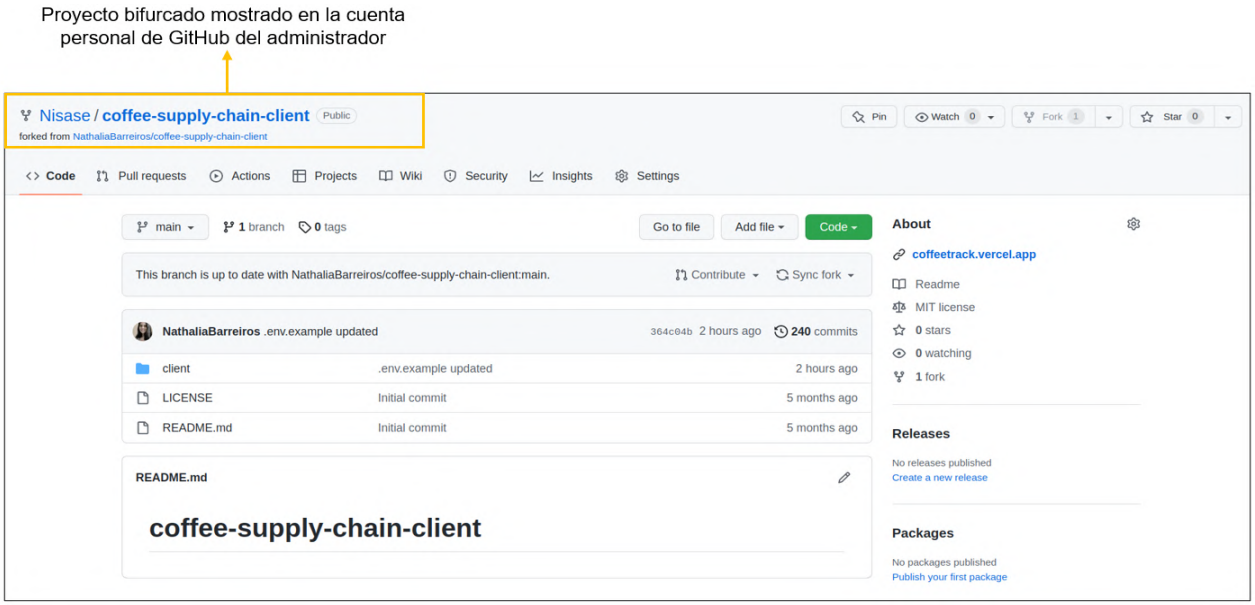


Figura 5.115. Paso 1. Proyecto bifurcado mostrado en el repositorio de GitHub del administrador

2. A continuación, se clona localmente el repositorio que se bifurcó. Para esto se selecciona la opción Code y se copia la URL del repositorio bifurcado en la cuenta personal de GitHub del administrador del sistema.

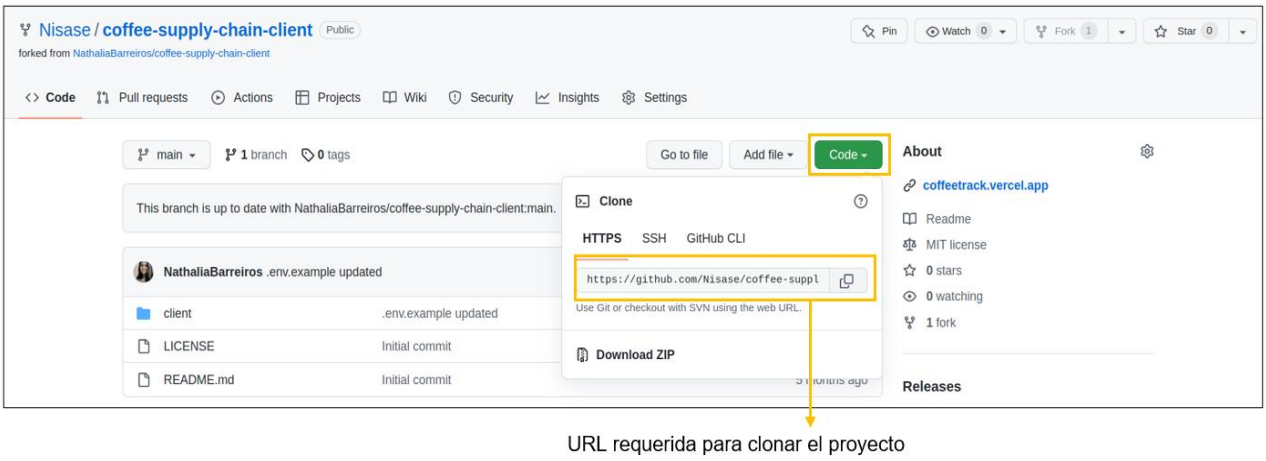


Figura 5.116. Paso 2. URL requerida para clonar el proyecto de la cuenta personal de GitHub del administrador

Luego, se accede a la carpeta del proyecto, */client*. Para esto se ejecutan los siguientes comandos en una terminal:

```
git clone https://github.com/NombreUsuarioGitHub/coffee-supply-chain-client.git
cd client
```

3. Para establecer las variables de entorno del proyecto, crear un archivo .env en la carpeta raíz del proyecto (*/client*), copiar el contenido del archivo .env.example y configurar las variables de entorno de manera personalizada.

Para obtener las variables de entorno correspondientes a las claves API del servicio de Google Maps, el nodo de Infura para conectarse a la testnet de Rinkeby y el nodo de Infura para conectarse a IPFS, se siguen los procedimientos del Anexo XXXV, Anexo XXI y Anexo XXXII, respectivamente.

Para definir las variables de entorno correspondientes al servicio de EmailJS, se realiza lo siguiente:

- **Creación de una cuenta en EmailJS y configuración del servicio**

EmailJS es una plataforma que se emplea para enviar y recibir correos electrónicos sin utilizar un servidor, debido a que la aplicación CoffeeTrack implementa esta funcionalidad en la página principal para que los usuarios interesados en utilizar la aplicación puedan escribir al correo de CoffeeTrack directamente desde la aplicación. Para esto, se requiere llevar a cabo la configuración de este servicio siguiendo los pasos siguientes:

a) Crear una cuenta en la plataforma de EmailJS a través de la URL https://www.emailjs.com/. Para esto, seleccionar el botón Create Free Account. En la

página de redirección, llenar el formulario con los datos correspondientes. En la siguiente página, ingresar a la cuenta con los datos previos y confirmar el correo enviado.

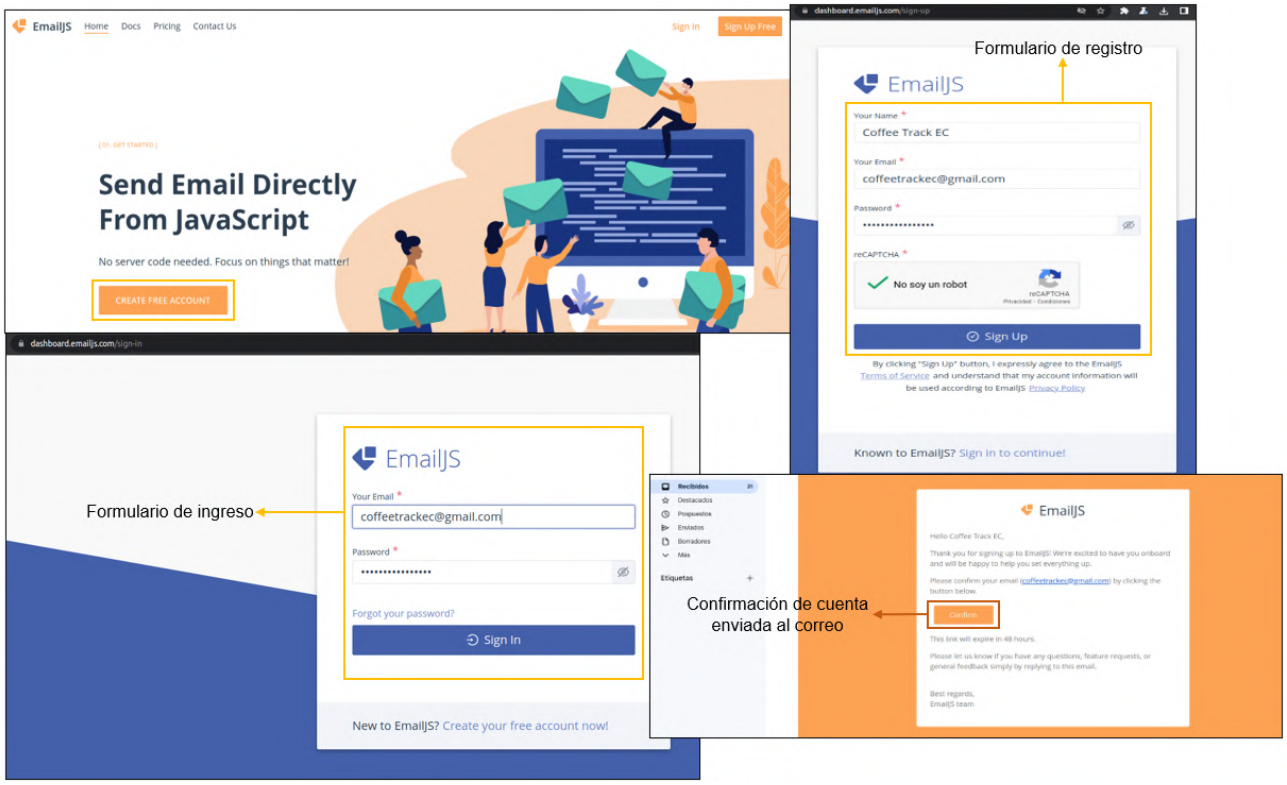


Figura 5.117. Paso a) Creación de cuenta en EmailJS

b) En el panel de control, añadir un nuevo servicio de email seleccionando la pestaña Email Services, luego el botón Add New Service. En la ventana emergente, seleccionar el servicio de Gmail. En la siguiente ventana modificar los campos del nombre del servicio y el ID, si se desea. Luego, seleccionar el botón Connect Account y en la ventana emergente de Gmail, seleccionar la cuenta de correo que se desea conectar y seleccionar Continuar. Finalmente, seleccionar el botón Create Service.

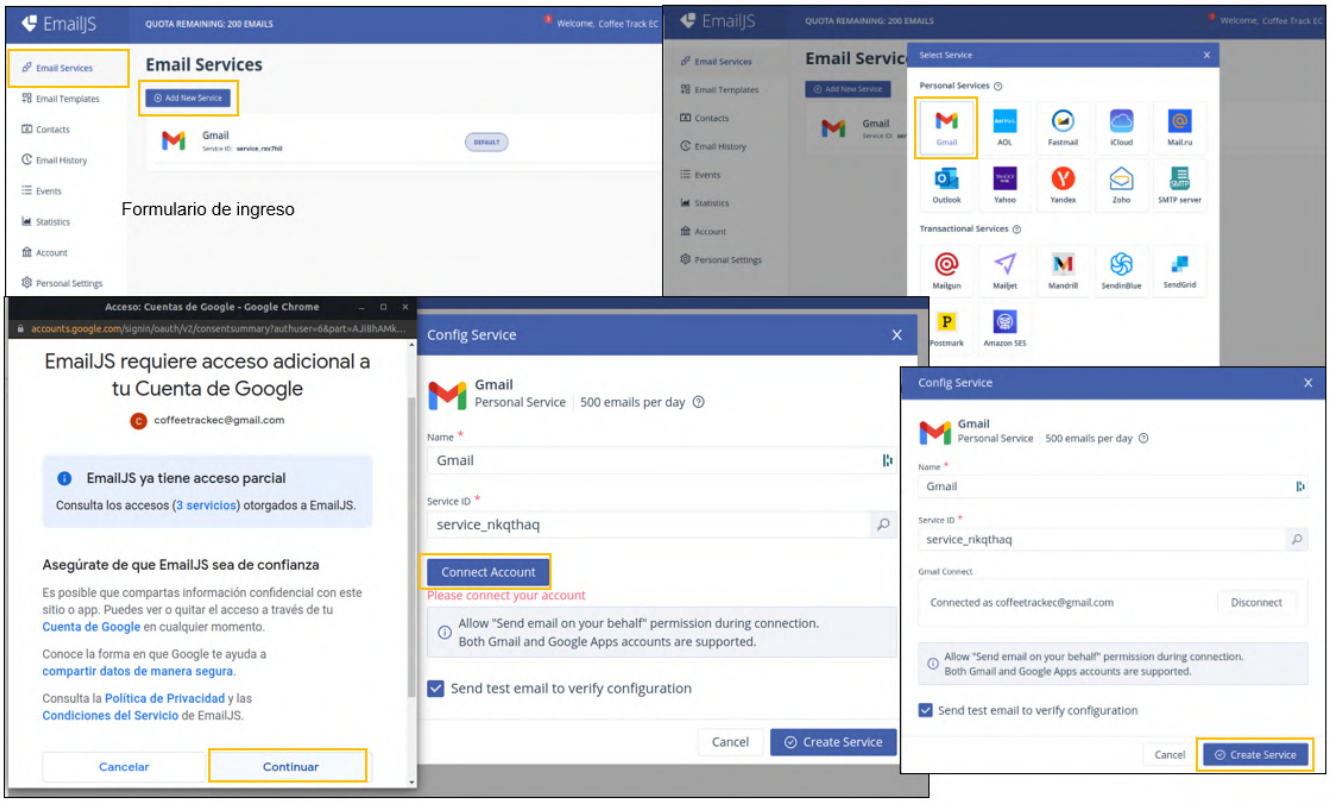


Figura 5.118. Paso b) Creación de un nuevo servicio con Gmail en la cuenta de EmailJS

c) Para generar una plantilla de envío y respuesta de correo automático, se selecciona la pestaña Email Templates en el panel de control. A continuación, se selecciona Create New Template. Se configura la pestaña Content como se muestra en la Figura 5.120, mientras que la pestaña Auto-Reply, se configura como en la Figura 5.121. Finalmente, seleccionar el botón Save.

Figura 5.119. Paso c) Creación de una plantilla para el envío y respuesta de correo automático

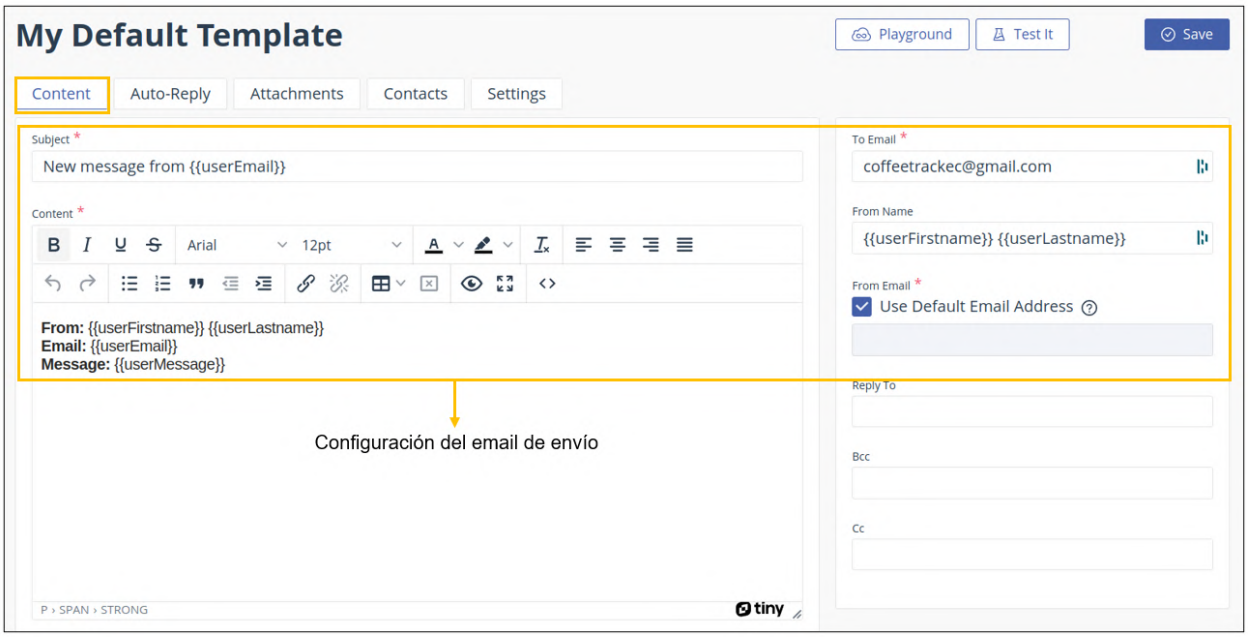


Figura 5.120. Paso c) Configuración de la pestaña Content para la plantilla de envío de correo automático

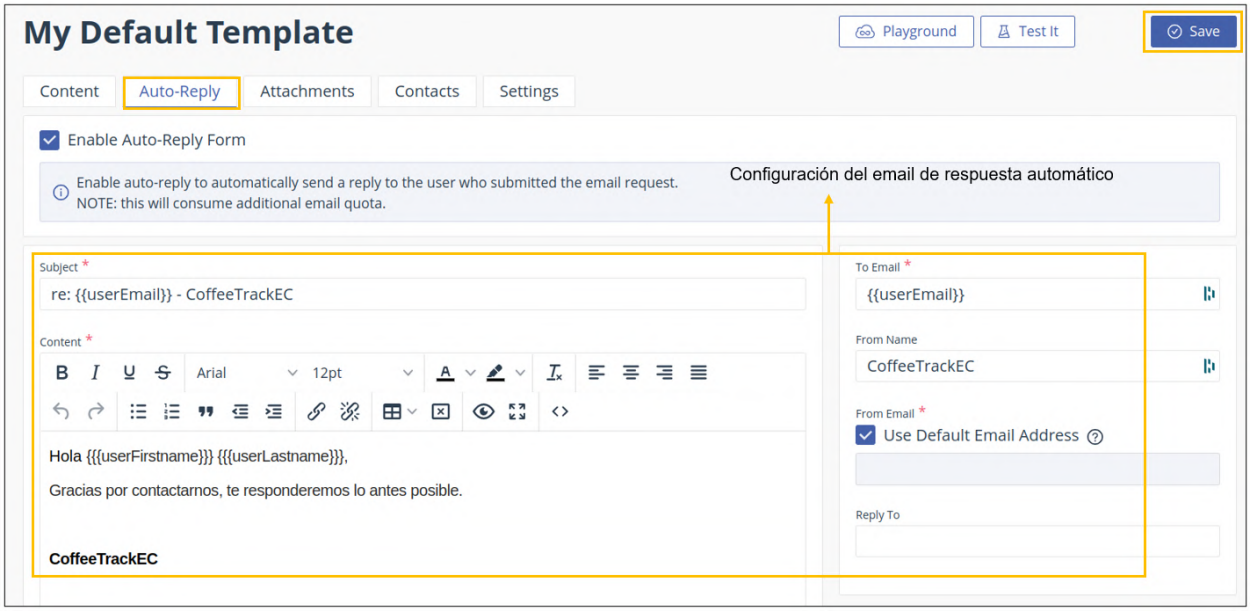


Figura 5.121. Paso c) Configuración de la pestaña Auto-Reply para la plantilla de respuesta de correo automático

Con lo anterior, se puede obtener el ID de la plantilla de envío y respuesta de correo generada, el ID del servicio de correo configurado y la clave API pública de la cuenta creada, cuyos valores son los requeridos para configurar el archivo .env del proyecto. Estos valores se muestran en las figuras siguientes.

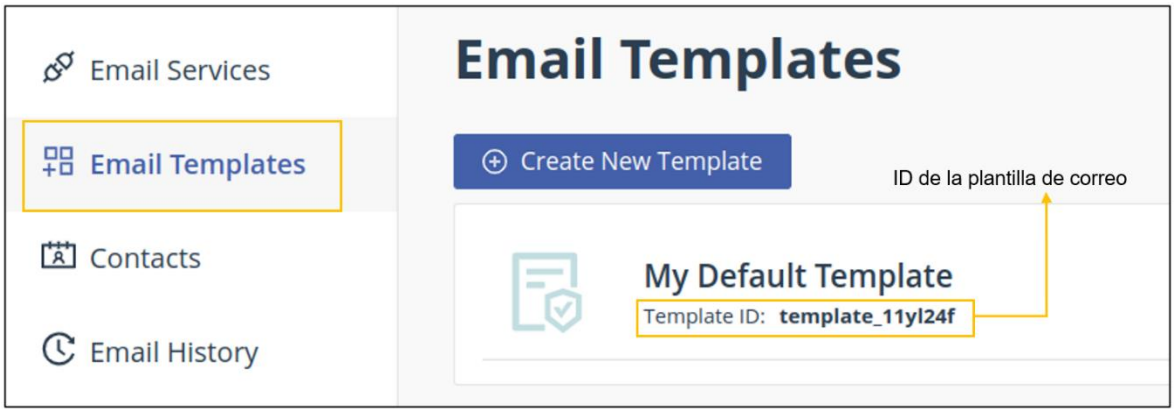


Figura 5.122. Paso c) ID de la plantilla de envío y respuesta de correo generada

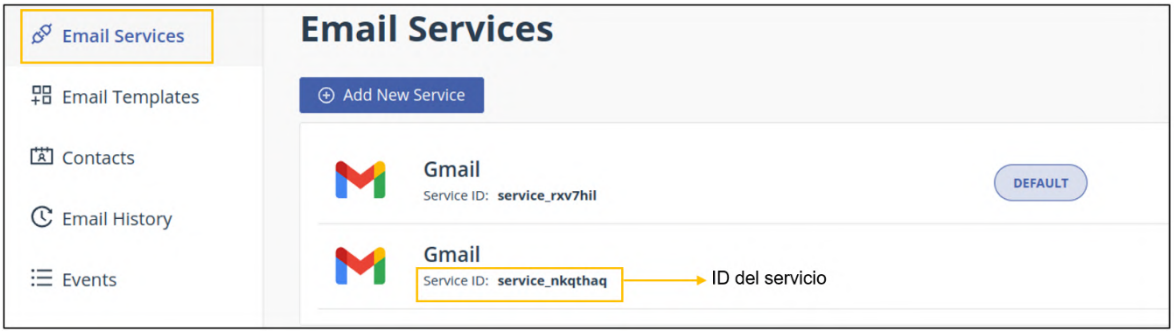


Figura 5.123. Paso c) ID del servicio de correo configurado

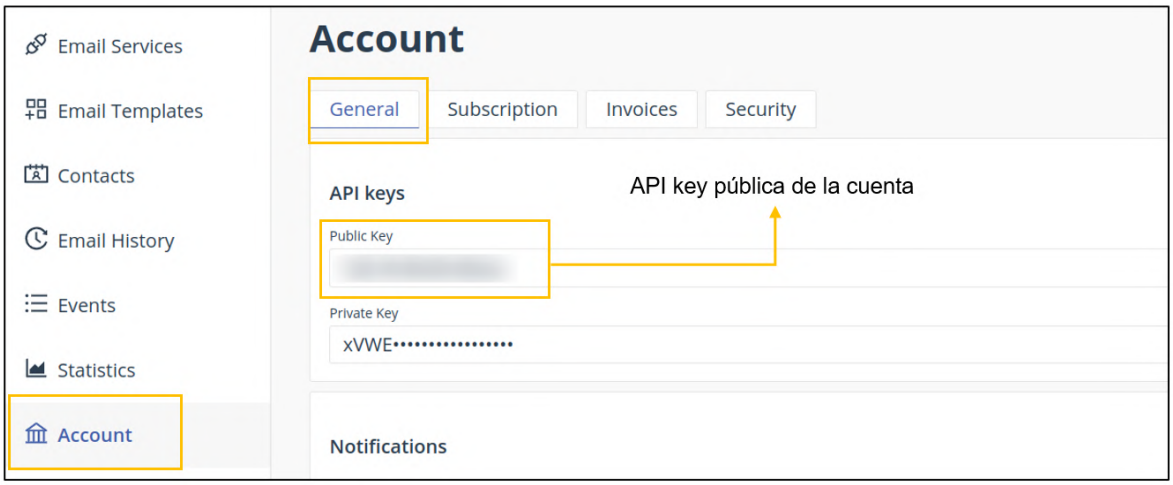


Figura 5.124. Paso c) Clave API pública de la cuenta creada

4. Para instalar los paquetes del proyecto, se corre el siguiente comando:

```
yarn add https://github.com/NombreUsuarioGitHub/coffee-supply-chain-client.git
```

5. En el archivo App.js de la carpeta */src* dentro de */client*, reemplazar las direcciones de los contratos inteligentes CoffeeSupplyChain.sol, CoffeeSupplyChain2.sol y SupplyChainUser.sol, por las direcciones obtenidas en el despliegue de los contratos antes mostrado. Estos valores se encuentran desde la línea 88 a la línea 90 del archivo App.js, los reemplazos deberían realizarse como sigue:

```
const userAddress = 'dirección_contrato_SupplyChainUser.sol';
const coffeAddress1 = 'dirección_contrato_CoffeeSupplyChain.sol';
const coffeAddress2 = 'dirección_contrato_CoffeeSupplyChain2.sol';
```

Figura 5.125. Paso 5. Líneas de código del archivo App.js para modificar las direcciones de los contratos desplegados

Se puede probar la aplicación web de manera local a través del comando *yarn start*, lo que abre la URL http://localhost:3000/ en el navegador.

6. Para hacer pública la implementación de la aplicación web en el servicio de Vercel, se sigue el procedimiento del Anexo XXXIX.

➢ **Modificaciones:**

Si se realizan cambios en los contratos inteligentes, se debe actualizar el código de la aplicación web. Sobre todo, se deben actualizar los archivos ABI de los contratos inteligentes presentes en el directorio */client/src/contracts*. Los archivos a modificar serían: CoffeeSupplyChain1.json, CoffeeSupplyChain2.json, SupplyChainUser1.json. Para esto se debe copiar el objeto abi de los contratos correspondientes, que son producto de la compilación de estos, los cuales se encuentran en la carpeta */build/contracts* del primer proyecto clonado.

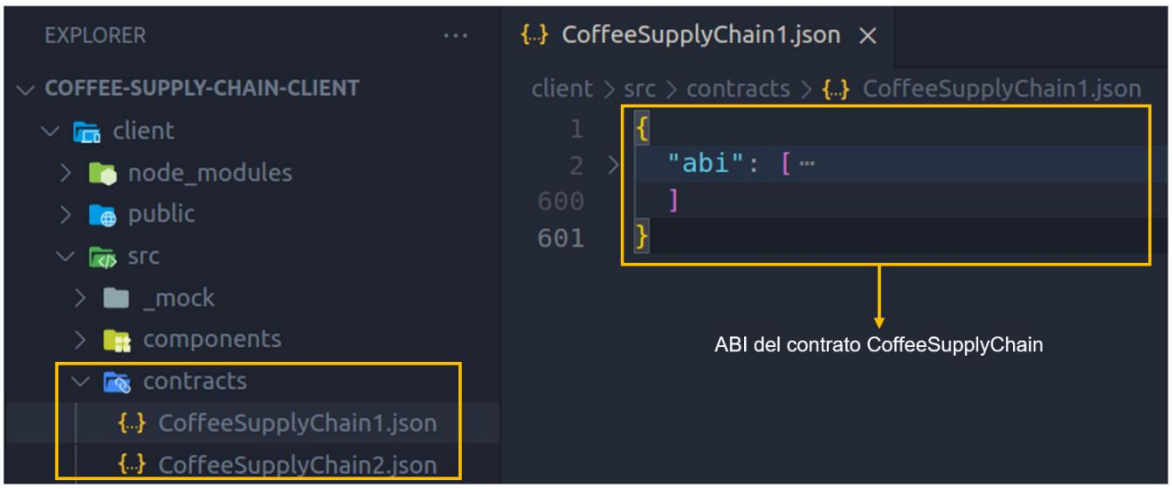


Figura 5.126. Código ABI de los archivos .json del proyecto de la aplicación web

Al realizar cambios en el repositorio local de la aplicación web, ya sea al actualizar la dirección de los contratos, cambiar el ABI de estos o cualquier otro cambio, se debe actualizar y subir los cambios en el repositorio remoto de GitHub a través de los comandos siguientes:

```
git add .
git commit –m “Comentario”
git push dirección_SSH_repositorio_GitHub
```

La dirección SSH del repositorio se obtiene accediendo al repositorio en GitHub, seleccionando la opción Code, luego la opción de dirección SSH. Para esto se debe vincular el repositorio remoto de GitHub con el repositorio local a través de una clave SSH previamente configurada.

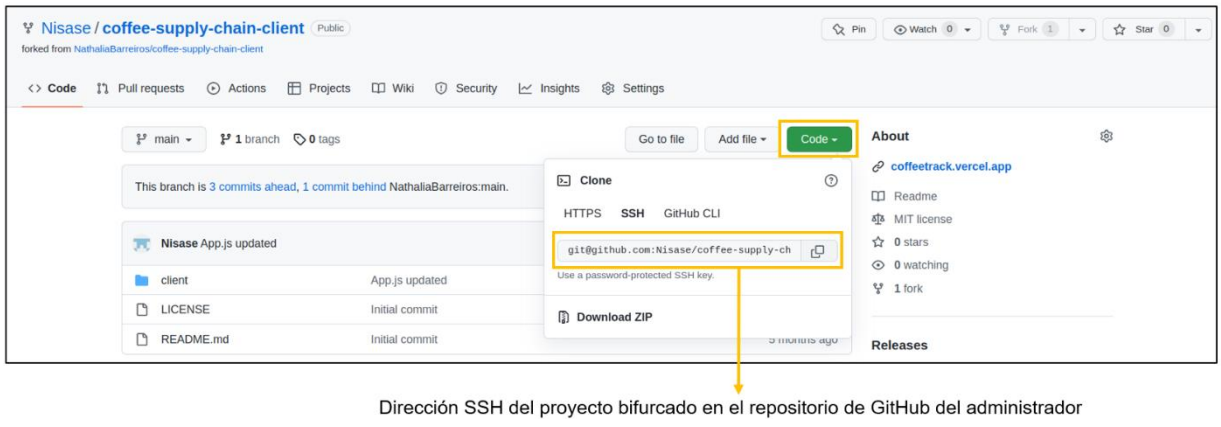


Figura 5.127. Dirección SSH del proyecto de la aplicación web bifurcado en la cuenta personal del administrador

- **USO DE LA APLICACIÓN WEB**

  - **Página Inicio (Home)**

Esta es la página inicial de la aplicación cuando se accede a la URL https://coffeetrack.vercel.app/home y contiene información sobre la aplicación CoffeeTrack como sistema de tracking de la cadena de suministro del café de especialidad, que utiliza la tecnología blockchain. La visualización de esta página se muestra a continuación.

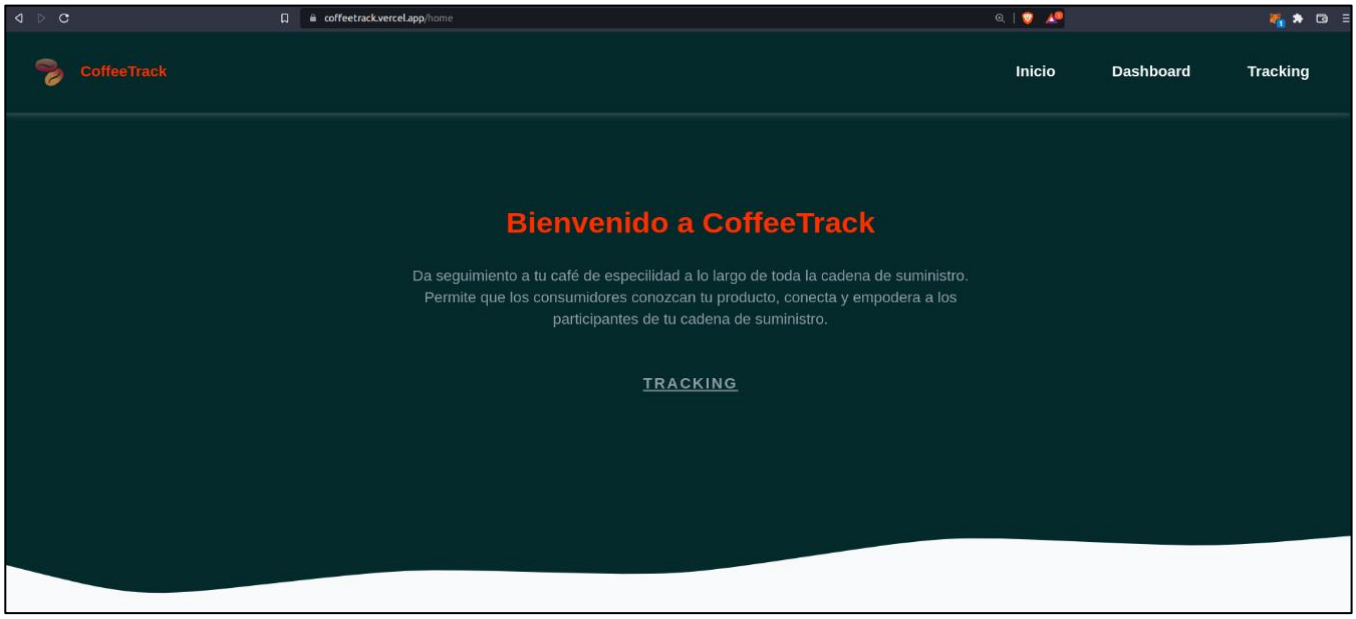


Figura 5.128. Página de Inicio: sección de bienvenida

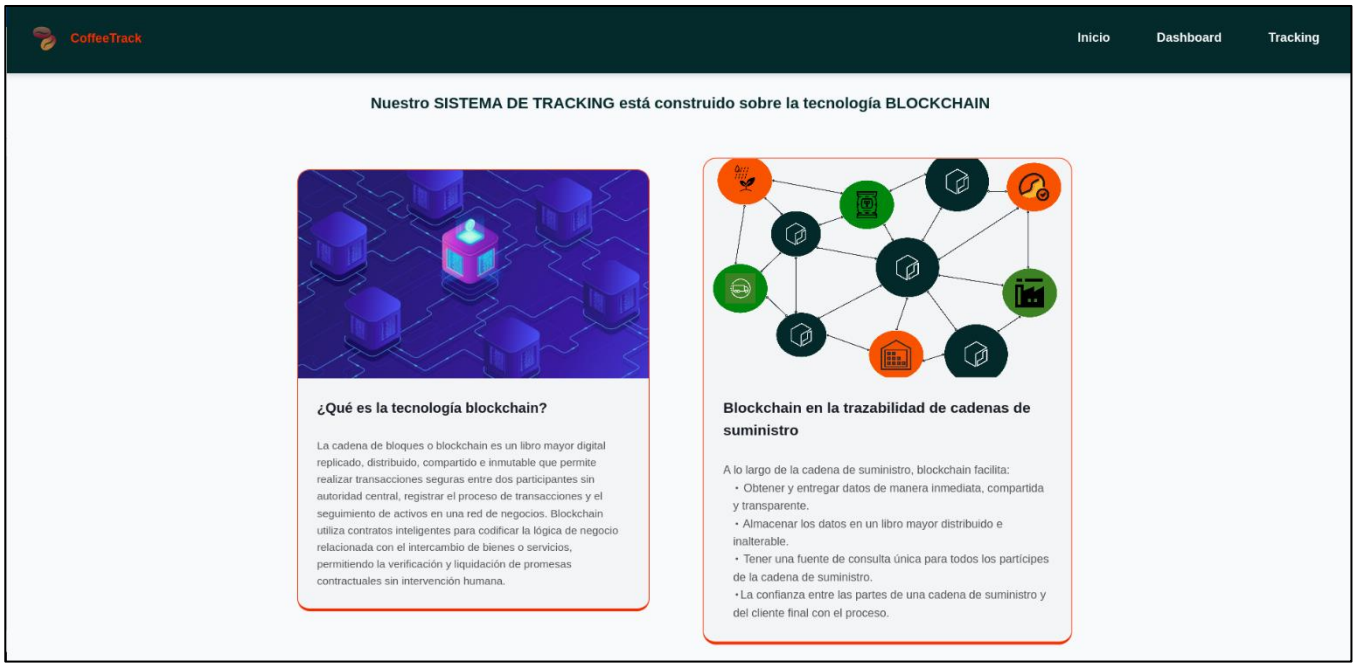


Figura 5.129. Página de Inicio: sección sobre la tecnología blockchain y la trazabilidad

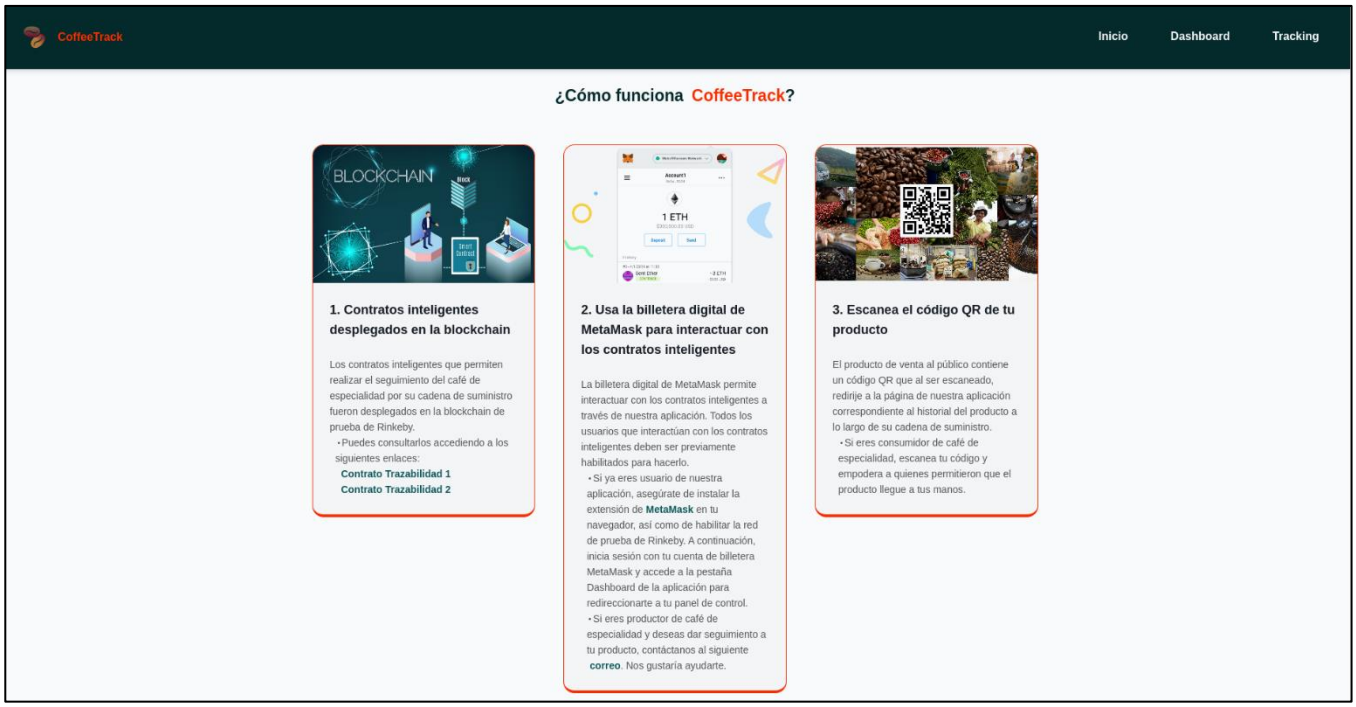


Figura 5.130. Página de Inicio: sección sobre el funcionamiento de CoffeeTrack

Además, en la sección final de esta página se muestra un formulario que permite a los usuarios que son parte de una cadena de suministro agrícola del café de especialidad y que se encuentran interesados en probar el sistema, comunicarse con el administrador del sistema. Este formulario al ser llenado y enviado manda de manera automática un correo a CoffeeTrack, y el usuario interesado recibirá un correo de respuesta automático, hasta que el administrador se contacte de manera personalizada con el usuario interesado. Un ejemplo del uso del formulario de contacto se muestra en la Figura 5.131.

Figura 5.131. Página de Inicio: sección del formulario de contacto

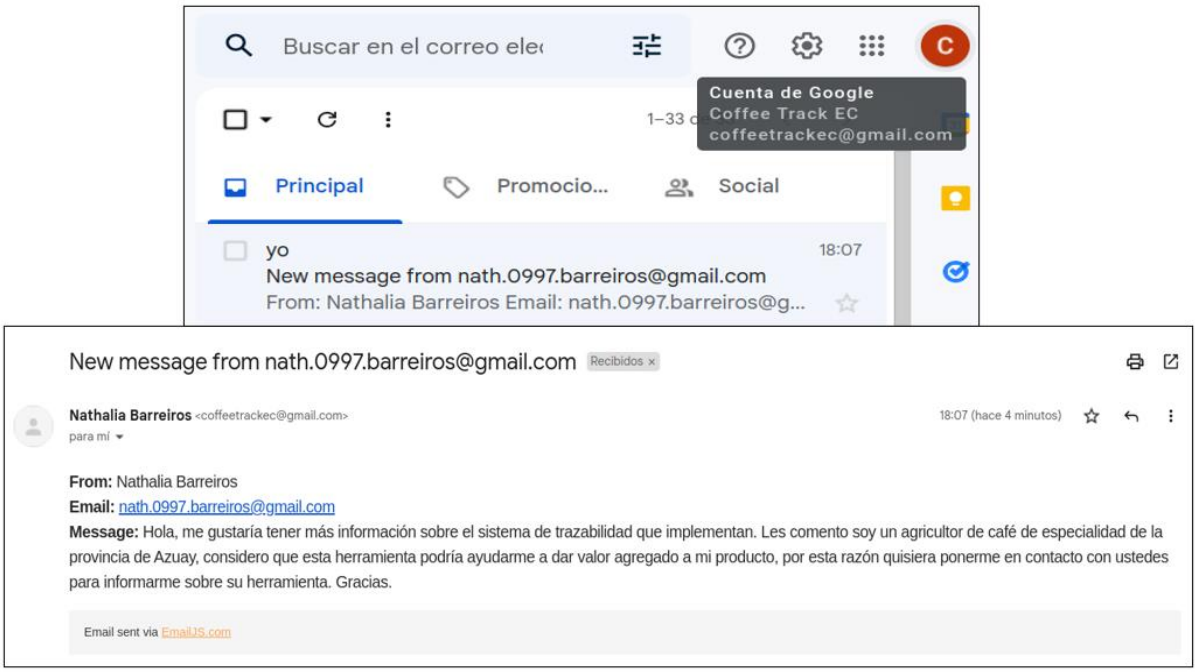


Figura 5.132. Mensaje enviado a la bandeja de entrada del correo de CoffeeTrack

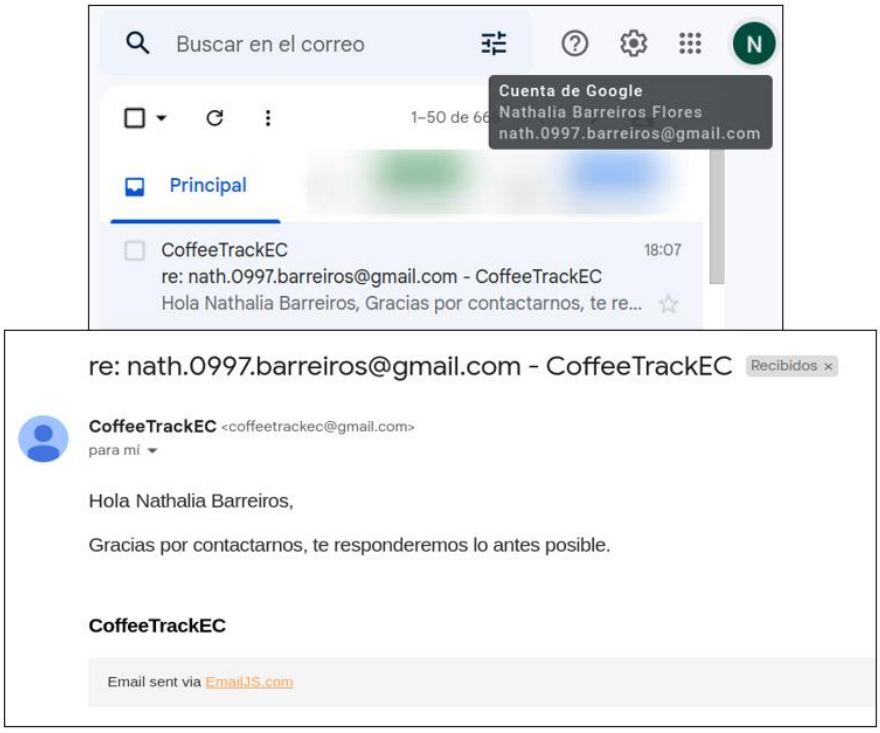


Figura 5.133. Correo de respuesta enviado al correo del usuario interesado

- **Página Dashboard**

Esta página puede ser accedida únicamente por el administrador del sistema y por los usuarios modificadores que hayan sido ingresados en el sistema por el administrador. Para

esto, se requiere instalar la extensión de MetaMask en el navegador, crear una cuenta e iniciar sesión con la cuenta de MetaMask con la que haya sido ingresado el usuario en el sistema, en el caso de un usuario modificador. En el caso del administrador, este debe iniciar sesión con su cuenta MetaMask con la que realizó el despliegue de los contratos inteligentes en la blockchain. Caso contrario, ninguno de estos usuarios podrá acceder a su panel de control. Así también, todos los usuarios que accedan a la aplicación deben configurar la red testnet Rinkeby en MetaMask, esto se muestra en el Anexo IV. Una vez ingresado al panel de control, todos los usuarios modificadores como el administrador, pueden ingresar la información referente a su intervención, así como consultarla. A continuación, se observa la página de Login para el inicio de sesión en la aplicación, el panel de control del administrador y de los usuarios modificadores con su rol asignado.

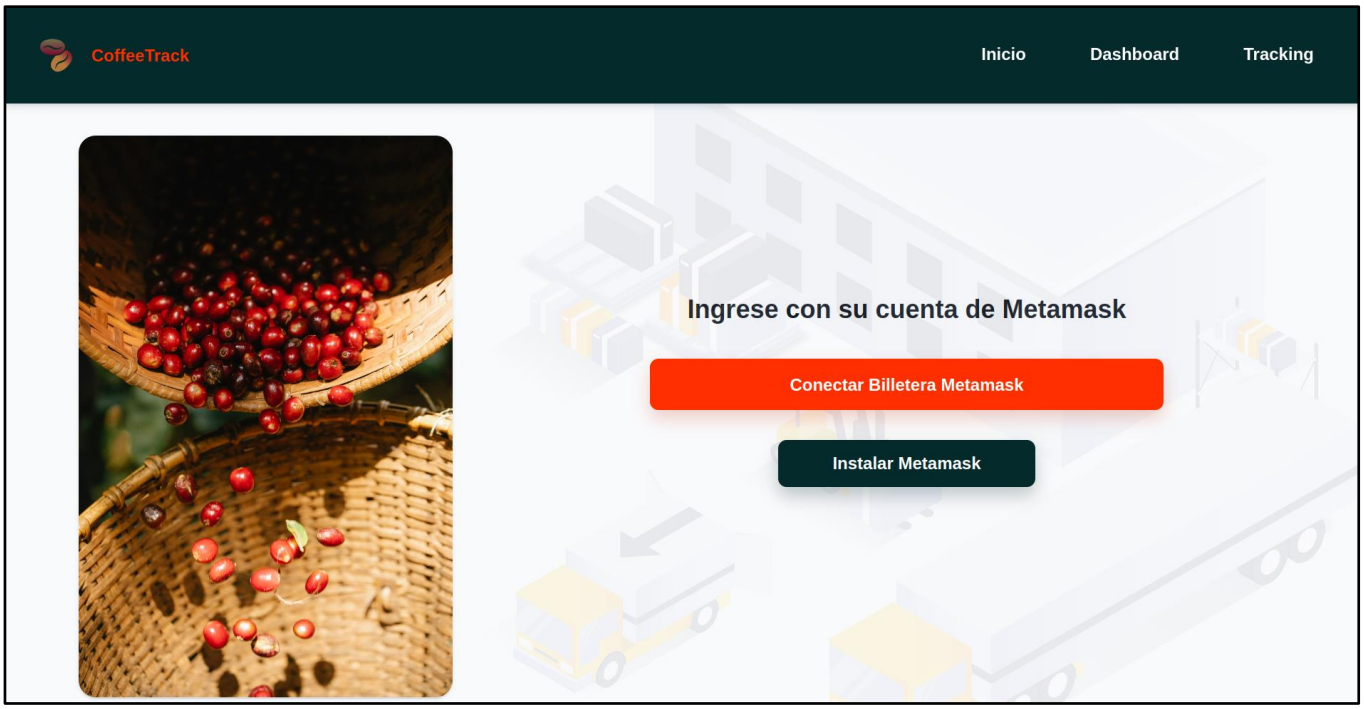


Figura 5.134. Login de la aplicación

a) Usuario Administrador

El panel de control del administrador muestra las opciones de AGREGAR USUARIOS y AGREGAR LOTE, así como una tabla que muestra todos los lotes de café creados en el sistema por parte del usuario administrador.

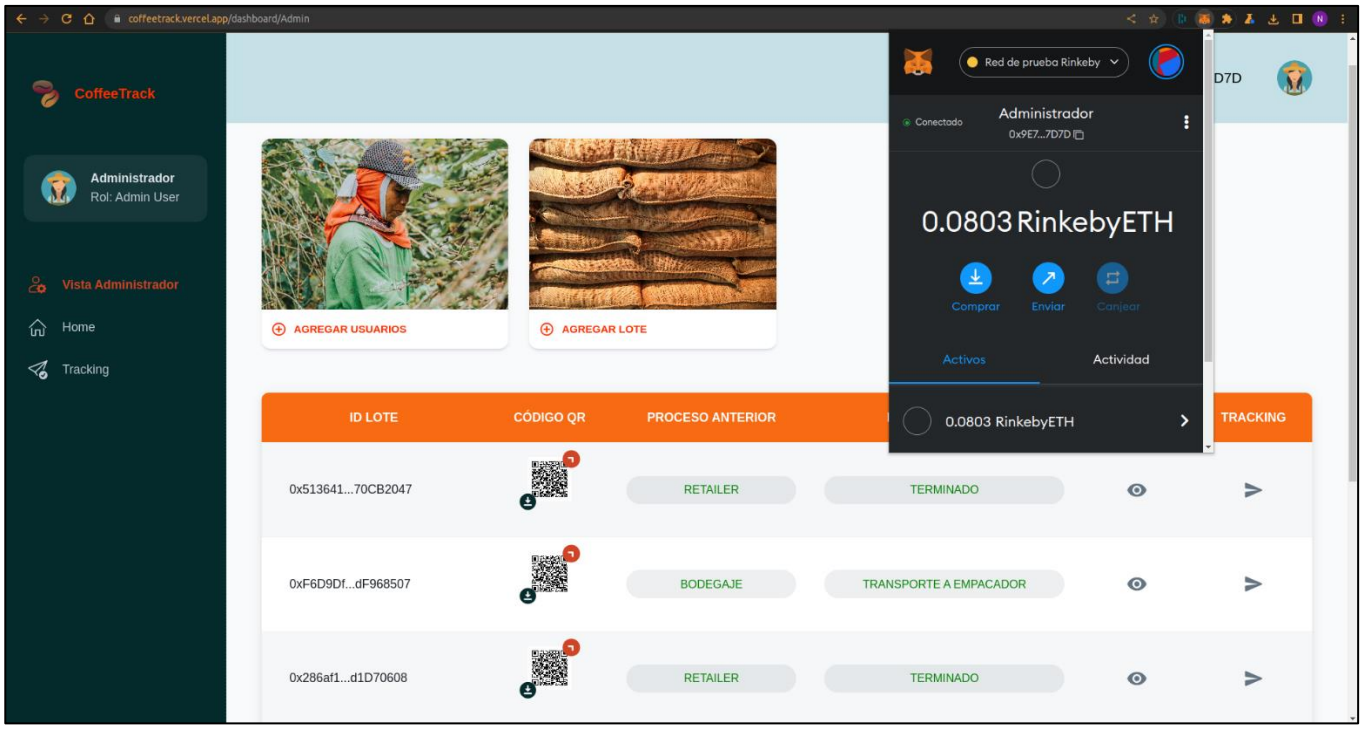


Figura 5.135. Panel de control del administrador mostrado al ingresar con su cuenta de billetera MetaMask

El formulario para agregar usuarios se muestra en la Figura 5.136. Mientras que la Figura 5.137 muestra el formulario para agregar lote.

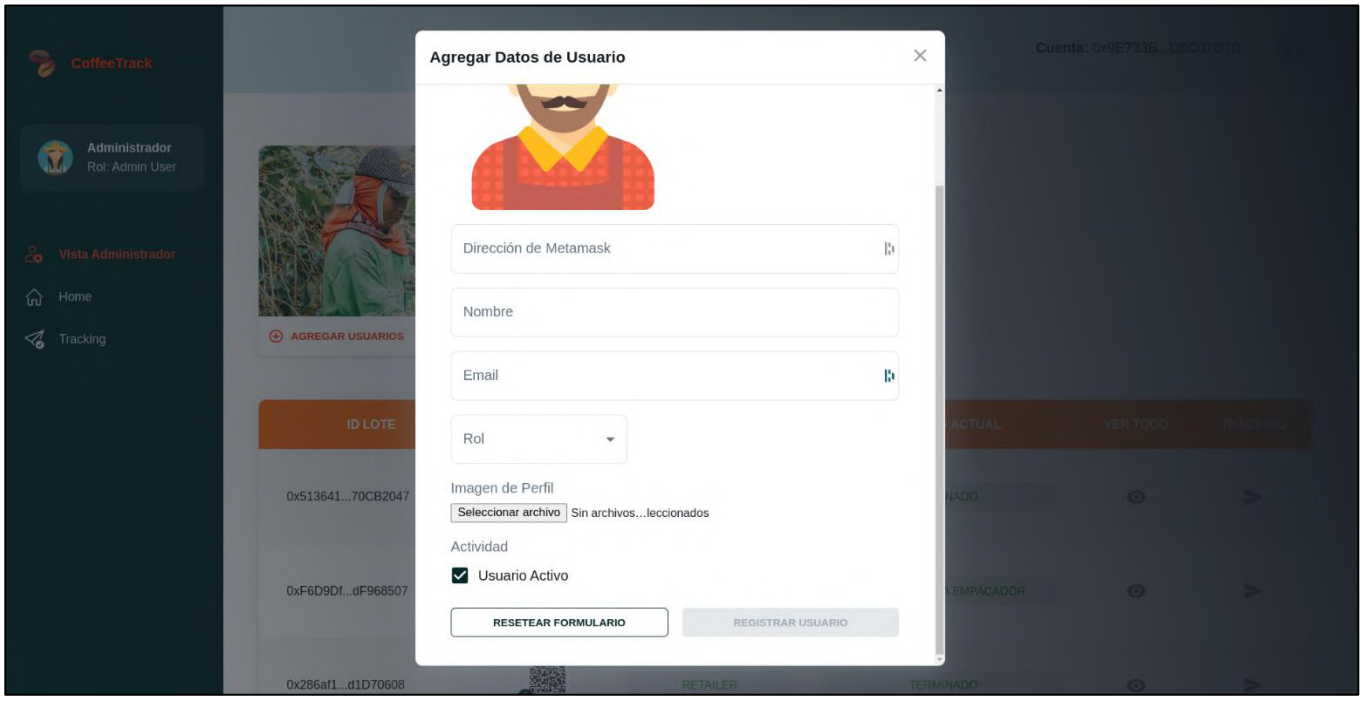


Figura 5.136. Formulario para agregar usuarios en el sistema

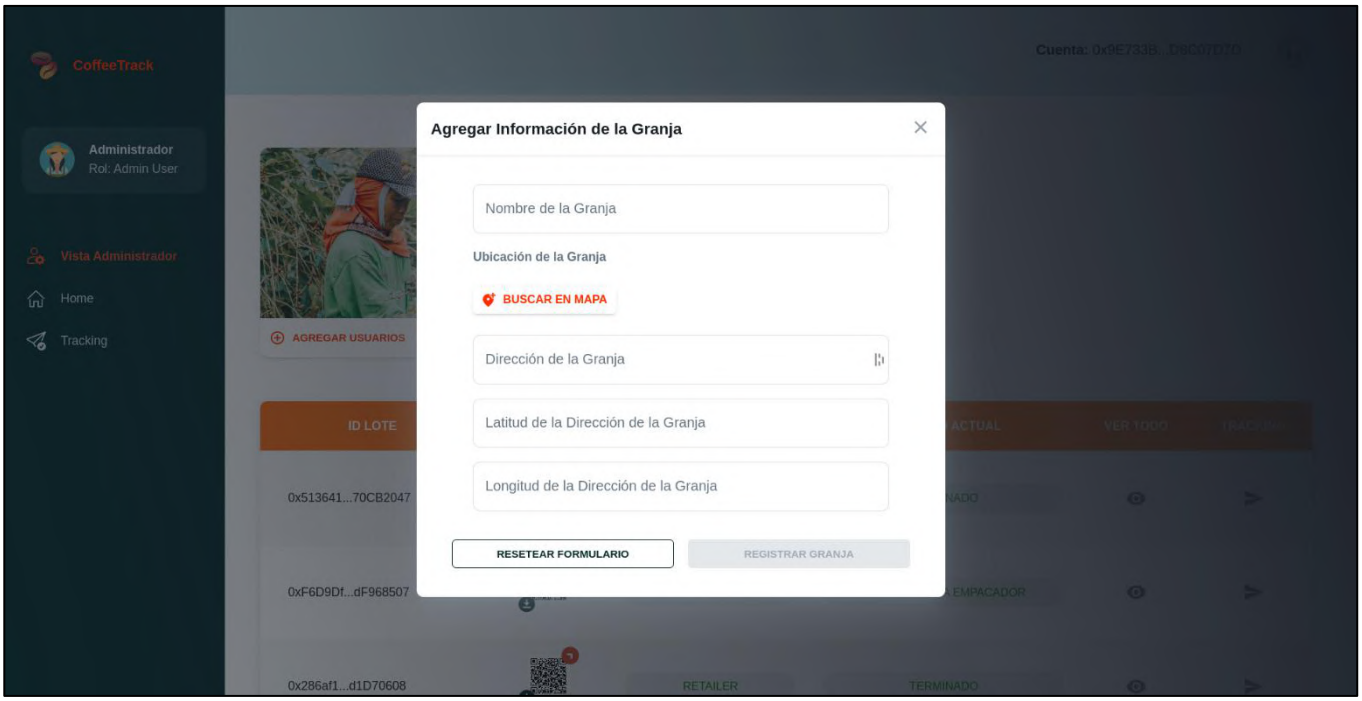


Figura 5.137. Formulario para registrar granja en el sistema y crear ID de lote de café

La opción de BUSCAR EN MAPA se muestra en la Figura 5.138. Cabe mencionar que esta ventana emergente es similar para los usuarios modificadores.

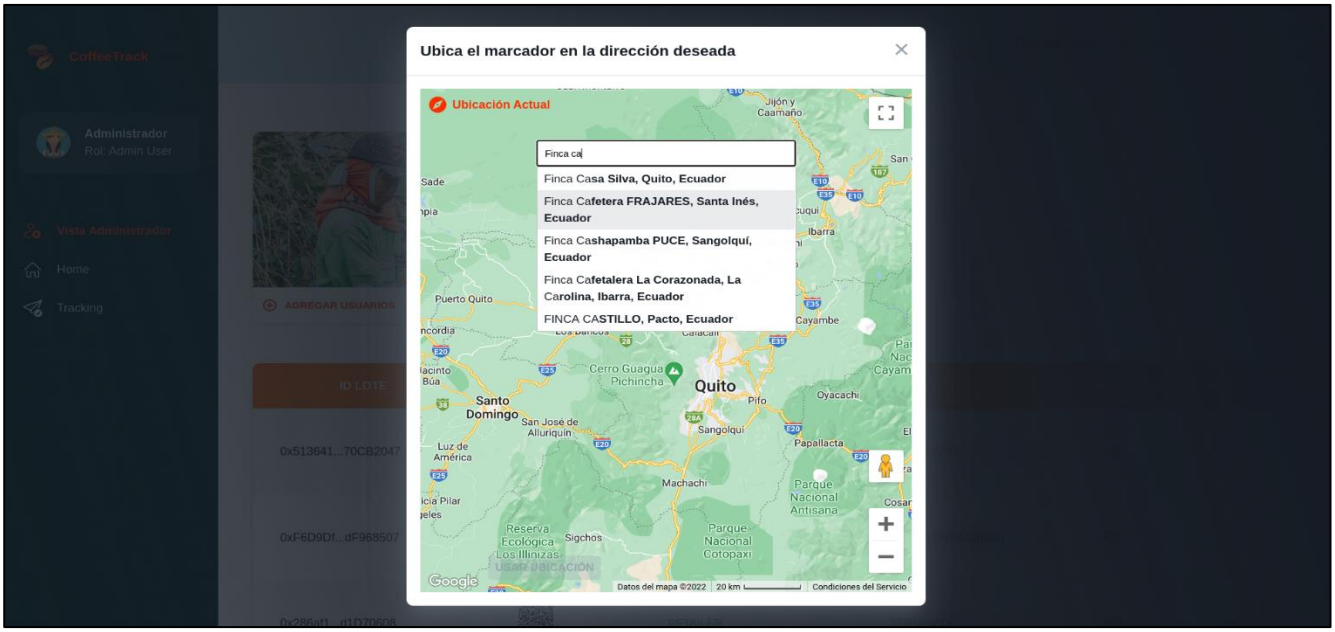


Figura 5.138. Ventana emergente de la opción BUSCAR EN MAPA

En la Figura 5.139, se muestra la tabla de los lotes de café creados en el sistema. Se observa que cada fila representa la información sobre un lote específico creado, la primera columna tiene el ID del lote y permite compartirlo. La segunda columna permite descargar

el código QR y ampliar la imagen del código, esto se visualiza en la Figura 5.140. La columna VER TODO contiene un ícono que abre una ventana emergente de la línea de tiempo de los procesos y sus estados para un lote específico, como se muestra en la Figura 5.141. La columna TRACKING, redirige a la página de la información de trazabilidad del lote específico, como se muestra en la Figura 5.142.

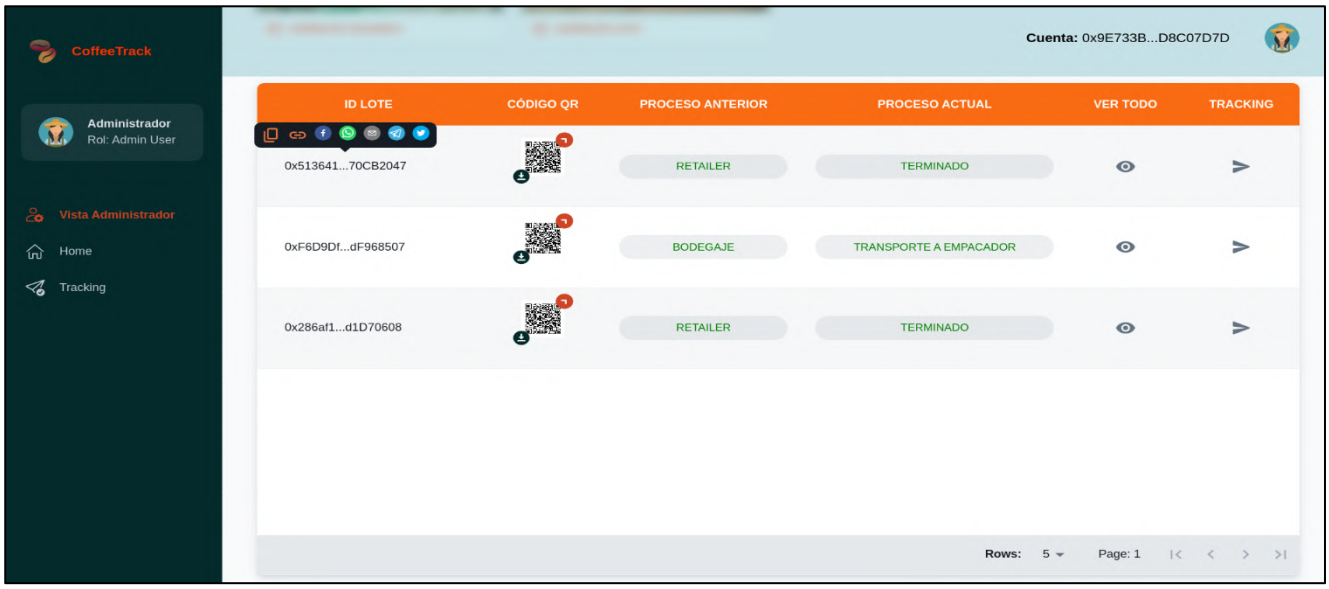


Figura 5.139. Tabla de los lotes de café creados en el sistema y mostrados en el panel de control del administrador

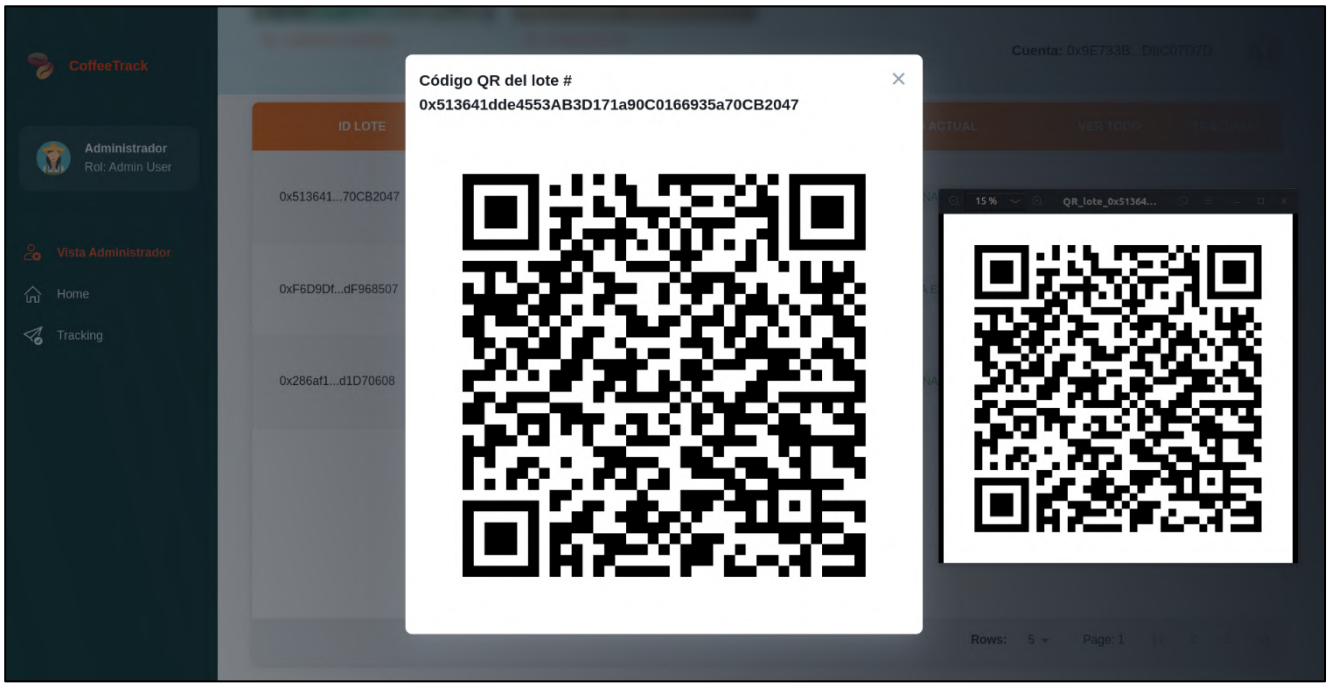


Figura 5.140. Opción para ampliar la imagen de un código QR específico e imagen descargada

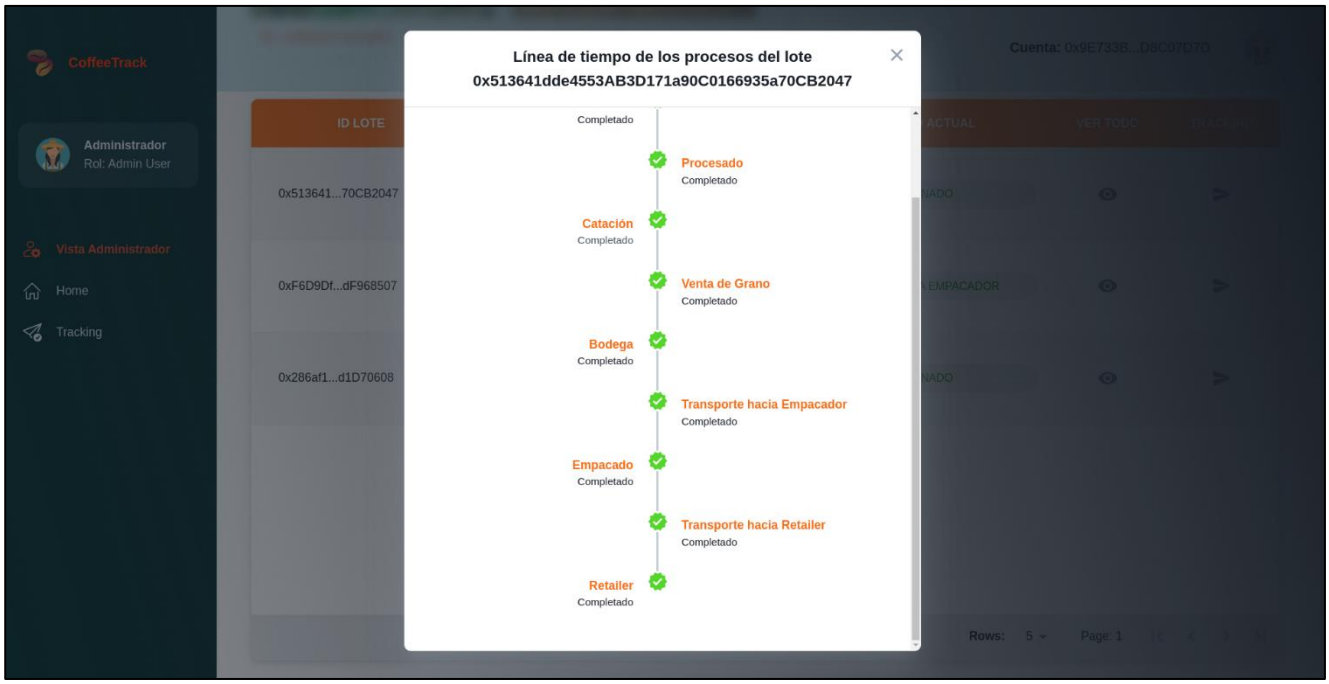


Figura 5.141. Línea de tiempo de los procesos y sus estados para un lote café específico

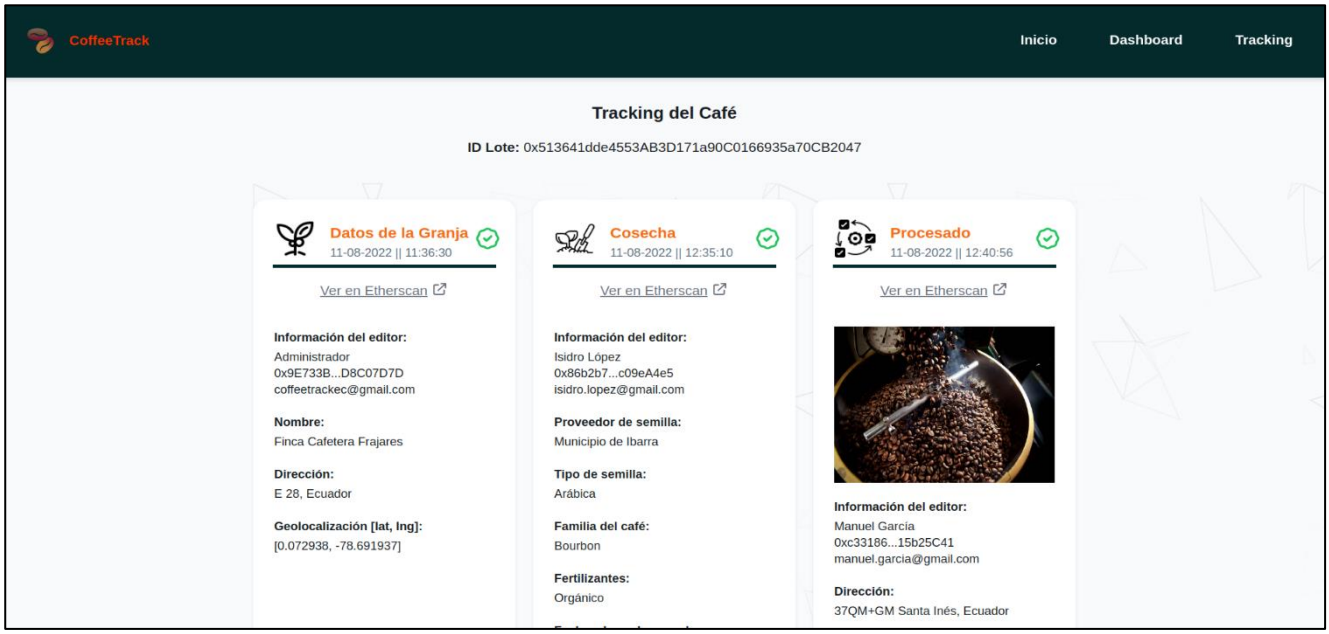


Figura 5.142. Página de redirección tras seleccionar el ícono TRACKING de un lote de café específico en la tabla del panel de control del administrador

b) Usuario Modificador

- **Rol Agricultor**

La Figura 5.143. muestra el panel de control del usuario modificador con los roles de Agricultor y Vendedor de Café. El panel de control muestra la vista de Agricultor como de

Vendedor de Café. La vista de Agricultor contiene las opciones de ACTUALIZAR PERFIL, AGREGAR COSECHA y AGREGAR COSECHA CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Agricultor.

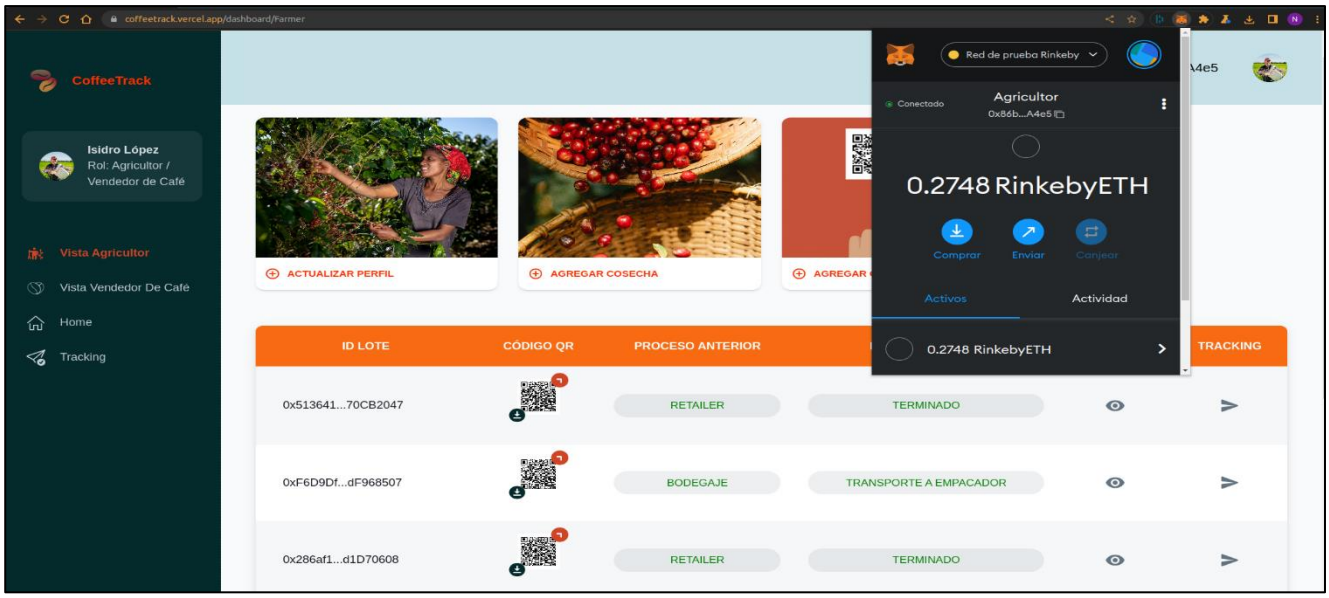


Figura 5.143. Panel de control de un usuario con rol de Agricultor

El formulario para actualizar perfil se muestra en la Figura 5.144. Mientras que la Figura 5.145 muestra el formulario para agregar cosecha. La Figura 5.146 muestra la ventana emergente de la opción AGREGAR COSECHA CON CÓDIGO QR, la cual al escanear un código QR válido, habilita la opción de AGREGAR COSECHA lo que abre el formulario mostrado en la Figura 5.145.

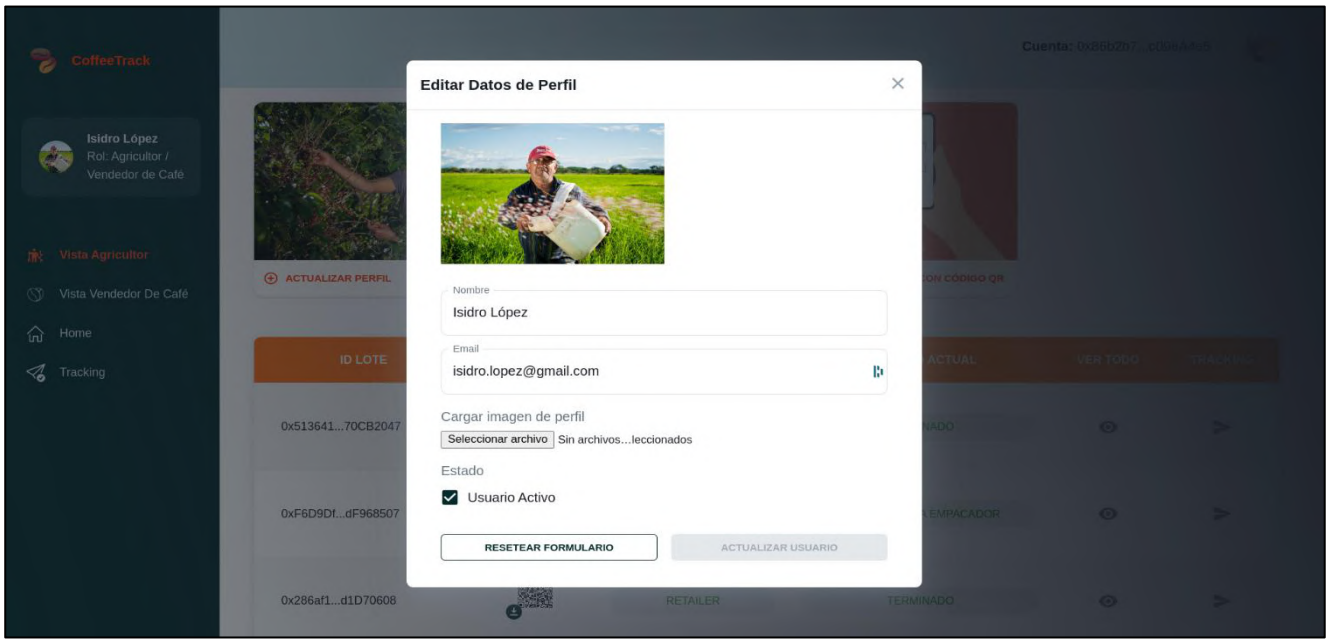


Figura 5.144. Formulario para actualizar perfil del usuario Agricultor

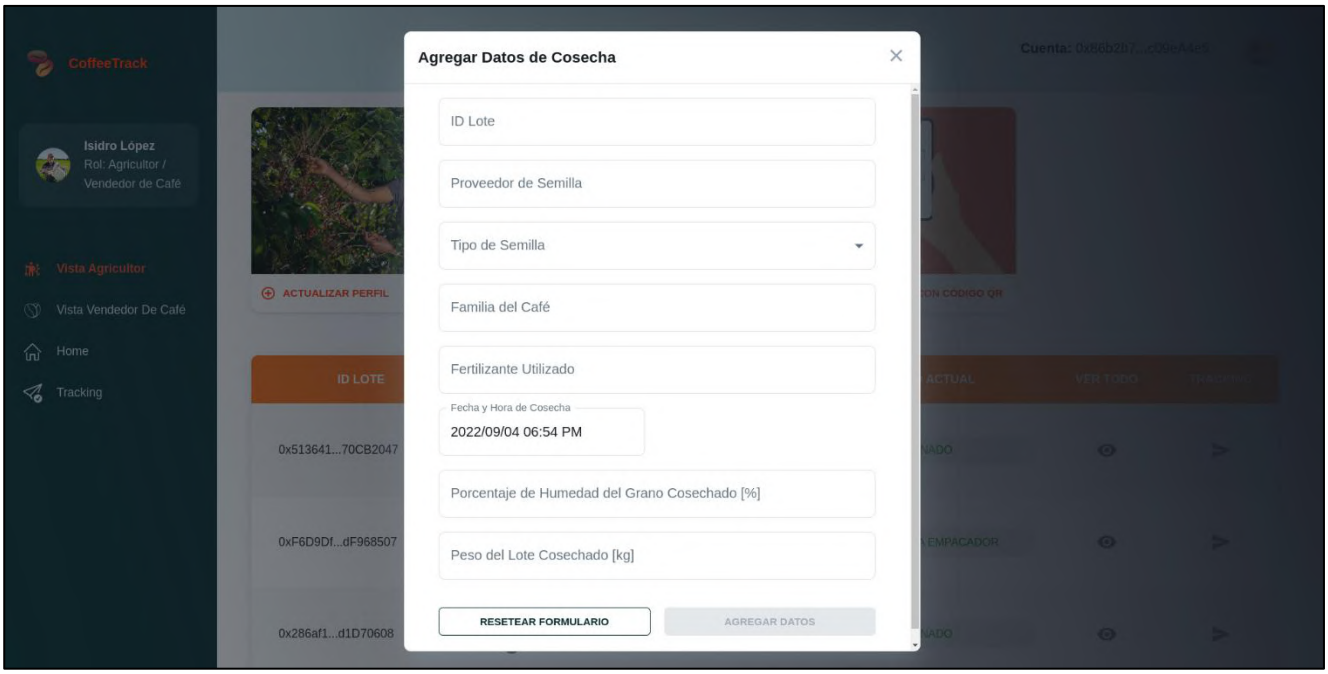


Figura 5.145. Formulario para agregar datos de cosecha

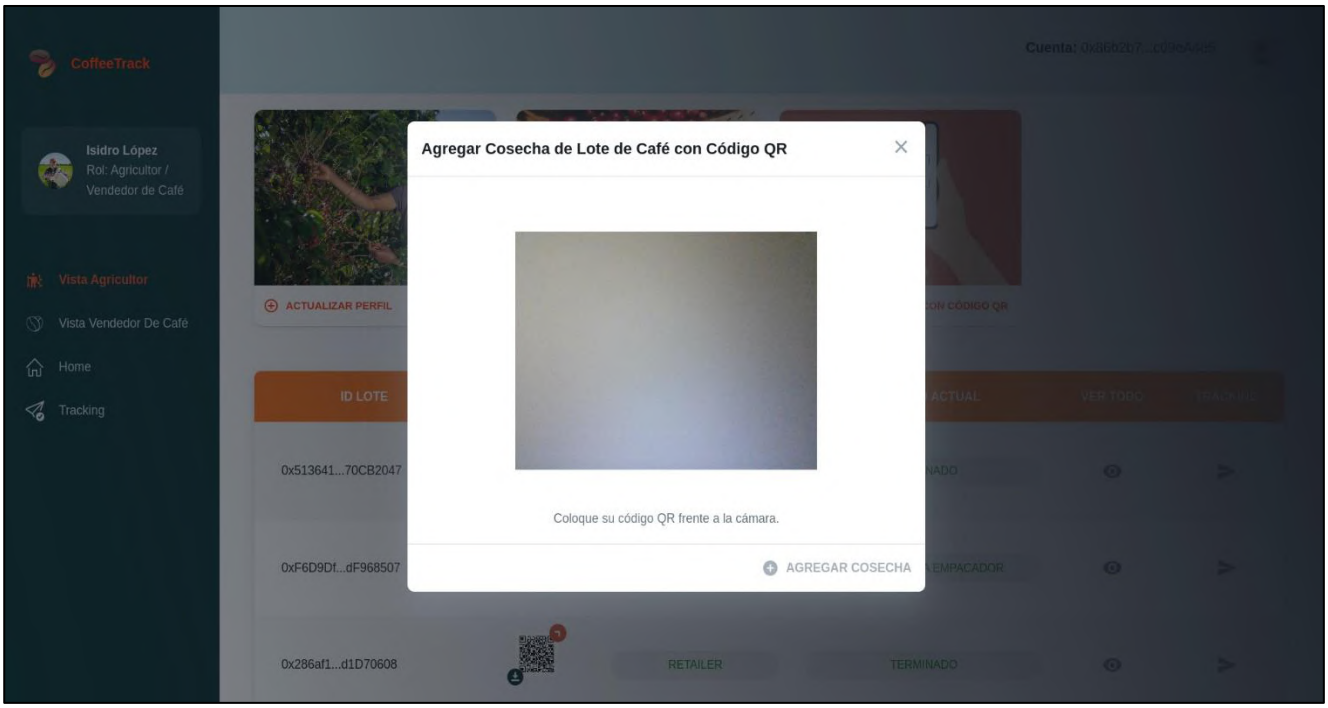


Figura 5.146. Ventana emergente de la opción AGREGAR COSECHA CON CÓDIGO QR

La información de la tabla para la vista del Agricultor y de los demás roles, tiene la misma distribución de la tabla del administrador mostrado en las figuras anteriores. De igual manera, la opción de ACTUALIZAR PERFIL y AGREGAR INTERVENCIÓN CON CÓDIGO QR, son similares para todos los roles modificadores, con diferencia en la información

personalizada mostrada en función de su rol y del botón que habilitan, el cual redirige al formulario respectivo según su intervención.

- **Rol Procesador**

La Figura 5.147 muestra el panel de control del usuario modificador con rol de Procesador. El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR PROCESADO y AGREGAR PROCESADO CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Procesador.

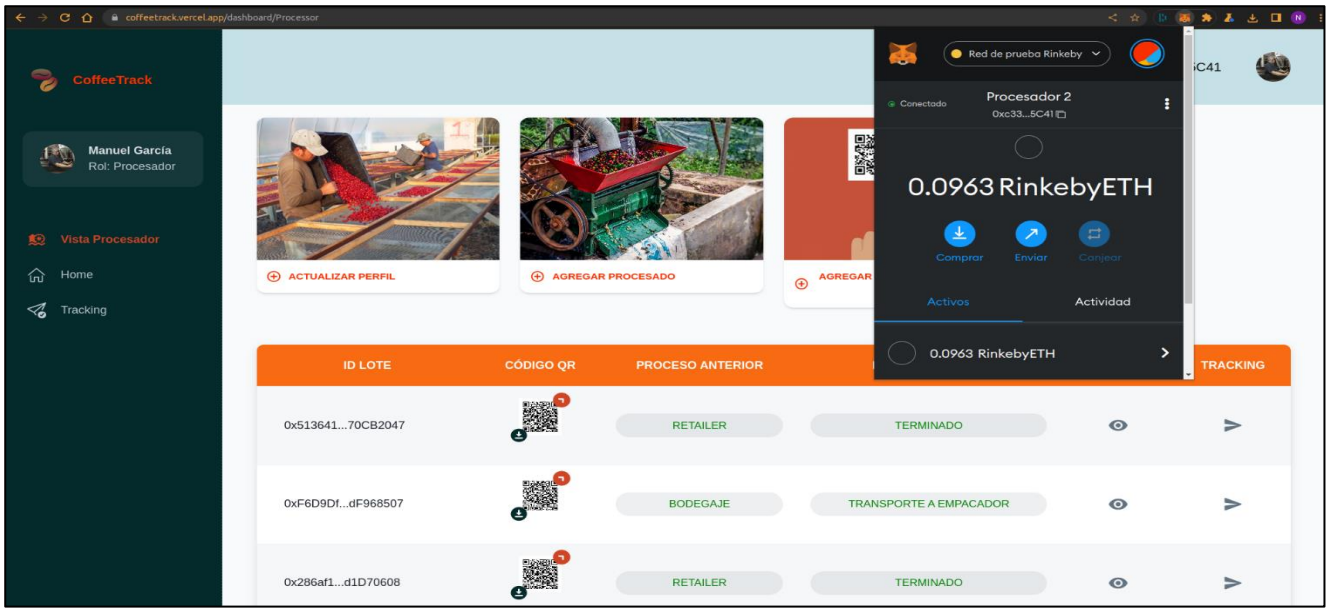


Figura 5.147. Panel de control de un usuario con rol de Procesador

El formulario para agregar procesado se visualiza en la Figura 5.148 y Figura 5.149.

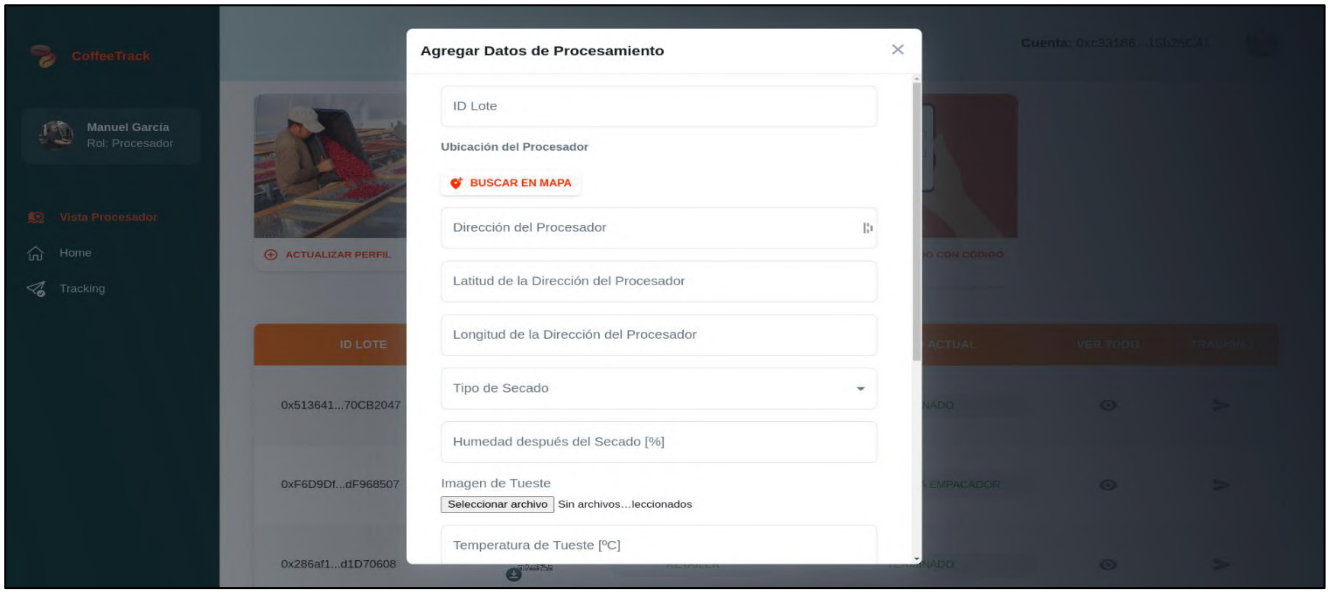


Figura 5.148. Formulario para agregar datos de procesado parte 1

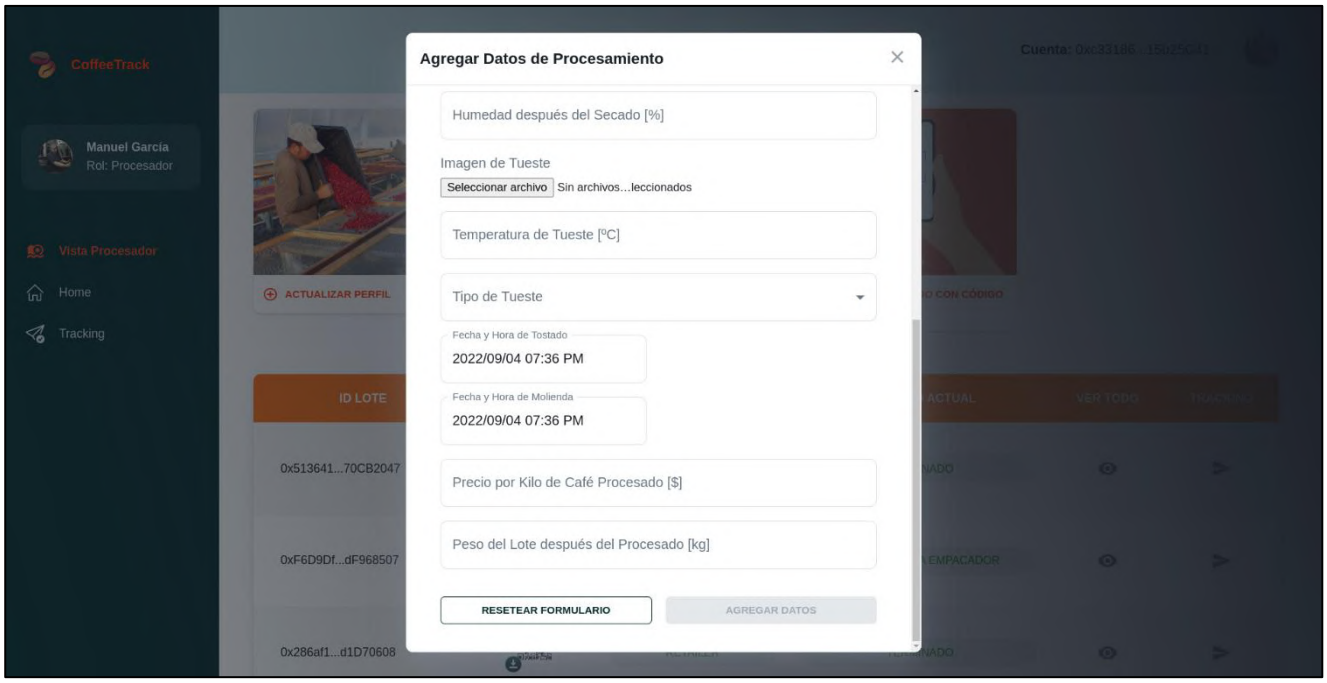


Figura 5.149. Formulario para agregar datos de procesado parte 2

- **Rol Catador**

La Figura 5.150 muestra el panel de control del usuario modificador con rol de Catador. El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR CATACIÓN y AGREGAR CATACIÓN CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Catador.

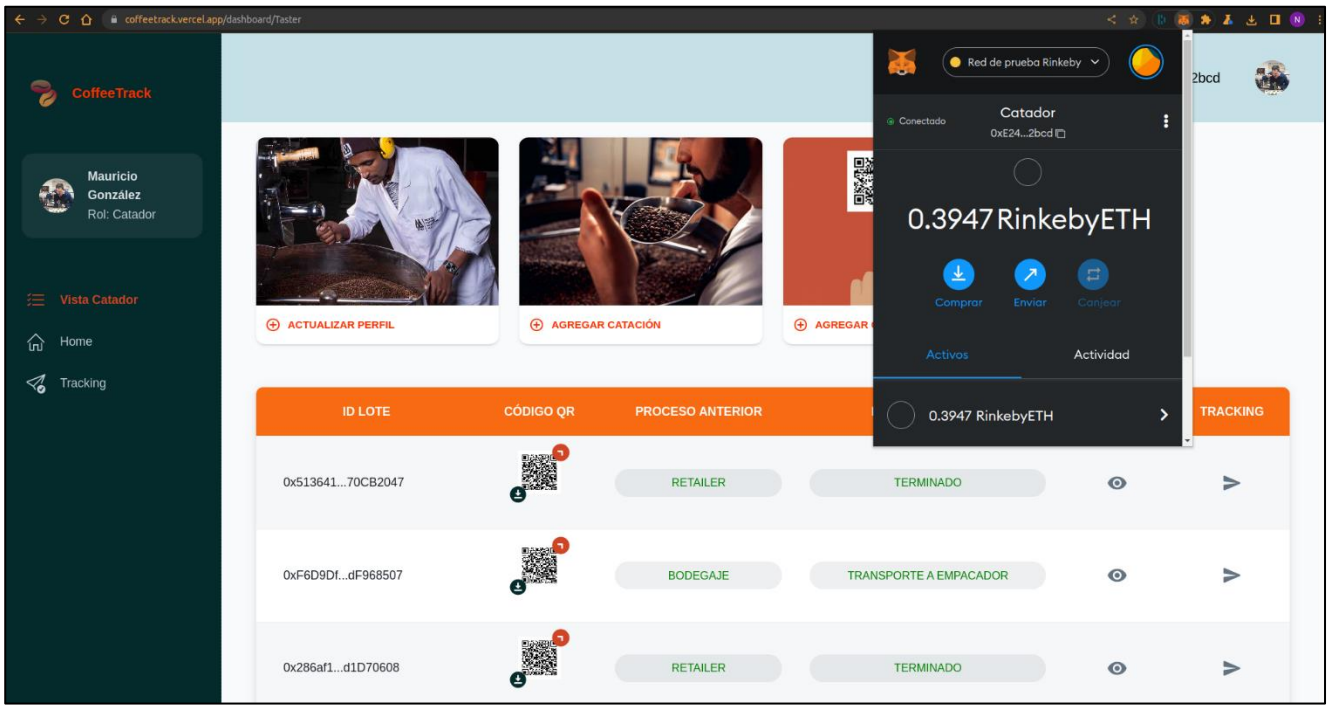


Figura 5.150. Panel de control de un usuario con rol de Catador

El formulario para agregar catación se visualiza en la Figura 5.151.

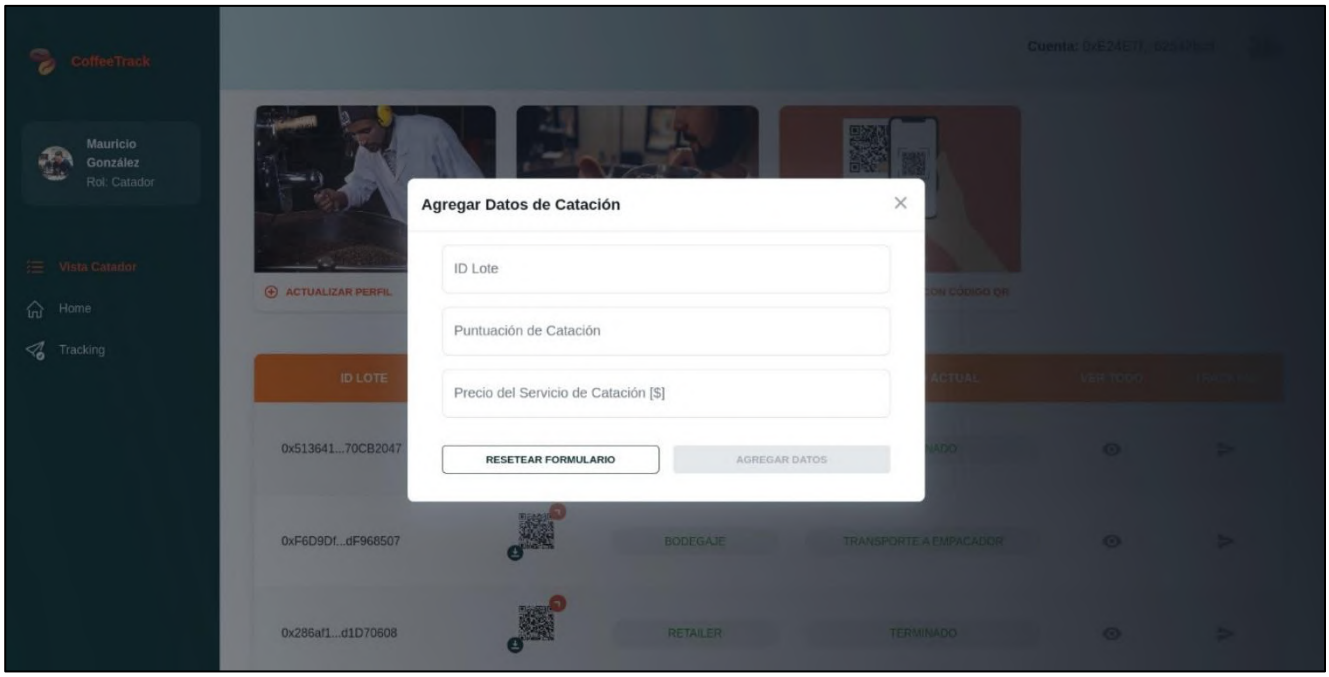


Figura 5.151. Formulario para agregar datos de catación

- **Rol Vendedor de Café**

La vista de Vendedor de Café contiene las opciones de ACTUALIZAR PERFIL, AGREGAR VENTA DE CAFÉ y AGREGAR VENTA DE CAFÉ CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Vendedor de Café.

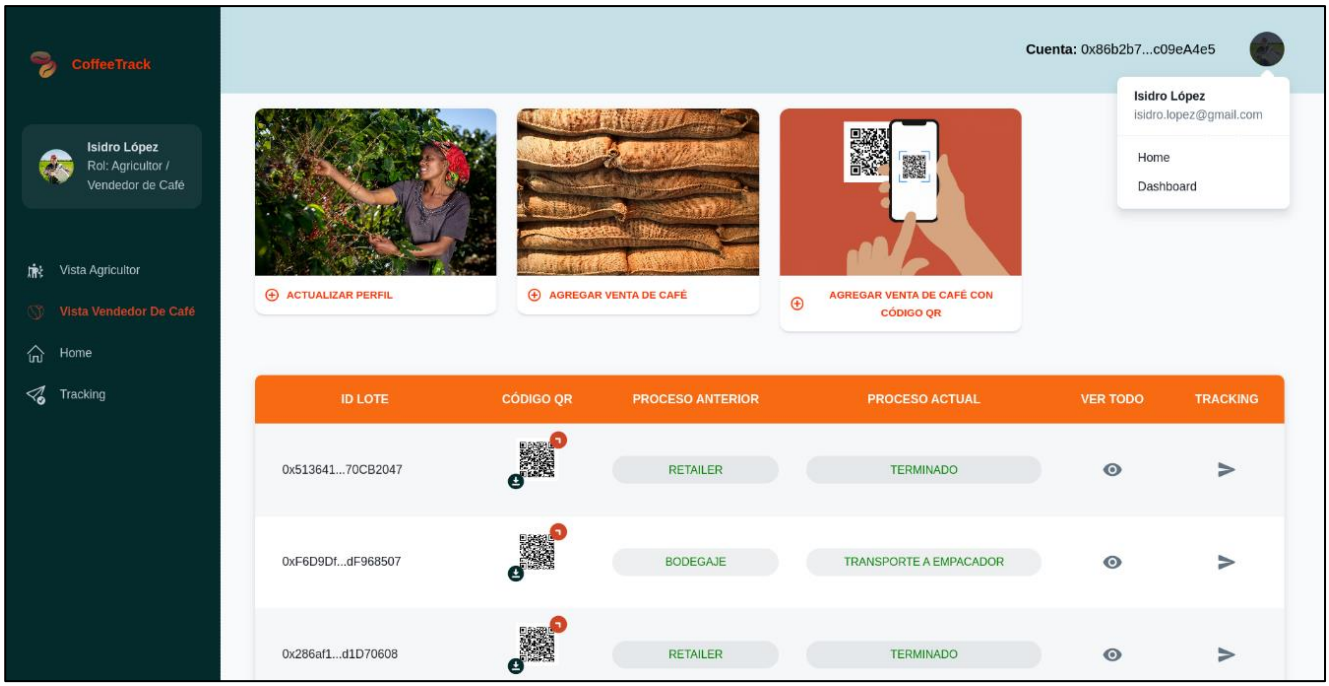


Figura 5.152. Panel de control de un usuario con rol de Vendedor de Café

El formulario para agregar la venta de café se visualiza en la Figura 5.153.

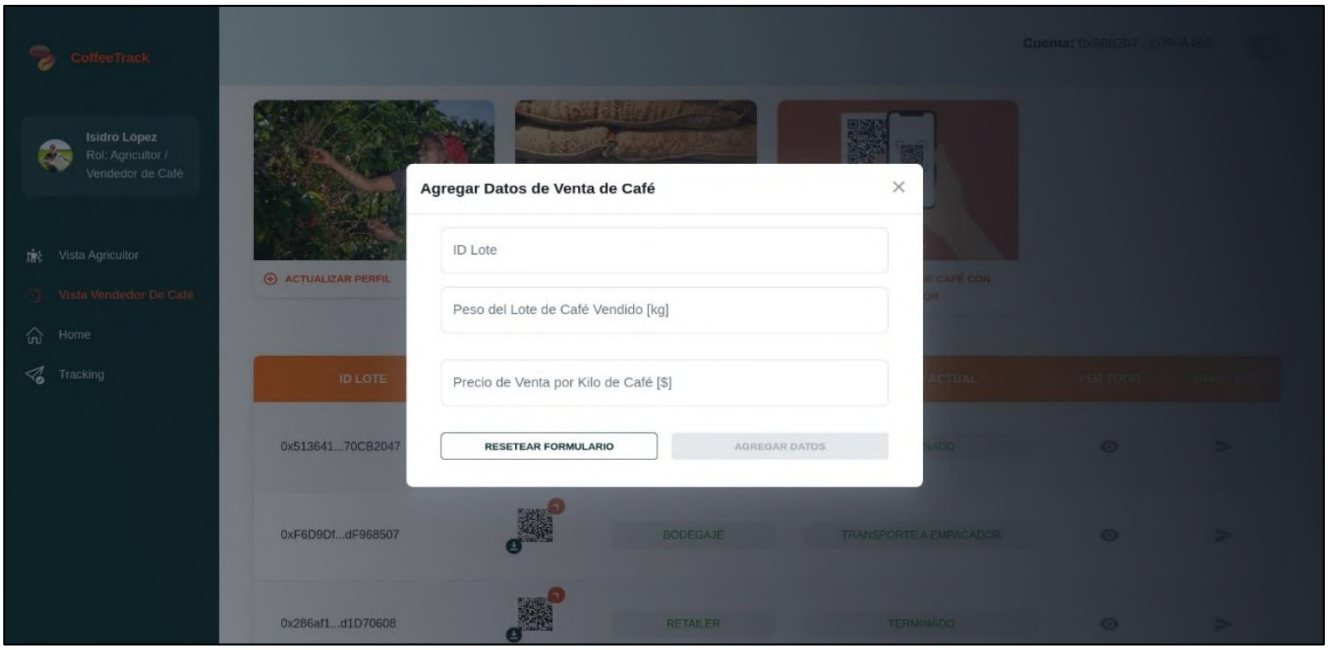


Figura 5.153. Formulario para agregar datos de venta de café

- **Rol Bodega**

La Figura 5.154 muestra el panel de control del usuario modificador con rol de Bodega. El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR BODEGAJE y AGREGAR BODEGAJE CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Bodega.

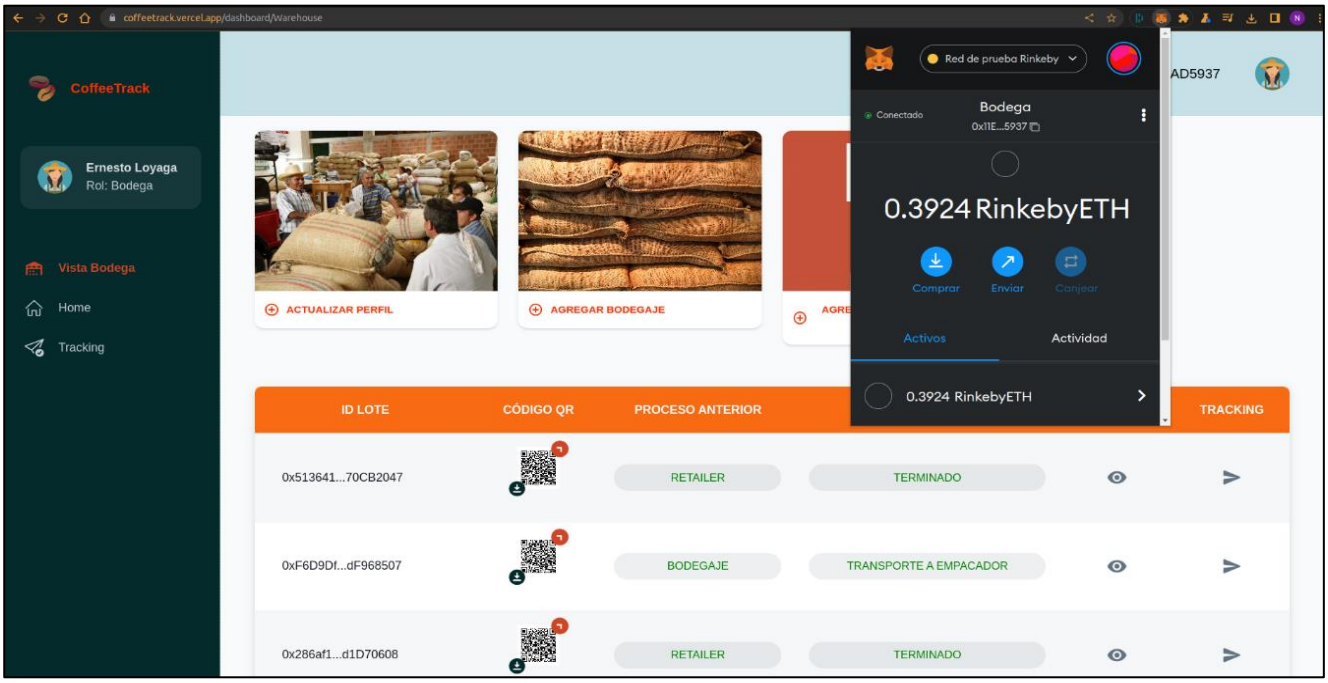


Figura 5.154. Panel de control de un usuario con rol de Bodega

El formulario para agregar bodegaje se visualiza en la Figura 5.155.

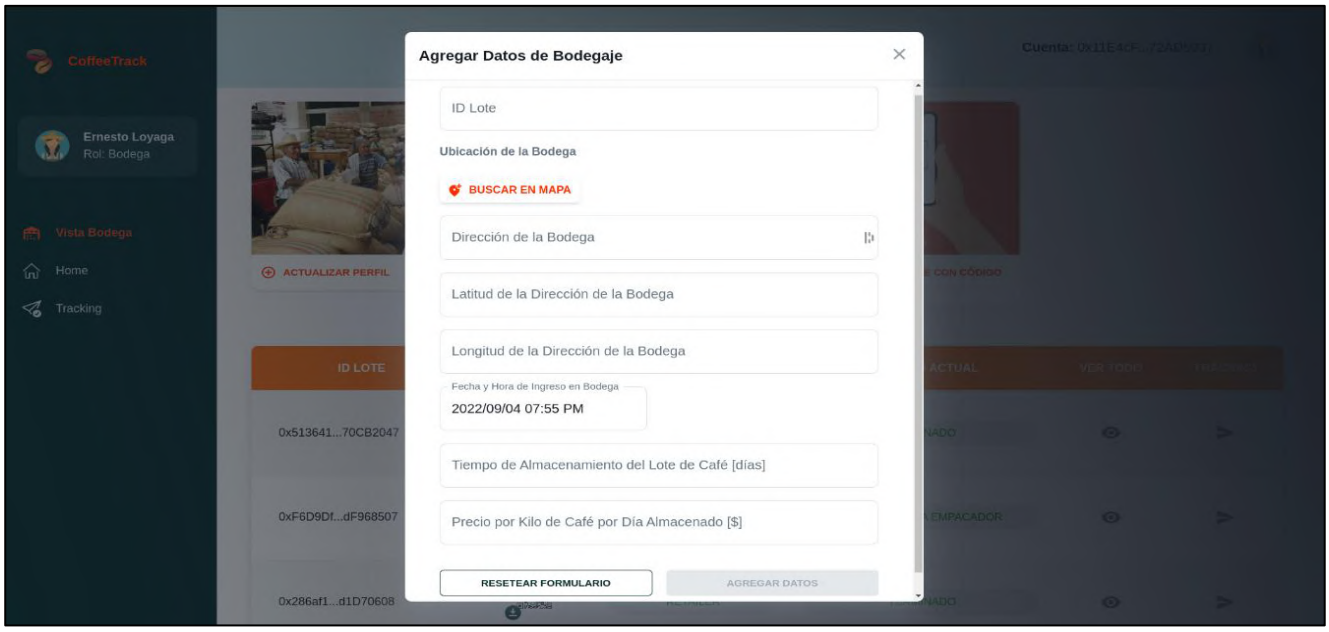


Figura 5.155. Formulario para agregar datos de bodegaje

- **Rol Transportista a Empacador**

La Figura 5.156 muestra el panel de control del usuario modificador con rol de Transportista a Empacador. El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR TRANSPORTE y AGREGAR TRANSPORTE CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Transportista a Empacador.

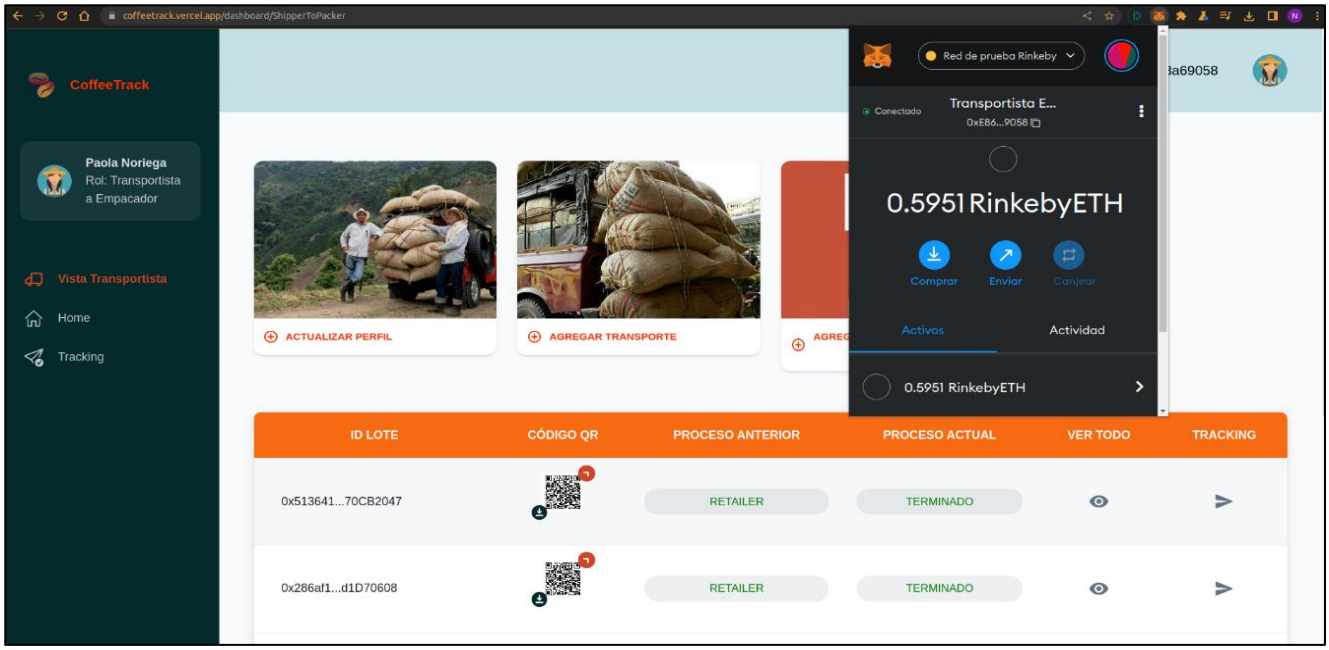


Figura 5.156. Panel de control de un usuario con rol de Transportista a Empacador

El formulario para agregar transporte se visualiza en la Figura 5.157.

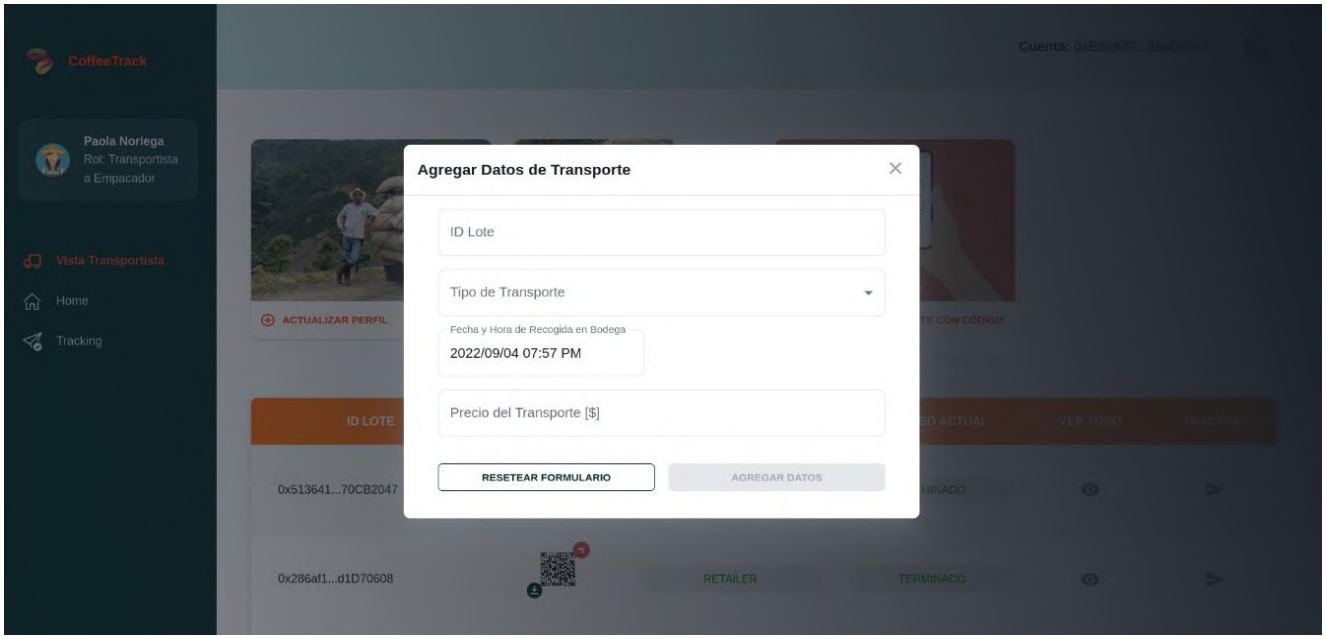


Figura 5.157. Formulario para agregar datos de transporte a empacador

- **Rol Empacador**

La Figura 5.158 muestra el panel de control del usuario modificador con rol de Empacador. El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR EMPACADO y AGREGAR EMPACADO CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Empacador.

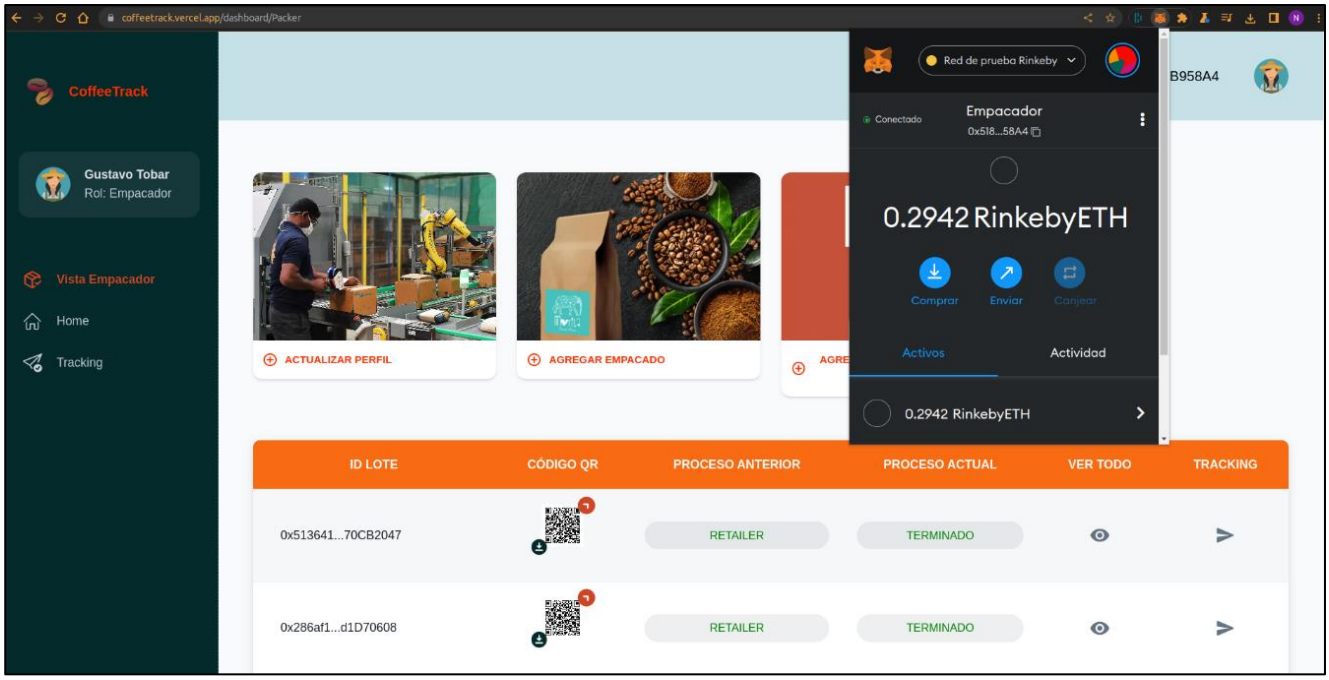


Figura 5.158. Panel de control de un usuario con rol de Empacador

El formulario para agregar empacado se visualiza en la Figura 5.159.

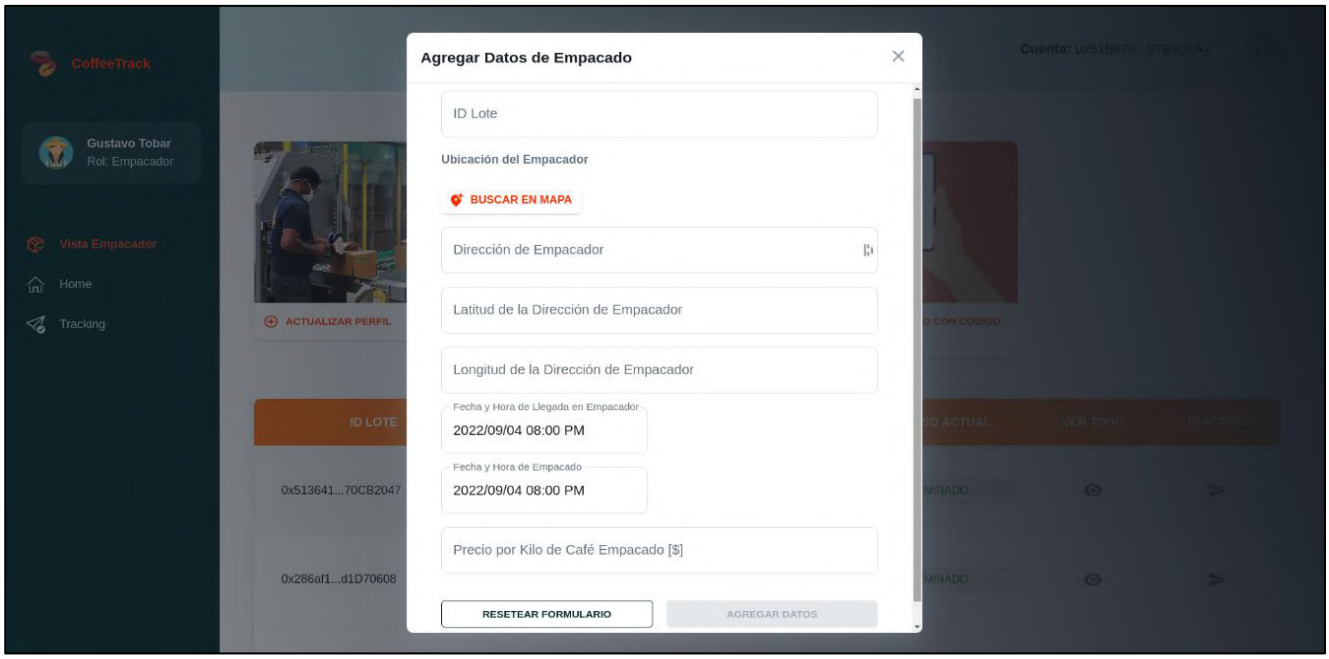


Figura 5.159. Formulario para agregar datos de empacado

- **Rol Transportista a Retailer**

La Figura 5.160 muestra el panel de control del usuario modificador con rol de Transportista a Retailer. El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR TRANSPORTE y AGREGAR TRANSPORTE CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Transportista a Retailer.

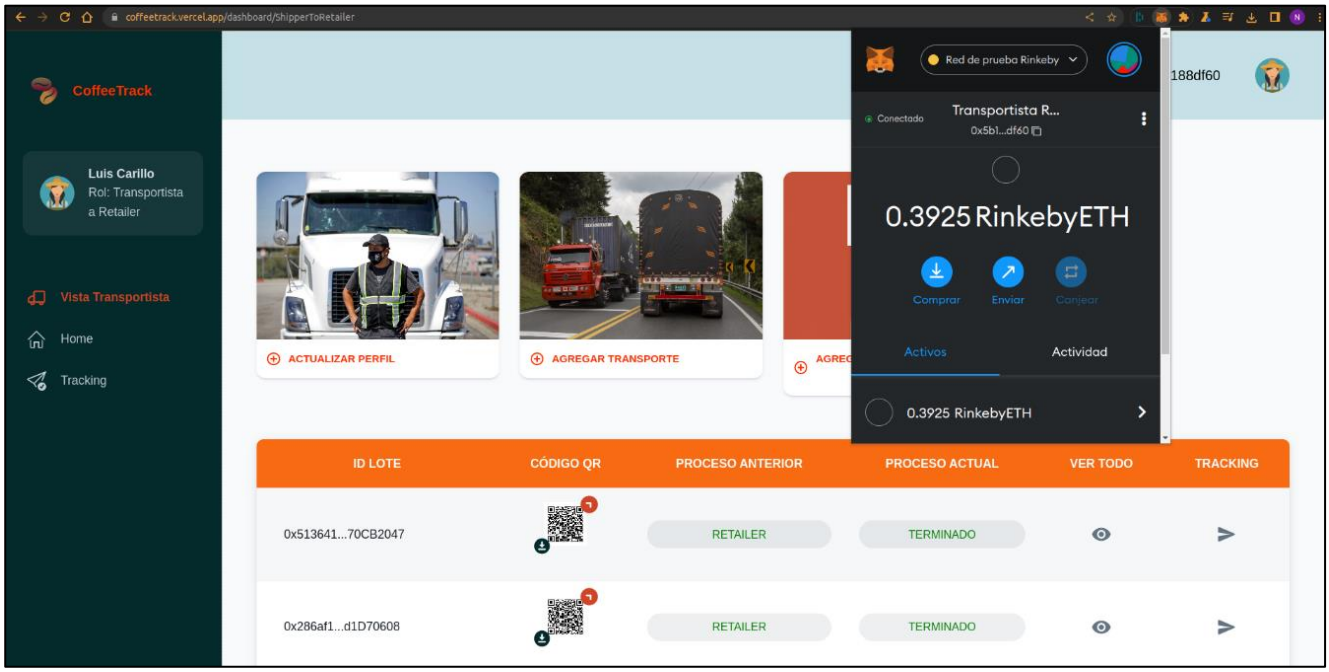


Figura 5.160. Panel de control de un usuario con rol de Transportista a Retailer

El formulario para agregar transporte se visualiza en la Figura 5.161.

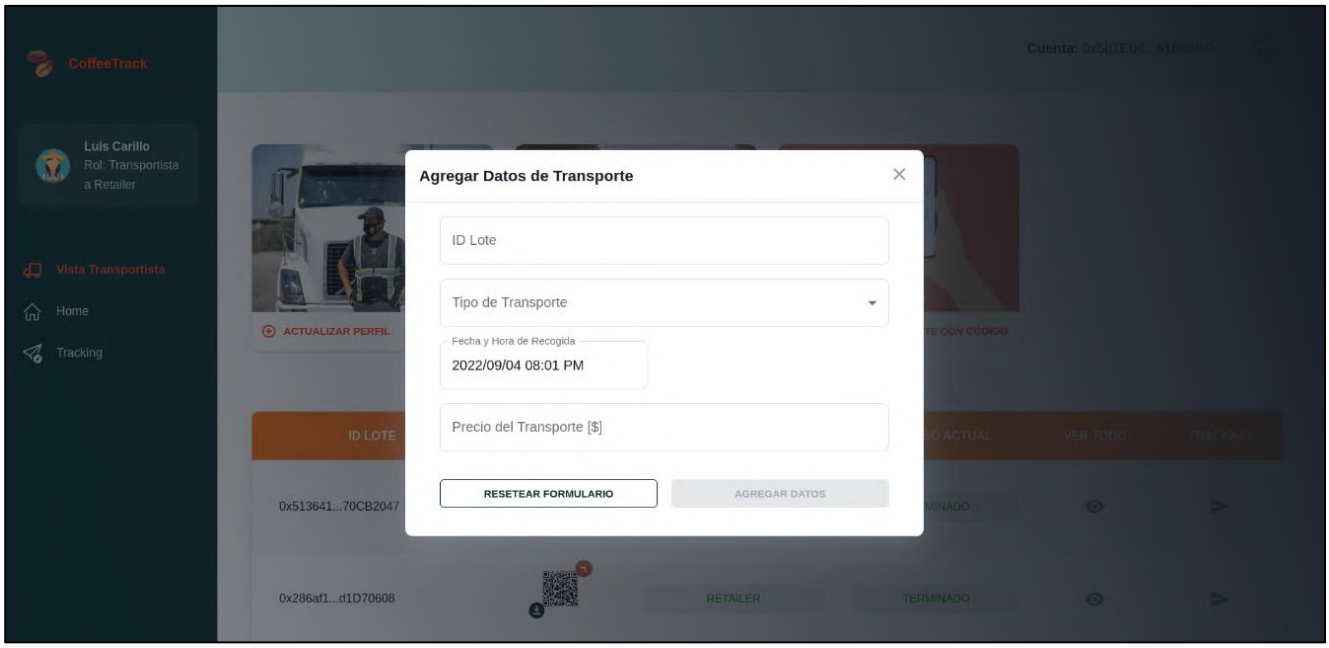


Figura 5.161. Formulario para agregar datos de transporte a retailer

- **Rol Retailer**

La Figura 5.162 muestra el panel de control del usuario modificador con rol de Retailer El panel de control contiene las opciones de ACTUALIZAR PERFIL, AGREGAR RETAILER y AGREGAR RETAILER CON CÓDIGO QR, así como una tabla que muestra todos los lotes intervenidos por este usuario con el rol de Retailer.

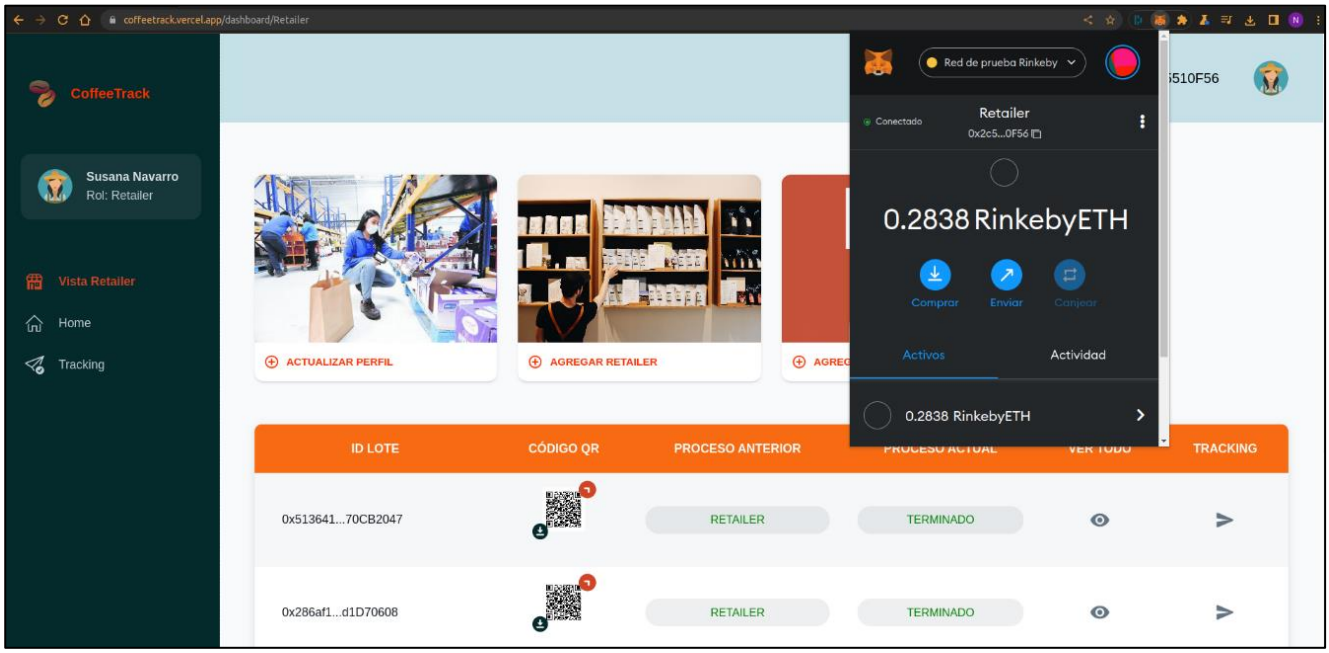


Figura 5.162. Panel de control de un usuario con rol de Retailer

El formulario para agregar retailer se visualiza en la Figura 5.163 y Figura 5.164.

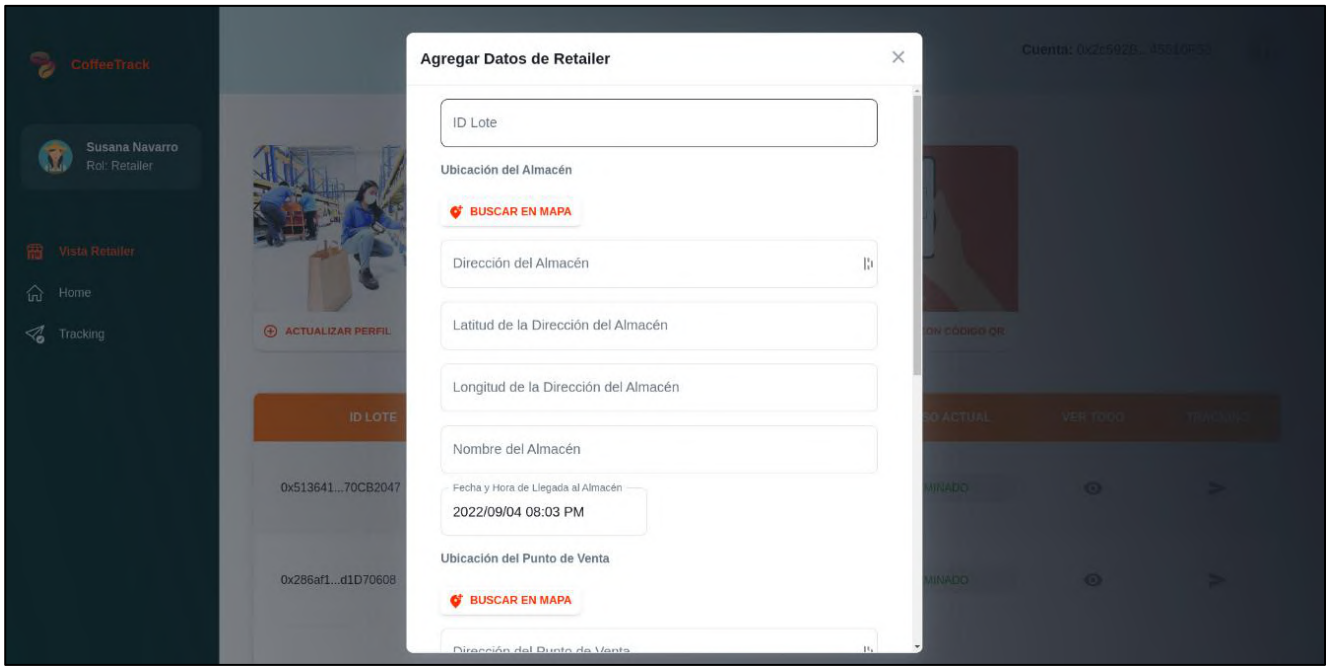


Figura 5.163. Formulario para agregar datos de comercialización en retailer parte 1

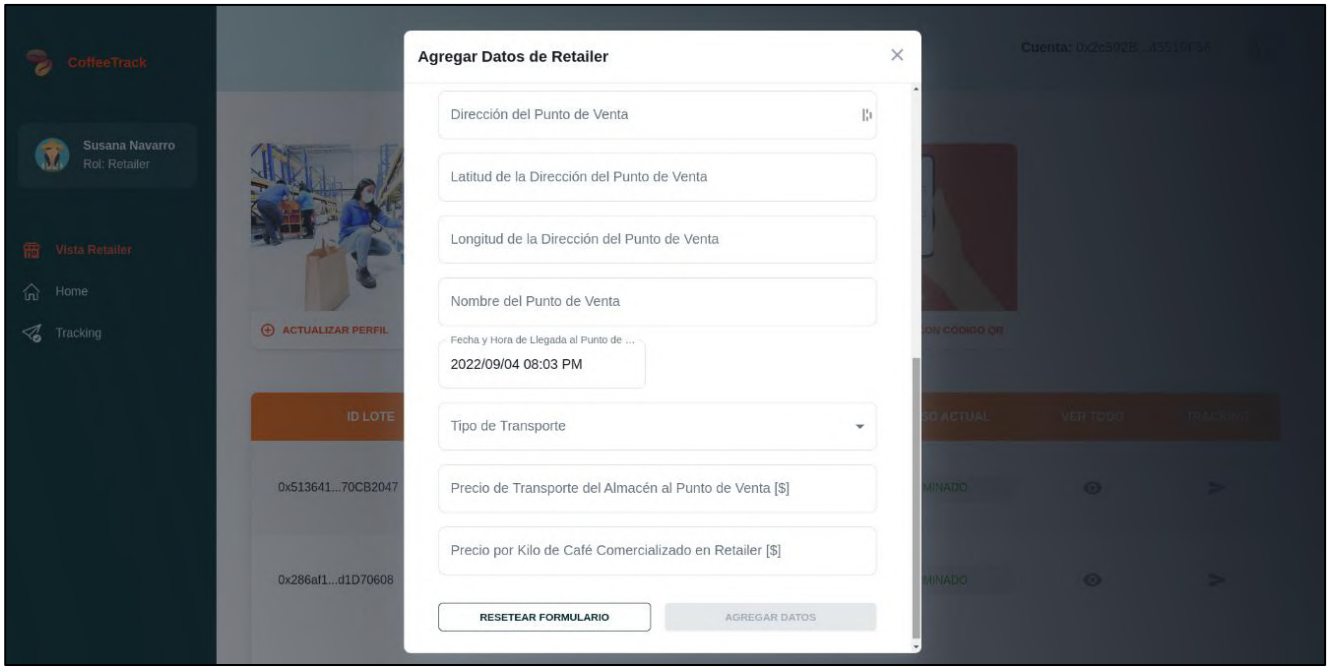


Figura 5.164. Formulario para agregar datos de comercialización en retailer parte 2

Tanto el panel de control del administrador como de los usuarios modificadores permiten redirigirse a la página de Inicio (Home) y de Tracking.

- **Página Tracking**

La Figura 5.165 muestra la página de Tracking con las opciones de LOTE DE EJEMPLO y ESCANEAR QR.

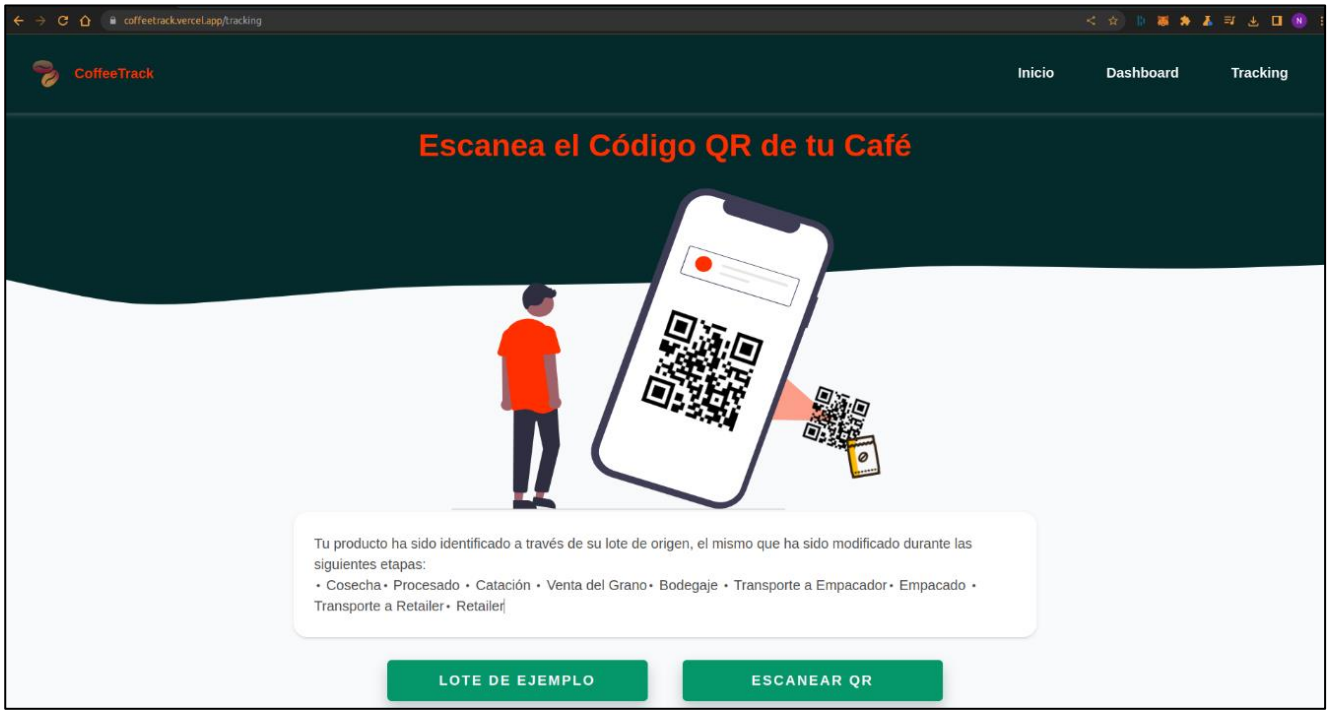


Figura 5.165. Página de Tracking de la aplicación web CoffeeTrack

En las figuras siguientes, se muestra la página de redirección al lote de ejemplo de trazabilidad, al seleccionar la opción LOTE DE EJEMPLO.

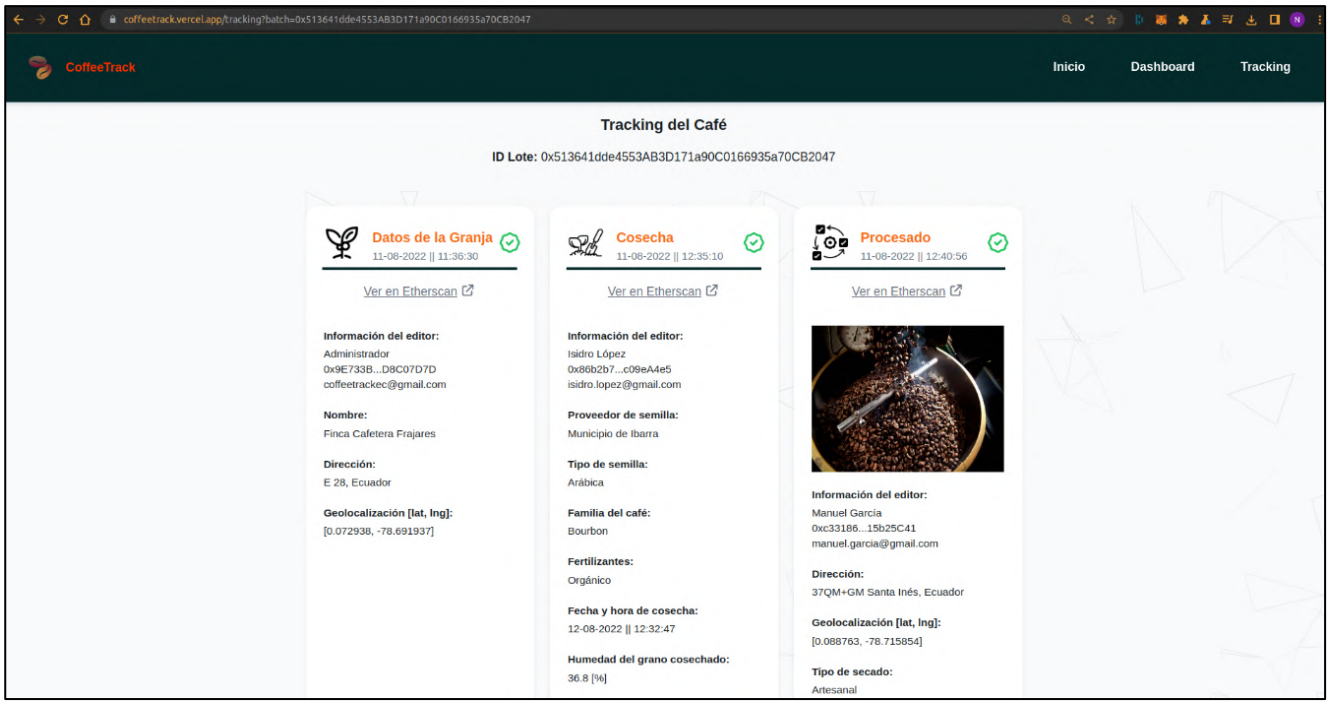


Figura 5.166. Página de Tracking: lote de ejemplo parte 1

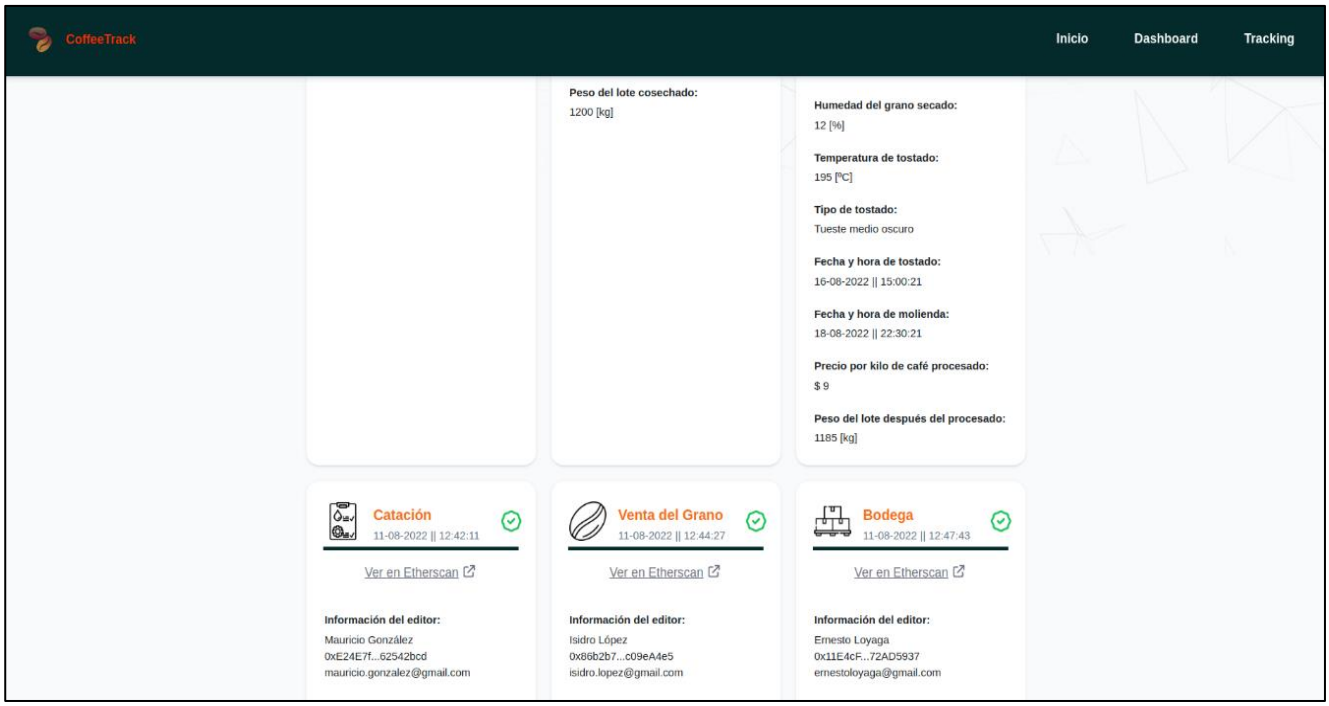


Figura 5.167. Página de Tracking: lote de ejemplo parte 2

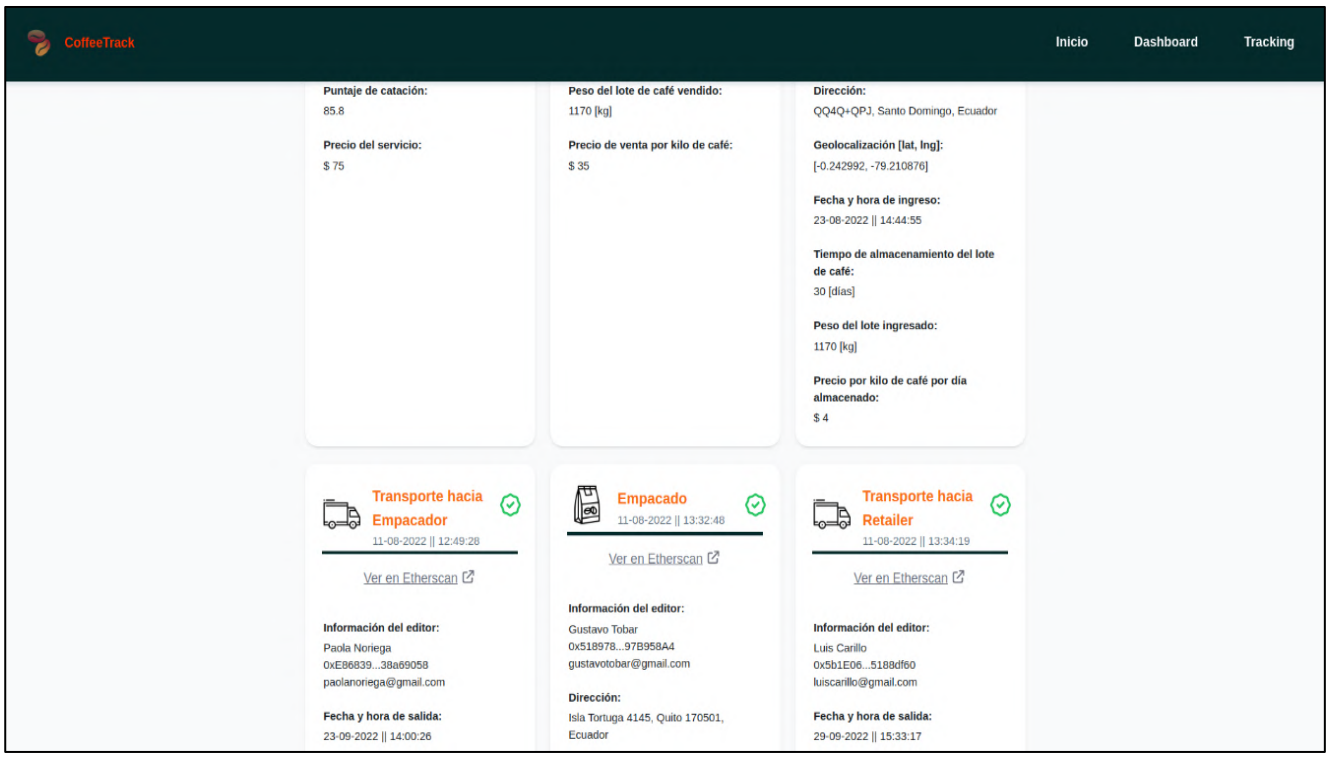


Figura 5.168. Página de Tracking: lote de ejemplo parte 3

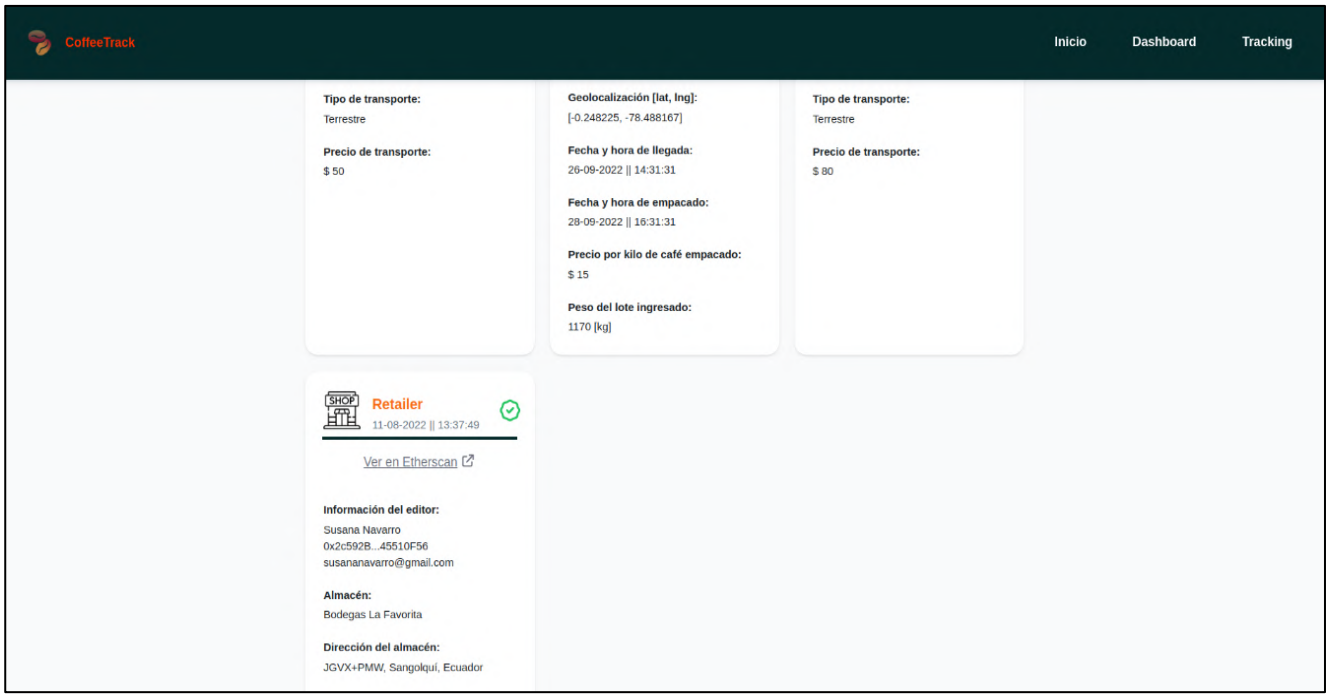


Figura 5.169. Página de Tracking: lote de ejemplo parte 4

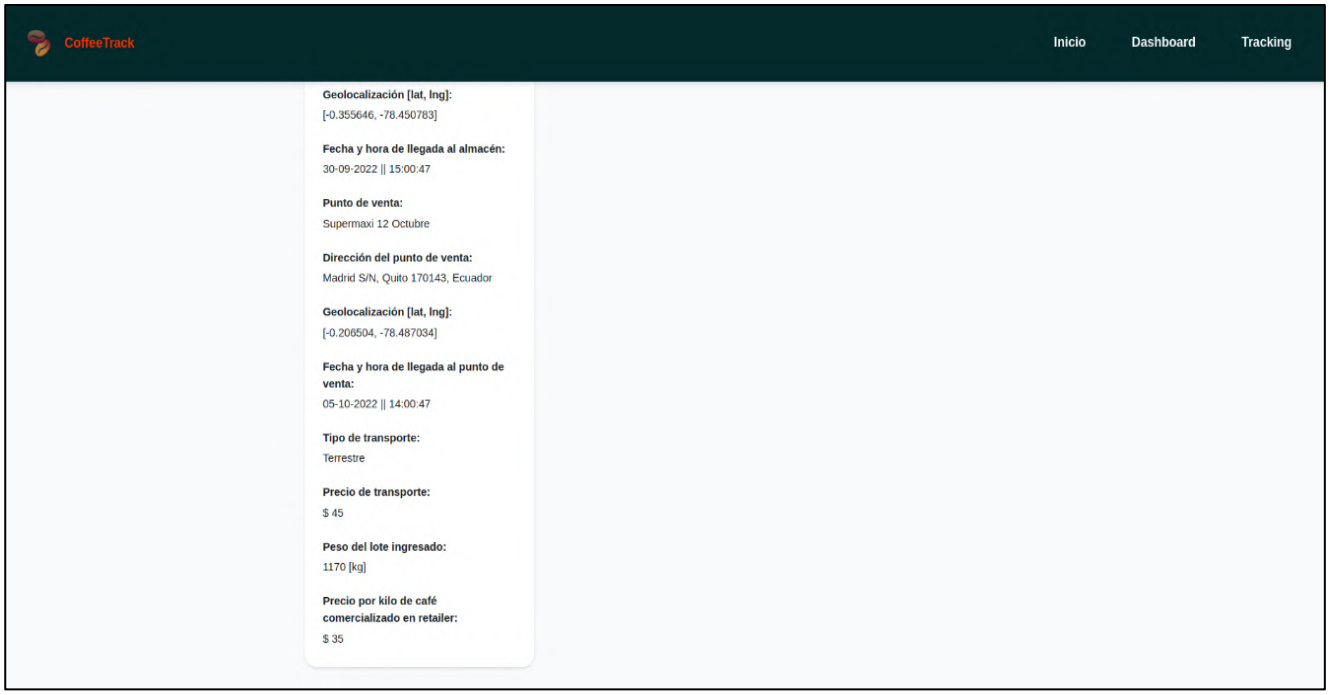


Figura 5.170. Página de Tracking: lote de ejemplo parte 5

Figura 5.171. Página de Tracking: lote de ejemplo parte 6

c) Usuario Consumidor

La opción de ESCANEAR QR es ideal para que los consumidores de café de especialidad puedan consultar el origen y trazabilidad de su café a través del escaneo del código QR presenta en la funda del producto. La Figura 5.172 muestra la ventana emergente de la opción ESCANEAR QR.

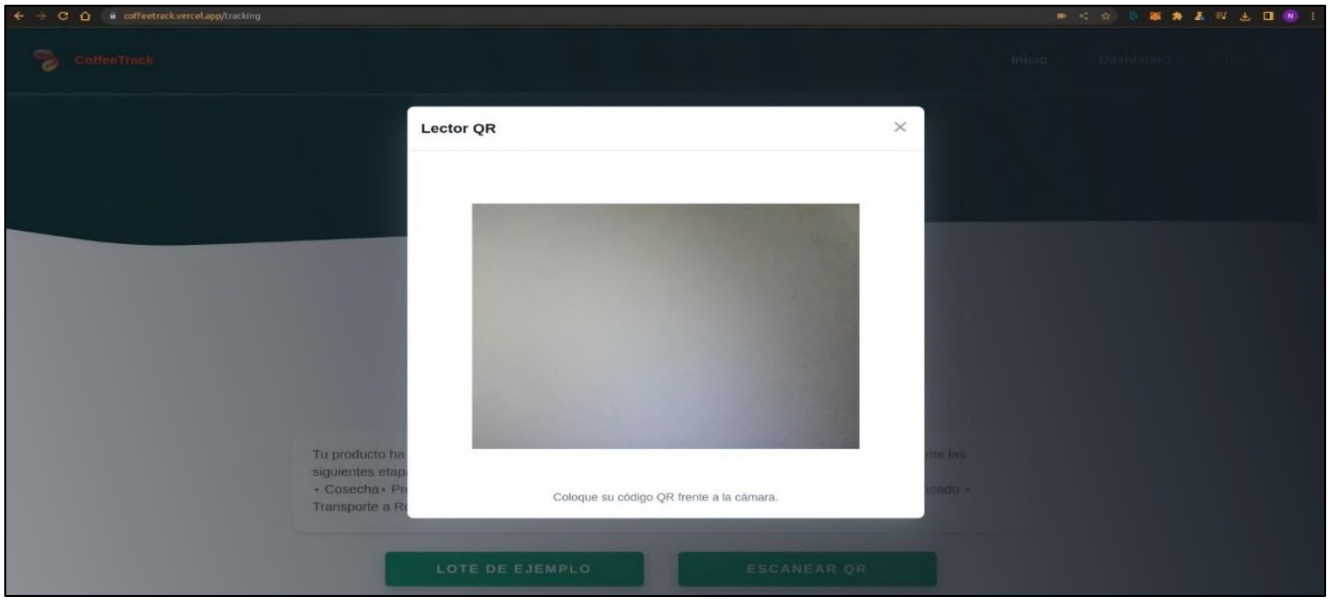


Figura 5.172. Página de Tracking: opción ESCANEAR QR

El funcionamiento de toda la aplicación web puede consultarse en el video demostrativo del Anexo XLI.

ANEXO XLI

Enlaces del video del funcionamiento del sistema de tracking y de los entregables del proyecto.

1. **Video de funcionamiento del Sistema de Tracking:**

https://drive.google.com/drive/folders/1UXhiziLAlAYdVubPzxdyva1ySeEvFsaw?usp=sharing

2. **Entregables:**

   a) **Contratos inteligentes desplegados en la testnet Rinkeby para ser consultados en el explorador Etherscan**

   - Contrato SupplyChainStorage.sol:

https://rinkeby.etherscan.io/address/0x8d634bf42f06d7904dd46d8ed4ab345230a03b7d#code

   - Contrato SupplyChainStorage2.sol:

https://rinkeby.etherscan.io/address/0x1e3DDD059f0F01258A7baC74dfBA2cA43C33A449#code

   - Contrato CoffeeSupplyChain.sol:

https://rinkeby.etherscan.io/address/0xdf0C594655C466B0b37CeFc519f38Ea8fEB465F9#code

   - Contrato CoffeeSupplyChain2.sol:

https://rinkeby.etherscan.io/address/0xcf76465C29A32F11D6A27a009eE7CB500669c5Ff#code

   - Contrato SupplyChainUser.sol:

https://rinkeby.etherscan.io/address/0xbf87Fd7e3416311dbef4F00e2ce73950A0F2a0D2#code

   b) **Repositorio de GitHub del código fuente de los contratos inteligentes**

https://github.com/NathaliaBarreiros/coffee-supply-chain-tracking-system-ethereum

   c) **Repositorio de GitHub del código fuente de la aplicación web CoffeeTrack**

https://github.com/NathaliaBarreiros/coffee-supply-chain-client

d) Aplicación web CoffeeTrack

https://coffeetrack.vercel.app/home