Desarrollo de sistema de trazabilidad alimentaria confiable basado en Blockchain

Trabajo de Fin de Máster

MÁSTER EN INGENIERÍA INFORMÁTICA (Semi-presencial)

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autor

Konstantin Danielov Kostandev

Tutor

Vidal Moreno Rodilla

Autorización del tutor

Dr. Vidal Moreno Rodilla, profesor del Departamento de Informática y Automática de la Universidad de Salamanca.

CERTIFICA:

Que el trabajo titulado "**Desarrollo de sistema de trazabilidad alimentaria confiable basado en Blockchain**" ha sido realizado por Konstantin Danielov Kostandev, con NIE X7700897Z y constituye la Memoria para la superación de la asignatura Trabajo Fin de Máster de la titulación de Máster en Ingeniería Informática (modalidad semi-presencial) de esta Universidad.

Y para que conste a los efectos oportunos, firma el presente documento, en: Salamanca, a 2 de septiembre de 2022.

MORENO RODILLA VIDAL - 07870365H

Firmado digitalmente por MORENO RODILLA VIDAL - 07870365H
Fecha: 2022.09.02 21:43:19 +02'00'

Fdo.- Vidal Moreno Rodilla

Resumen

La trazabilidad de los alimentos se entiende como la capacidad de rastrear el ciclo de vida de un alimento desde su origen hasta su llegada al consumidor final, siendo un requisito fundamental para cumplir los estándares de calidad de los productos y garantizar la seguridad de los consumidores. Para conseguir este propósito se requiere almacenar gran cantidad de información relativa a cada uno de los procesos por los que pasa el alimento, garantizando su seguridad, fiabilidad y disponibilidad, pudiendo convertirse en una tarea complicada a la hora de llevarse a cabo. En el presente trabajo de fin de máster se propone el desarrollo de un sistema de trazabilidad confiable basado en blockchain, una innovadora tecnología que ofrece una manera de garantizar la seguridad, disponibilidad e inmutabilidad de los datos. Como ejemplo práctico se desarrollará un prototipo que permita realizar la trazabilidad del ciclo de vida del jamón ibérico, un producto en el que la trazabilidad es fundamental para garantizar y certificar su calidad. El objetivo de la creación de este prototipo es demostrar la viabilidad del uso de la tecnología blockchain para este propósito, cubriendo gran parte del ciclo de vida del alimento, pudiendo servir como base para futuros desarrollos con esta tecnología.

Palabras clave: Trazabilidad alimentaria, blockchain, ethereum

Summary

Food traceability make it possible to trace the path that a food article has followed from its source to the final consumer, being a fundamental requirement to meet product quality standards and guarantee consumers safety. It will be needed to store a large amount of data related to each process that the food article passes by to achieve this purpose, guaranteeing that the stored data is safe, reliable and available. In this master's thesis, the development of a reliable traceability system based on blockchain is proposed: an innovative technology that offers a way to guarantee the security, availability and immutability of data. As a practical example, a prototype will be developed to trace the life cycle of Iberian Ham, where having certified quality is a must. The purpose of the prototype is to demonstrate the viability of using blockchain technology for these systems, covering a large part of the food life cycle, and being able to serve as a basis for future developments with this technology.

Keywords: Food traceability, blockchain, ethereum

Tabla de contenido

Tabla de Ilustraciones

1 Introducción

Con los avances en la tecnología, hoy en día es posible almacenar de manera informática toda la información relevante de un proceso de manera sencilla, permitiendo su posterior manejo para lograr un propósito concreto, siendo muy interesante su aplicación en sectores como la alimentación. En este sector la ley obliga a mantener un sistema de trazabilidad, cuyos principios y requisitos generales se recogen en el artículo 18 del reglamento 178/2002 del Parlamento Europeo y del Consejo del 28 de enero de 2002 [1] [2]. Con unas necesidades de tal calibre, el sistema informático encargado del almacenaje y la gestión de los datos debe ser seguro, fiable e inmutable, asegurando que los datos introducidos no puedan ser alterados por ninguno de los participantes de la cadena. Para satisfacer estas necesidades se propone la utilización de una red blockchain para el almacenamiento y gestión de los datos.

Blockchain es una tecnología novedosa que ha supuesto una gran revolución, ya que permite realizar transacciones seguras entre personas de todo el mundo sin necesidad de intermediarios a través de los contratos inteligentes, pudiéndose definir como "un libro mayor de acontecimientos digitales que se comparte entre diferentes partes" [3]. La primera de estas redes blockchain en hacerse conocida y llegar a un gran público fue Bitcoin, que surgió debido a la necesidad de realizar transacciones monetarias de manera segura y sin intermediarios, donde se intercambian activos digitales llamados criptomonedas. En el caso de Bitcoin, su activo digital o criptomoneda recibe el mismo nombre que la red blockchain: Bitcoin. El funcionamiento de esta red se recoge en un documento publicado en 2008 por Satoshi Nakamoto: "A Peer-to-Peer Electronic Cash System" [4]. Posteriormente comenzaron a surgir otras redes blockchain, como Ethereum: una blockchain de propósito general (a diferencia de Bitcoin que únicamente está enfocada a las operaciones monetarias), que dota a los desarrolladores de herramientas para crear aplicaciones sobre su red, contando con todas ventajas que proporciona. En los apartados siguientes se verá más en detalle Ethereum, ya que es la blockchain elegida para el desarrollo de este trabajo debido a su alcance, fiabilidad, comunidad de desarrolladores y herramientas de desarrollo [5].

Como caso de estudio en este trabajo se desarrollará un prototipo de aplicación de trazabilidad alimentaria confiable basado en blockchain que permita realizar la trazabilidad de un producto concreto: el jamón ibérico. Un sistema de trazabilidad seguro, fiable e inmutable es esencial para productos como el seleccionado, ya que la capacidad de certificar y garantizar la calidad de la pieza se vuelve parte de la marca y tiene un impacto directo en la elección de los consumidores.

A la hora de crear el sistema de trazabilidad no se perseguirá que se asemeje a un sistema real en cuanto cantidad o tipo de datos, ya que se pueden almacenar miles de datos en cada proceso involucrado y para cada tipo de proceso y/o empresa será relevante almacenar datos diferentes. Además, no se cubrirá el ciclo de vida completo del producto sino únicamente la parte de creación y transformación, dejando de lado la distribución, que se podría desarrollar de manera similar a las anteriores. Lo que se persigue es conseguir un sistema funcional (con unos datos básicos, que posteriormente se pueda escalar según las necesidades) conectado a la red blockchain para comprobar las ventajas e inconvenientes de la implementación para este propósito.

2 Definición de objetivos

2.1 Objetivos generales

Los objetivos que se buscan completar para la finalización de este trabajo son los siguientes:

- Modelar un sistema de trazabilidad enfocado en un producto concreto: el jamón ibérico. El sistema deberá permitir tanto guardar como recuperar los datos relevantes de los procesos involucrados, a través de contratos inteligentes en ejecución en una red blockchain.
- Desarrollar el software necesario para la interacción con los contratos inteligentes que se ejecutan en la blockchain.
- Generar un prototipo funcional del sistema de trazabilidad.
- Extraer conclusiones sobre la viabilidad y aplicación de la tecnología blockchain para este propósito.

2.2 Objetivos personales

Para completar estos objetivos generales, se deberán cumplir los siguientes objetivos personales:

- Adquirir un mayor grado de conocimiento sobre: qué es una red blockchain, el funcionamiento a bajo nivel de este tipo de redes y las características que hacen que sean tan interesantes.
- Aprender a usar las herramientas y lenguajes que están destacando actualmente en el mundo del desarrollo blockchain.
- Avanzar en el uso de nuevas herramientas para la creación de una aplicación web conectada a un servidor que se comunique con la red blockchain.

3 Conceptos teóricos

En esta sección se van a tratar los conceptos teóricos más relevantes para el desarrollo de este trabajo.

3.1 Trazabilidad alimentaria

Según La Agencia Española de Seguridad Alimentaria y Nutrición (AESAN) la trazabilidad alimentaria se definiría como "*la posibilidad de encontrar y seguir el rastro, a través de todas las etapas de producción, transformación y distribución, de un alimento, un pienso, un animal destinado a la producción de alimentos o una sustancia destinados a ser incorporados en alimentos o piensos o con probabilidad de serlo*". Este concepto lleva inherente la necesidad de poder identificar cualquier producto dentro de la empresa, desde la adquisición de las materias primas o mercancías de entrada, a lo largo de las actividades de producción, transformación y/o distribución que desarrolle, hasta el momento en que el operador realice su entrega al siguiente eslabón de la cadena [2].

El procedimiento o sistema de trazabilidad que se adopte dentro de cada empresa deberá tener en cuenta [2]:

1. La **identificación del producto**: Un medio único, lo más sencillo posible, para identificar un producto o agrupación de productos.
2. Los **datos del producto**:
    a. Las materias primas, partes constituyentes del producto o mercancías que entran en cada empresa.
    b. La manera en que fue manejado, producido, transformado y presentado, en caso de existir tales procesos.
    c. Su procedencia y destino, así como las fechas de ambos (una etapa antes y una etapa después).
    d. Los controles de que ha sido objeto, en su caso, y sus resultados.

La aplicación de un sistema de trazabilidad presenta amplias ventajas tanto para el operador económico como para los consumidores.

Para las **empresas** se traduce en un **aumento de la seguridad y de los beneficios económicos**, donde el sistema cumple diversas funciones, entre las que se pueden destacar [2]:

- Servir de instrumento para lograr un nivel elevado de protección de la vida y la salud de las personas.
- Proporcionar información dentro de la empresa para facilitar el control de procesos y la gestión (por ejemplo, el control de *stocks*).
- Contribuir al aseguramiento de la calidad y la certificación del producto.
- Servir de apoyo cuando se produzca algún incidente, facilitando la localización y la gestión de los alimentos.

Para el **consumidor** se traduce en un **aumento de la confianza** debido a los siguientes motivos [2]:

- El sistema da certeza de que los productos se producen con la conveniente transparencia informativa a lo largo de toda la cadena, desde el productor al consumidor, evitando las prácticas fraudulentas, la adulteración de los alimentos o cualquier otra práctica que pueda conducir a engaño al consumidor.
- Con la aplicación del sistema el consumidor tiene la garantía de que ante cualquier problema las acciones a tomar se realizarán con la máxima eficacia, rapidez y coordinación.

El modelo que se plantea en el presente trabajo de fin de máster consiste en un sistema de trazabilidad alimentaria en el que cada operador económico, de manera independiente y sin intermediarios, pueda almacenar los datos relevantes que necesite asegurando que no puedan ser modificados de ninguna manera por otros eslabones de la cadena de producción.

3.1.1 Proceso de elaboración del jamón

El objeto de estudio de este trabajo será la elaboración del jamón ibérico, por lo que es necesario definir cuáles son los procesos involucrados sobre los que se quiere almacenar información.

Estos procesos pueden variar en función de la metodología de cada empresa, por lo que se va a tener en cuenta un modelo básico formado por dos bloques:

- **Tratamiento de la materia prima**: En este bloque se almacenará toda la información de los procesos que tienen relación con el ganado y su transporte. Tipos de proceso de este bloque:
  - Venta de Ganado
  - Transporte de Ganado
- **Obtención y tratamiento de la pieza**: En este bloque se almacenará toda la información de los procesos a los que se somete cada una de las piezas. Tipos de proceso de este bloque:
  - Obtención de la pieza (sacrificio)
  - Procesos de transformación
  - Transporte

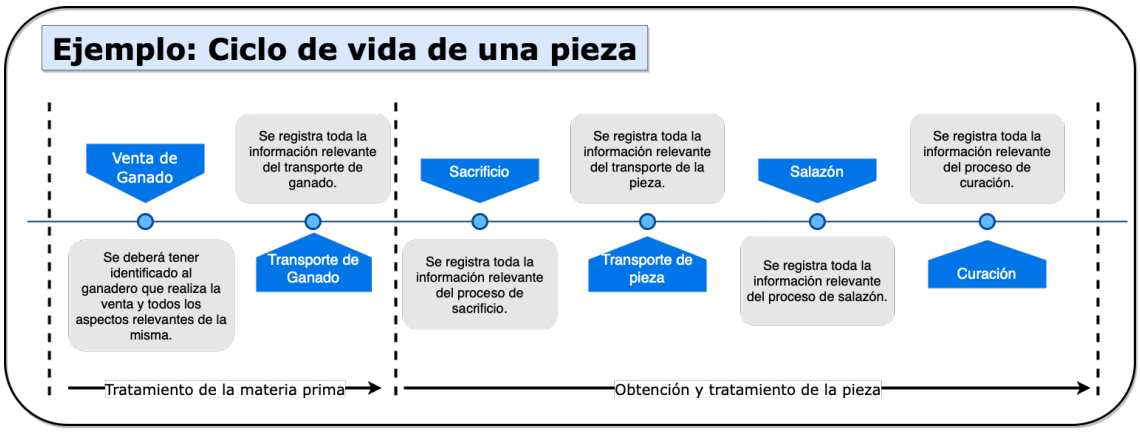

Ilustración 1. Ejemplo del ciclo de vida de una pieza.

3.2 Blockchain

Una blockchain o cadena de bloques es una estructura de datos **descentralizada**, cuya información se agrupa en conjuntos ordenados **en una red distribuida** en la que todos los participantes **almacenan una copia idéntica de la cadena** y se coordinan mediante un **algoritmo de consenso**. Un participante será cualquier ordenador que esté conectado a la red, conocido como "nodo" [6].

Las principales características de estas redes son [6]:

- Están securizadas de manera criptográfica para prevenir su manipulación y uso indebido.
- Son de tipo "*append-only*": la información solo puede ser añadida a la cadena de bloques de manera secuencial, permitiendo mantener un registro temporal de todos los eventos.
- Únicamente se puede añadir la información por consenso: para que cualquier información sea añadida a la cadena de bloques debe ser validada a través de un algoritmo de consenso.

3.2.1 Elementos genéricos de una cadena de bloques

Transacción: Es la unidad fundamental de una cadena de bloques y **representa el registro de un evento**, que puede tener una estructura diferente dependiendo de la blockchain. Estas transacciones se autentican mediante el uso de la **firma digital**, permitiendo de esta manera su validación. La mayoría de los sistemas prefieren utilizar un algoritmo de firma digital basado en curvas elípticas (ECDSA) frente a uno basado en la factorización en números primos (RSA), por su capacidad de lograr un nivel de seguridad muy similar con una clave de menor tamaño, siendo un factor determinante ya que esta información se almacena en la blockchain [6] [7].

Bloque: Se trata de un **conjunto de transacciones** agrupadas que, mediante el uso de un árbol de Merkle, permite calcular un **hash del conjunto de las transacciones** que componen el bloque, preservando así su integridad [6] [7].

Cadena de bloques: Secuencia de bloques validados y ordenados. Cada uno de estos bloques tiene almacenado en su cabecera el **hash del bloque anterior** (excepto el primer bloque, conocido como "*genesis block*"), de tal forma que el hash de un bloque depende de todos los bloques anteriores y de su orden. Este sistema dificulta enormemente la manipulación de los datos de la cadena, al tener que recalcular todos los bloques posteriores al que se desea cambiar [6] [7].

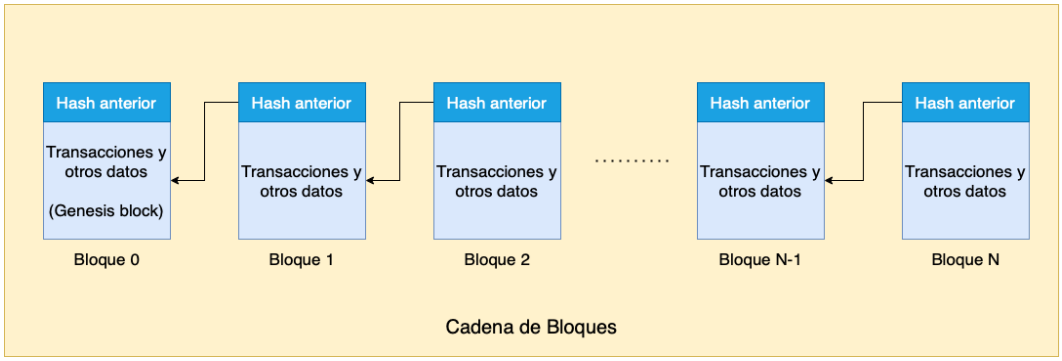

Ilustración 2. Elementos de una blockchain genérica.

3.2.2 Algoritmos de consenso

En un sistema distribuido y descentralizado pueden existir participantes no confiables o maliciosos, haciendo frecuente encontrarse con **errores bizantinos o arbitrarios** y generando la necesidad de hacer uso de algoritmos de consenso resistentes a este tipo de errores [6] [7].

Algunos de los algoritmos de consenso de uso más común en las cadenas de bloques son los siguientes:

- ***Proof of Work (PoW):*** Este mecanismo de consenso se basa en la validación de los bloques mediante la resolución de puzles hash, donde los nodos compiten para ser los primeros en solucionarlo y ser elegidos para generar el nuevo bloque. La resolución del hash se realiza mediante el uso de fuerza bruta, que consiste en ir probando todas las posibles soluciones hasta encontrar la correcta. Con este mecanismo la validación de un bloque requerirá cierto tiempo, evitando ser posible reescribir la cadena a menos que se posea un 51% o más de la capacidad de computación de la red. Este algoritmo es considerado el mejor a la hora de garantizar la integridad de la red debido a que sería necesaria una enorme cantidad de recursos computacionales para llevar a cabo ataques cuando la red está formada por un gran número de nodos. Su principal desventaja es su elevado consumo energético y su reducida velocidad, lo que limita el ancho de banda de transacciones [6] [7].
- ***Proof of Stake (PoS):*** Este mecanismo de consenso se basa en intervalos temporales, eligiendo de manera pseudoaleatoria un nodo en cada uno de estos intervalos. El nodo elegido es el único que puede crear un nuevo bloque en el intervalo de tiempo que le corresponda, validándolo mediante su propia firma digital. La lista de nodos elegibles está formada por todos aquellos nodos que congelen parte de sus activos en un fondo llamado "*stake*". El algoritmo de selección dará mayor probabilidad a aquellos nodos que posean un mayor número de activos congelados. Este algoritmo garantiza la integridad siempre y cuando ningún participante posea más del 50% del total de los fondos congelados. A pesar de ser más vulnerable que el anterior su consumo energético es mucho menor [6] [7].
- ***Delegated Proof of Stake (DPoS):*** Este mecanismo de consenso es muy similar al anterior, pero en este caso los nodos con activos congelados podrán votar para elegir a unos cuantos nodos que se encargarán de crear los nuevos bloques, llamados representantes. Esta estrategia aumenta la centralización a cambio de poder incrementar el ancho de banda de transacciones gracias a la mayor velocidad de validación de los bloques por parte de los nodos representantes, ya que se trata de una cantidad de nodos significativamente menor [6] [7].
- **Practical Byzantine Fault Tolerance (PBFT):** Se trata de un algoritmo clásico capaz de lograr un consenso en una red en la que pueden existir fallos bizantinos o arbitrarios. Este algoritmo es ampliamente usado en sistemas distribuidos, siendo capaz de lograr el consenso siempre y cuando el número de nodos con fallos bizantinos no supere la tercera parte del total. Este algoritmo se usa en redes privadas y permisionadas por su mayor ancho de banda de transacciones, no siendo factible en redes públicas debido a su mayor vulnerabilidad a ataques por parte de los participantes de la red [29].

- **Ripple:** Se trata de un algoritmo basado en PBFT, creado para el sistema blockchain con el mismo nombre. Consiste en un sistema de votación por rondas en el que se requiere de un 80 % de la red. El algoritmo garantiza la integridad siempre y cuando cada nodo posea una lista de nodos vecinos en la que los atacantes sean inferiores al 20 %. En caso de ser mayor, este nodo podría ser engañado, comprometiendo la integridad de su copia de la cadena [29].
- **Tendermint:** Se trata de un algoritmo similar a PBFT, basado en el protocolo gossip, el cual consiste en la propagación epidémica de los mensajes. De la misma forma que PBFT, se trata de un algoritmo pensado para su uso en redes con restricciones, siendo fundamental que el número de atacantes sea inferir a un tercio de los participantes con capacidad para la creación de nuevos bloques [29].

Dependiendo de los requisitos de la aplicación será más conveniente el uso de uno u otro algoritmo, existiendo sistemas capaces de configurarse para trabajar con varios de ellos [29].

Property	PoW	PoS	PBFT	DPOS	Ripple	Tendermint
Node identity management	open	open	permissioned	open	open	permissioned
Energy saving	no	partial	yes	partial	yes	yes
Tolerated power of adversary	< 25% computing power	< 51% stake	< 33.3% faulty replicas	< 51% validators	< 20% faulty nodes in UNL	< 33.3% byzantine voting power
Example	Bitcoin [2]	Peercoin [21]	Hyperledger Fabric [18]	Bitshares [22]	Ripple [23]	Tendermint [24]

Ilustración 3. Comparación de los algoritmos de consenso utilizados en blockchain.

Habitualmente, al inicio del ciclo de vida de una blockchain se apuesta por la utilización de un algoritmo de consenso tipo PoW debido a su mayor probabilidad de garantizar la integridad del sistema, pero cuando la red crece hasta alcanzar un tamaño considerable se busca la migración a un algoritmo de consenso PoS, DPoS o derivado, para reducir el consumo energético necesario para mantener la red operativa.

3.3 Ethereum

Ethereum, lanzado en 2015, se basa en la innovación iniciada con la blockchain de Bitcoin en 2008 [4], pero con algunas diferencias significativas. Ambas permiten el uso de dinero digital sin intermediarios, pero Ethereum es programable, por lo que también permite crear e implementar aplicaciones descentralizadas en su red. Que Ethereum sea programable se traduce en la posibilidad de crear aplicaciones que usan la cadena de bloques para almacenar datos o controlar lo que puede hacer la aplicación, dando como resultado una blockchain de propósito general en la que se puede desarrollar cualquier aplicación [5].

3.3.1 Funcionamiento de Ethereum

En el universo Ethereum hay una sola máquina canónica llamada "Ethereum Virtual Machine" o EVM, cuyo estado se rige de forma consensuada entre todos los participantes de la red Ethereum, guardando cada nodo de la red una copia de esta [8].

Cualquier nodo puede transmitir una solicitud para que la EVM realice un cálculo arbitrario y los demás nodos de la red recibirán la solicitud y se encargarán de verificar, validar y ejecutar el cálculo, provocando un cambio de estado en la EVM que se propaga por toda la red [8].

Las solicitudes de cómputo se denominan "solicitudes de transacción". El registro de todas las transacciones y el estado actual de la EVM se almacena en la cadena de bloques, que a su vez se almacena y consensua por todos los nodos [8].

Los mecanismos criptográficos aseguran que una vez que las transacciones se verifican como válidas y se agregan a la cadena de bloques no se puedan manipular más adelante. Estos mismos mecanismos también garantizan que todas las transacciones se firmen y ejecuten con los "permisos" apropiados (nadie debería poder crear transacciones desde la cuenta de una persona excepto ella misma) [8].

3.3.1.1 Algoritmo de Consenso

Actualmente Ethereum utiliza un algoritmo de consenso de tipo *"Proof of Work"* en el que para añadir un nuevo bloque a la cadena se debe resolver un puzle hash que requiere de una alta capacidad de computación. Este mecanismo se conoce con el nombre de "minado" y consiste en utilizar la fuerza bruta para buscar la solución del puzle hash a través de un método de prueba y error. Los nodos que se dedican a resolver estos puzles hash para añadir los nuevos bloques a la cadena se conocen como "mineros" y los que consiguen crear el nuevo bloque son recompensados con Ether (ETH), que es el activo digital o criptomoneda de la blockchain Ethereum. Los nuevos bloques se transmiten a todos los nodos de la red, se comprueban y verifican, actualizando así el estado de la cadena de bloques para todos [5].

Se está trabajando en la transición desde el mecanismo *"Proof of Work"* al *"Proof of Stake"*. Los responsables de Ethereum estiman la transición podrá realizarse entre el tercer y cuarto trimestre de 2022.

Al realizar este cambio se podrán aprovechar de las siguientes ventajas [9]:

- Mejor eficiencia energética: no será necesario usar mucha energía para minar bloques.
- Barreras de entrada más bajas, requisitos de hardware reducidos: no será necesario hardware de primer nivel para tener una oportunidad de crear nuevos bloques.
- Mayor inmunidad a la centralización: aumentará la cantidad de nodos que puedan validar los bloques gracias a una barrera de entrada más baja.

3.3.2 Ether

Ether (ETH) es la criptomoneda nativa de Ethereum, cuyo propósito es permitir la creación de un mercado para la computación. Dicho mercado proporciona un incentivo económico para que los participantes verifiquen y ejecuten solicitudes de transacciones y proporcionen sus recursos computaciones a la red [8].

Cualquier participante que transmita una solicitud de transacción también debe ofrecer cierta cantidad de Ether a la red como recompensa (llamada tarifa de transacción o *"gas"*), la cual se otorgará a quien finalmente verifique la transacción, la ejecute, la introduzca en la cadena de bloques y propague la información al resto de la red [8].

La cantidad de Ether pagada corresponde al tiempo requerido para hacer el cómputo. Estas recompensas también evitan que los participantes maliciosos obstruyan intencionalmente la red al solicitar la ejecución de un cómputo infinito u otros scripts que consumen muchos recursos [8].

3.3.2.1 Creación de nuevos Ether

La creación de nuevas criptomonedas se conoce con el término *"minting"* y es el proceso por el cual se crean nuevos Ether en Ethereum. El protocolo Ethereum subyacente es el único capaz de crear nuevos Ether, siendo imposible crearlos por un usuario [10].

Los Ether son creados a través del *"minting"* cuando se crea un nuevo bloque en la cadena. Como incentivo para construir bloques, el protocolo otorga una recompensa en cada bloque, incrementando el saldo de una dirección establecida por el productor del bloque. La recompensa por bloque ha cambiado con el tiempo y actualmente es de 2 ETH por bloque [10].

3.3.2.2 Quemado de Ether

Además de la creación de Ether mediante las recompensas por la creación de los bloques, el Ether puede destruirse mediante un proceso llamado "*burning*" (quemado), siendo retirado de la circulación de manera permanente [10].

La quema de Ether ocurre en cada transacción: cuando los usuarios pagan por sus transacciones se destruye una tarifa de gas base, establecida por la red de acuerdo con la demanda transaccional. Esto, junto con tamaños de bloque variables y una tarifa de gas máxima definida, simplifica la estimación de la tarifa de transacción. Cuando la demanda de la red es alta, los bloques pueden quemar más Ether del que se crea, compensando efectivamente la emisión de Ether [10].

La quema de la tarifa base evita la manipulación del sistema por parte de los productores de bloques. Por ejemplo, si los productores de bloques recibieran la tarifa base, podrían incluir sus propias transacciones de manera gratuita y aumentar la tarifa base para todos los demás [10].

Alternativamente podrían reembolsar la tarifa base a algunos usuarios fuera de la cadena, lo que generaría un mercado de tarifas de transacción más opaco y complejo [10].

3.3.2.3 Valor actual de Ether

Para comprender si es atractivo para los participantes de la red crear y mantener un nodo para su correcto funcionamiento se va a mostrar el valor de Ether actual en dólares (aunque se debe tener en cuenta que este es un activo muy volátil, por lo que su precio es muy variable en periodos de tiempo reducidos).

Según CoinMarketCap, una web dedicada al seguimiento del precio de los criptoactivos, el precio de Ether (ETH) actualmente se encuentra entorno a los $1.350, aunque su máximo histórico alcanzado en noviembre de 2021 era entorno a los $4.500 [11].

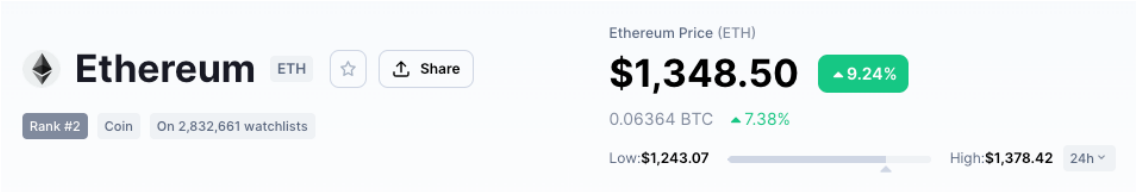

Ilustración 4. Precio actual de Ether (ETH).

En secciones posteriores se analizará el coste de realizar cada transacción teniendo en cuenta que se debe pagar una tarifa en Ether, ya que la cantidad a pagar dependerá de cada transacción.

3.3.3 Contratos inteligentes

En la práctica, los participantes de la red no escriben código nuevo cada vez que quieren solicitar un cálculo en la EVM, sino que más bien los desarrolladores de aplicaciones cargan programas (fragmentos de código reutilizables) en la EVM y los usuarios realizan solicitades para ejecutar estos fragmentos de código con diferentes parámetros. Estos programas que se conocen con el nombre de contratos inteligentes. A nivel básico, un contrato inteligente puede definirse como un programa que, al llamarse con ciertos parámetros realiza algunas acciones o cálculos sobre la blockchain si se cumplen ciertas condiciones [8].

Cualquier desarrollador puede crear un contrato inteligente y hacerlo público en la red, utilizando la cadena de bloques como su capa de datos, por una tarifa pagada a la red. Cualquier usuario puede llamar al contrato inteligente para ejecutar su código, nuevamente por una tarifa pagada a la red [8].

Ethereum pone a disposición de los desarrolladores un lenguaje para la creación de contratos inteligentes llamado Solidity, que es un lenguaje de alto nivel orientado a contratos, cuya sintaxis es similar a la de JavaScript y está enfocado específicamente a la Máquina Virtual de Ethereum (EVM) [12].

4 Estado del arte

En esta sección se presenta el estado del arte en lo que respecta a la trazabilidad alimentaria en la actualidad y aplicaciones de la tecnología blockchain en este sector, concretamente para el jamón ibérico, resaltando los puntos más relevantes y oportunidades de mejora.

Existen aplicaciones para la trazabilidad basadas en blockchain para diferentes tipos de alimentos, por ejemplo, si se consulta la página web de "ALTIM Food" ofrecen servicios de trazabilidad algunos sectores de transformación alimentaria como el lácteo, el aceite, la panadería y bollería o el vinícola [13]. Además de esta empresa existen otras dedicadas a varios sectores, pero en el sector del jamón ibérico todavía no se ha implantado el uso de esta tecnología.

Existe un servicio de trazabilidad para el jamón llamado "ÍTACA" ofrecido por la Asociación Interprofesional del Cerdo Ibérico (ASICI). La descripción que proporcionan de esta herramienta es la siguiente: "Para colaborar en la correcta implementación de la normativa y como reflejo del absoluto compromiso del sector con la transparencia y la calidad, ASICI financia, diseña, desarrolla e implanta ÍTACA (Sistema de Identificación, Trazabilidad y Calidad) una pionera herramienta digital que integra y transmite la trazabilidad de los productos ibéricos, desde el nacimiento del lechón hasta la aparición del producto en los puntos de venta, pasando por los mataderos e industrias. Una vez que la información queda registrada en ÍTACA el equipo de técnicos de la Interprofesional verifica en toda la cadena que efectivamente los datos incluidos en el Sistema son verídicos" [14].

Ilustración 5. Trazabilidad proporcionada por Ítaca.

Además, tienen disponible de manera gratuita la app "IBÉRICO" que permite verificar y consultar la trazabilidad de cualquier pieza [14].

Existe una oportunidad de mejora del sistema que proporciona Ítaca introduciendo el uso de la tecnología blockchain para asegurar la inmutabilidad de los datos y evitar el uso de intermediarios, simplificando la operativa de todos los eslabones de la cadena, donde los datos sean verificados antes de entrar en el sistema y una vez almacenados no se puedan alterar, tomando como base el sistema existente y reutilizando toda la funcionalidad posible.

5 Herramientas utilizadas

En esta sección se listan las herramientas utilizadas para el desarrollo del presente trabajo de fin de máster, agrupadas por categorías.

Sistemas externos:

- **Ethereum**: Red de cadena de bloques utilizada para el desarrollo del proyecto. Implementa la criptodivisa Ether, así como una máquina virtual llamada EVM capaz de crear y ejecutar programas llamados contratos inteligentes.
- **MongoDB**: Sistema de base de datos no relacional utilizado para almacenar la información referente a los usuarios de la aplicación que no se almacenará en la blockchain [15].

Herramientas de desarrollo y dependencias:

- **Git**: Sistema de control de versiones utilizado para gestionar los cambios y facilitar el desarrollo del proyecto [16].
- **Solidity**: Lenguaje de programación orientado a contratos específico para la máquina virtual de Ethereum [12].
- **Hardhat**: Kit de herramientas para el desarrollo de contratos inteligentes en Ethereum. Incluye un gestor de compiladores de Solidity, herramientas de integración continua, una red blockchain local y despliegue automático [17].
- **Node JS**: Intérprete de Javascript que permite la creación de aplicaciones back-end [18].
- **Ethers**: Biblioteca que permite la interacción de aplicaciones externas con la API de los nodos de Ethereum, permitiendo el envío de transacciones, así como la interacción con contratos inteligentes [19].
- **Express**: Framework web para Node JS que permite la creación de aplicaciones web y API [20].
- **React**: Framework de Javascript empleado para la creación de interfaces gráficas de aplicaciones web [21].
- **React-bootstrap**: Biblioteca con multitud de componentes de React para la construcción de páginas web de diseño responsive [22].
- **React-icons**: Biblioteca de iconos diseñados como componentes para React [23].
- **React-hot-toast:** Biblioteca que permite mostrar notificaciones diseñadas como componentes para React [24].

Editores de texto y otros documentos:

Visual Studio Code: Entorno integrado de desarrollo pensado para trabajar con una gran variedad de modelos de proyecto [25].

Diagrams: Herramienta en línea para la creación de diagramas [26].

6 Aspectos relevantes del desarrollo

En esta sección se expondrán los aspectos más relevantes para el desarrollo del proyecto: planificación, requisitos y arquitectura.

6.1 Planificación

Para el desarrollo del trabajo se ha elegido seguir un proceso iterativo e incremental.

El proceso incremental consiste en realizar el diseño del producto completo para posteriormente dividir su desarrollo en trozos que se puedan construir por separado y que serán integrados una vez creados. Para el desarrollo de cada uno de los trozos se seguirá una secuencia de fases (incrementos):

- **Análisis:** Se realiza un estudio para definir los requisitos del sistema y fijar los objetivos del incremento.
- **Diseño:** Se realiza la creación del modelo de datos y se define la arquitectura del sistema para satisfacer los requisitos y objetivos marcados en la fase anterior.
- **Implementación:** Fase de programación que dará como resultado el sistema final.
- **Pruebas:** Se comprueba el correcto funcionamiento del sistema a través de diferentes tipos de pruebas (unitarias, de integración) y se subsanan los errores encontrados.

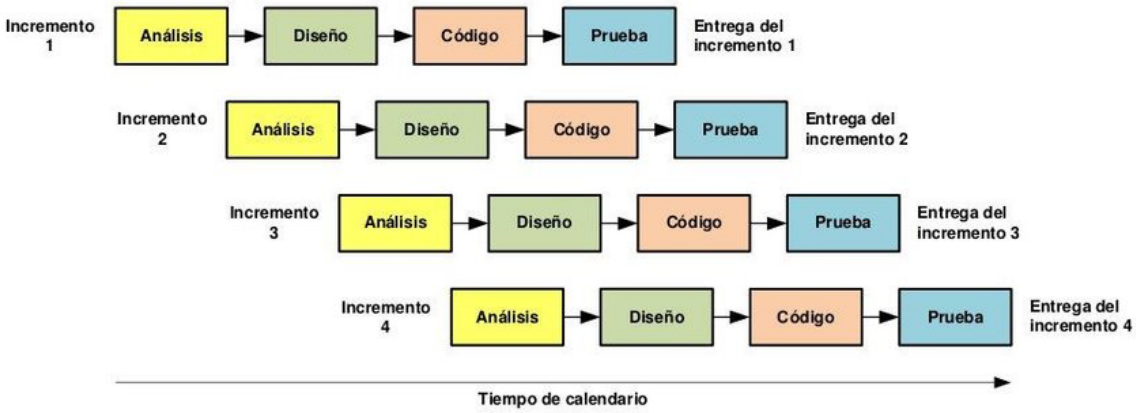

Ilustración 6. Modelo incremental.

El proceso iterativo consiste en comenzar diseñando, construyendo y probando la versión más pequeña posible del sistema e ir analizando los resultados obtenidos, haciendo aproximaciones al producto final mediante un proceso de prueba y error. Para este trabajo se parte de una idea inicial que sirve como base, pero debido al desconocimiento de las tecnologías a utilizar y del modelo que se trabaja es muy posible que existan variaciones en los requisitos definidos inicialmente, por lo que seguir un proceso iterativo permite adaptarse a estos cambios durante el desarrollo de cada incremento.

Se han definido los siguientes incrementos:

- **Incremento 1:** Desarrollo de toda la lógica de los contratos inteligentes.
- **Incremento 2:** Desarrollo del servidor intermediario para comunicación con la blockchain.
- **Incremento 3:** Desarrollo de la interfaz gráfica de usuario.

En base a lo definido hasta el momento se realiza una estimación temporal para cada una de las fases basada en su complejidad estimada.

Incremento	Análisis	Diseño	Implementación	Pruebas	Estimación incremento
Incremento 1	5 días	5 días	6 días	3 días	19 días
Incremento 2	3 días	5 días	4 días	3 días	15 días
Incremento 3	4 días	5 días	14 días	5 días	28 días
Estimación total					**62 días**

6.2 Requisitos

Una de las partes más importantes a la hora de iniciar un proyecto software es hacer una toma correcta y completa de los requisitos.

6.2.1 Requisitos de información

En esta definición de requisitos se recogerán los aspectos referentes a la información que se almacenará en el sistema de manera permanente. Se han identificado los siguientes:

- **[RI-001] Usuario**: Se debe almacenar la información de los usuarios para su autenticación en el sistema y permitir la interacción con la API.
- **[RI-002] Lote**: Se deben almacenar las operaciones que generan los usuarios referentes a lotes ganaderos.
- **[RI-003] Pieza:** Se deben almacenar las operaciones que generan los usuarios referentes a las piezas.

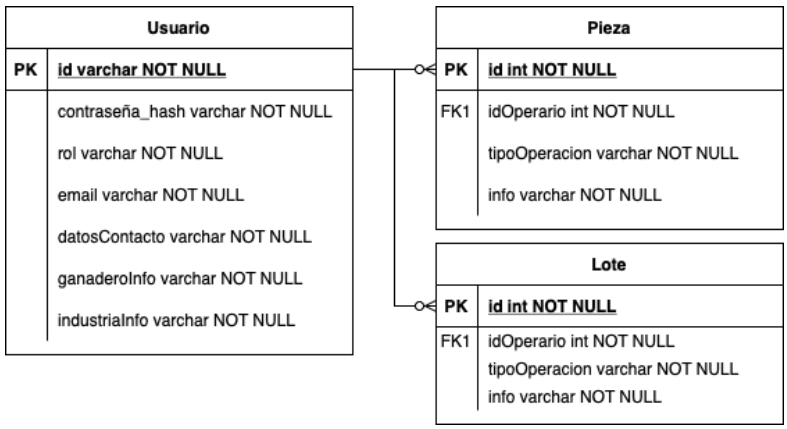

Ilustración 7. Diagrama Entidad-Relación de los requisitos de información.

En esta definición de requisitos se recogerán los aspectos referentes a la funcionalidad del sistema, incluyendo las interacciones de los usuarios.

Los requisitos funcionales se han planteado en base a los siguientes actores:

- **[A-001] Participante:** Actor genérico que agrupa a todos los actores del sistema, excepto a **[A-006] Administrador.**
- **[A-002] Consumidor:** Este actor representa a los clientes finales que compran una pieza de jamón ibérico.
- **[A-003] Ganadero:** Este actor representa a un operario que puede generar operaciones con lotes de ganado.
- **[A-004] Industria:** Este actor representa a un operario que puede generar operaciones con lotes de ganado o piezas.
- **[A-005] Auditor:** Este actor representa a un operario externo que tiene capacidades de monitorización.
- **[A-006] Administrador:** Este actor representa a la entidad encargada de la gestión de los usuarios, así como de la monitorización.

Existen las siguientes herencias entre actores:

[A-006] Administrador hereda de **[A-005] Auditor**.

[A-002] Consumidor, **[A-003] Ganadero**, **[A-004] Industria**, **[A-005] Auditor** heredan de **[A-001] Participante**.

Se han identificado los siguientes requisitos funcionales:

- **[RF-001] Solicitar alta de consumidor:** El sistema debe permitir a cualquier participante darse de alta como consumidor a través de un formulario.
- **[RF-002] Solicitar alta de ganadero:** El sistema debe permitir a cualquier participante darse de alta como ganadero a través de un formulario.
- **[RF-003] Solicitar alta de industria:** El sistema debe permitir a cualquier participante darse de alta como industria a través de un formulario.
- **[RF-004] Iniciar sesión:** El sistema debe permitir la autenticación de los usuarios mediante un formulario de inicio de sesión formado por nombre de usuario y contraseña.
- **[RF-005] Cambiar contraseña:** El sistema debe permitir a los usuarios cambiar su contraseña.
- **[RF-006] Editar información de contacto:** El sistema debe permitir a los usuarios cambiar su información de contacto.
- **[RF-007] Cerrar sesión:** El sistema debe permitir a los usuarios cerrar su sesión, invalidando su código de acceso.
- **[RF-008] Consultar una pieza:** El sistema debe permitir a los consumidores consultar una información básica de una pieza a través de su identificador.
- **[RF-009] Insertar venta de ganado:** El sistema debe permitir a los ganaderos insertar una operación de venta de ganado en el sistema.

- **[RF-010] Insertar transporte de ganado:** El sistema debe permitir tanto a los ganaderos como a las industrias insertar una operación de venta de ganado en el sistema.
- **[RF-011] Ver histórico de sus operaciones:** El sistema debe permitir tanto a los ganaderos como a las industrias ver el histórico de sus operaciones.
- **[RF-012] Ver detalle operación:** El sistema debe permitir tanto a un operario (industria o ganadero) como a un auditor o administrador ver toda la información de una operación.
- **[RF-013] Insertar Sacrificio:** El sistema debe permitir a las industrias insertar una operación de sacrificio en el sistema.
- **[RF-014] Insertar Curación:** El sistema debe permitir a las industrias insertar una operación de curación en el sistema.
- **[RF-015] Insertar transporte de pieza:** El sistema debe permitir a las industrias insertar una operación de transporte de pieza en el sistema.
- **[RF-016] Ver histórico de operaciones:** El sistema debe permitir tanto a los auditores como a los administradores ver el histórico de operaciones global.
- **[RF-017] Alta auditor:** El sistema debe permitir al administrador dar de alta un nuevo usuario auditor.
- **[RF-018] Listar usuarios:** El sistema debe permitir al administrador listar a los usuarios del sistema.
- **[RF-019] Ver detalle usuario:** El sistema debe permitir al administrador ver información detallada de los usuarios.
- **[RF-020] Eliminar usuario:** El sistema debe permitir al ver administrador información detallada de los usuarios.

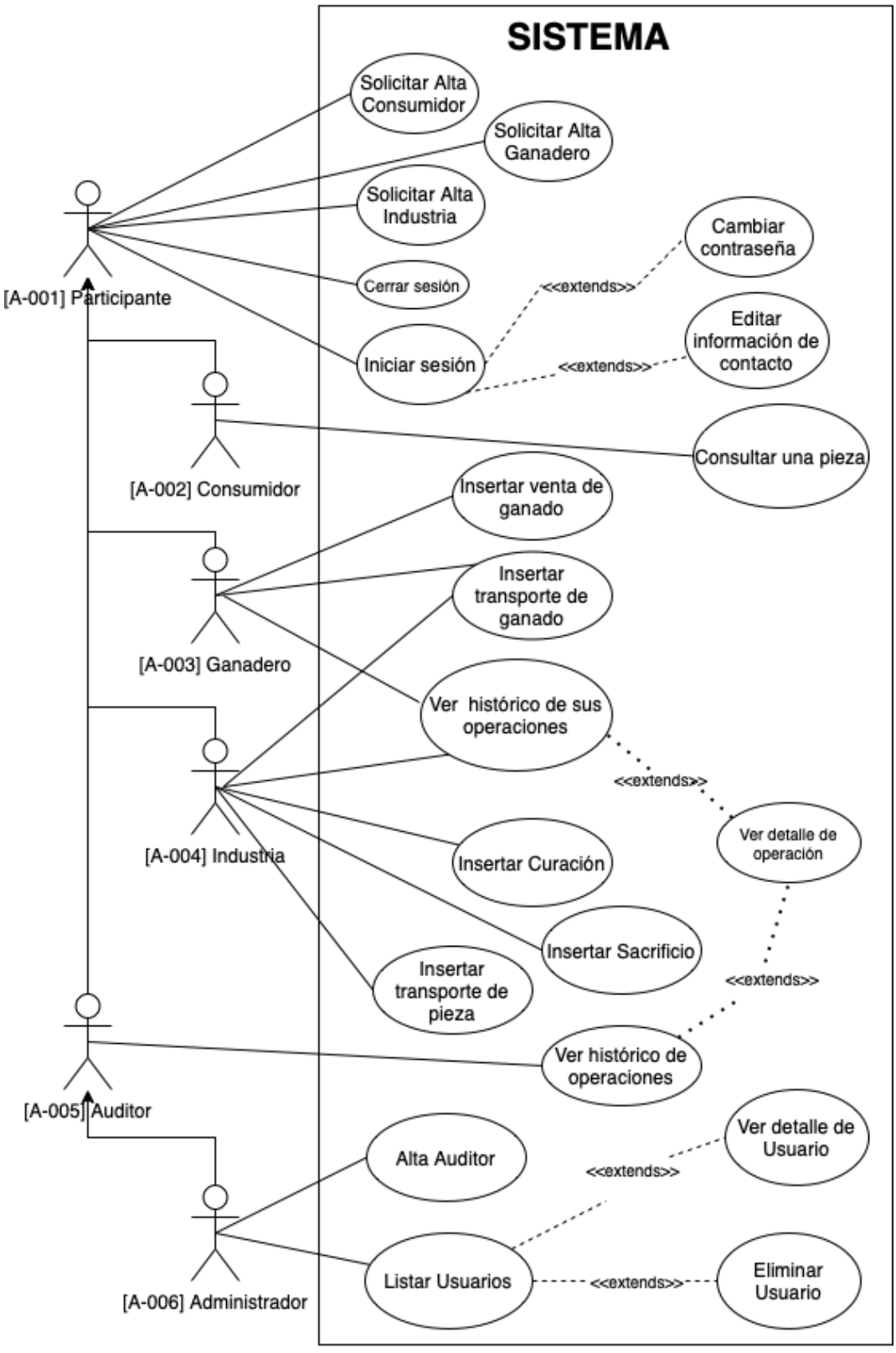

Ilustración 8. Diagrama de casos de uso del sistema.

6.2.2 Requisitos no funcionales

En esta definición de requisitos se recogerán el resto de los aspectos no relacionados con la funcionalidad del sistema.

Se han identificado los siguientes requisitos no funcionales:

- **[RN-001] Inmutabilidad y seguridad de los datos**: El sistema debe garantizar que la información almacenada es veraz y no se puede manipular.
- **[RN-002] Autenticación:** El sistema debe asegurar que todos los usuarios que realicen operaciones estén identificados para evitar acciones maliciosas.

6.3 Arquitectura

El sistema planteado se dividirá en tres bloques principales, correspondientes a cada uno de los incrementos definidos, que interactuarán entre sí para formar el sistema completo. El flujo de la información comenzará en el servidor web (interfaz gráfica de usuario) donde los actores podrán realizar solicitudes a la API del servidor intermediario y este será el encargado de gestionar las transacciones en la blockchain.

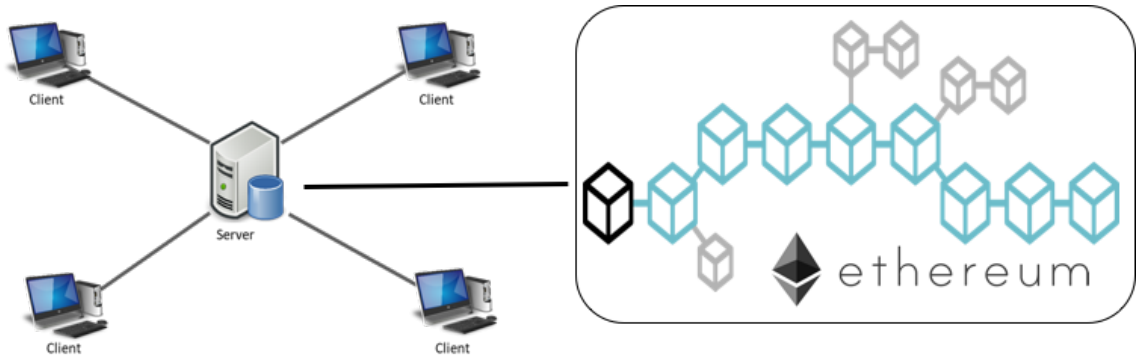

Ilustración 9. Arquitectura global.

En este proyecto se ha tomado un planteamiento en el que únicamente existe una cartera (*wallet*) manejada por el servidor para conectarse a la blockchain por sencillez, para el caso de estudio, perdiendo de esta manera una de las grandes ventajas de la blockchain: que es un sistema distribuido y descentralizado. En un entorno real o de producción, cada cliente debería tener su propia cartera y conectarse directamente a la blockchain desde el navegador para firmar sus propias transacciones.

Para el desarrollo existen herramientas que permiten crear blockchain locales, ofreciendo una enorme comodidad a la hora de desarrollar ya que la velocidad de las transacciones es considerablemente mayor a la que se obtiene de las redes oficiales, además de no necesitar utilizar criptodivisas reales, que tienen un alto valor económico. Además de las redes oficiales existen redes para pruebas, en las que probar los contratos inteligentes en un entorno real, pero también se obtienen tiempos de respuesta por transacción muy altos. En este caso se utilizará una red blockchain local para poner a prueba el sistema, pero para un entorno real o de producción, tras el testeo correspondiente en las redes habilitadas para tal uso se debería utilizar la red principal.

6.3.1 Modelado de contratos inteligentes

Los contratos inteligentes serán los encargados de generar las transacciones para almacenar la información de los procesos de los usuarios en la blockchain.

Se debe tener en cuenta que una blockchain no es una base de datos, ya que cuanta más información se almacena en el contrato inteligente más gas (tarifa de transacción) costarán las transacciones. Los métodos de los contratos inteligentes que recuperan información de la blockchain no tienen coste, pero los métodos que almacenan información sí.

Todas las transacciones pueden generan un log o registro de eventos, que es una estructura de datos indexada en la que se puede escribir la información de manera más "barata" mediante los eventos de Solidity. Los contratos inteligentes no pueden acceder a los datos del log después de crearse, pero estos datos pueden ser accedidos desde fuera de la blockchain de forma eficiente, pudiendo utilizar hasta tres parámetros para filtrar la información del log.

Con esta premisa se crea un único contrato inteligente (llamado `Trazabilidad`) en el que se almacenará la información mínima necesaria para el funcionamiento del sistema, enviando el resto de información al log mediante eventos.

El contrato inteligente contará con cuatro atributos:

Tipo	Visibilidad	Atributo	Descripción
`address`	`public`	`owner`	Almacena la dirección que despliega el contrato.
`uint256`	`private`	`s_numLote`	Almacena el ultimo identificador de lote.
`uint256`	`private`	`s_numPieza`	Almacena el último identificador de pieza.
`mapping(uint256 =>uint256)`	`private`	`s_piezaALote`	Relación que determina a qué lote pertenece cada pieza.

El tipo de visibilidad `private` indica que estos atributos no pueden escribirse ni leerse desde fuera del contrato inteligente, mientras que el tipo de visibilidad `public` sí lo permite.

Para poder recuperar el valor de los atributos privados se hace uso de los siguientes métodos:

Método	Tipo	Descripción
`getNumLotes()`	`public view`	Devuelve el valor del atributo `s_numLote`.
`getNumPiezas()`	`public view`	Devuelve el valor del atributo `s_numPieza`.
`getLoteOfPieza(uint256 _idPieza)`	`public view`	Devuelve el lote al que pertenece la pieza.

Los métodos con la palabra clave `view` indican que el método únicamente devuelve información de la blockchain y no escribirá ningún valor.

Para garantizar la seguridad de la información almacenada se ha creado el modificador `onlyOwner`, que restringe los métodos a los que se aplica para que solo puedan ser ejecutados por la dirección almacenada en `owner`, haciendo que nadie más pueda incluir operaciones en la blockchain a través de este contrato. Este modificador se ha aplicado a todos los métodos del contrato.

```
    modifier onlyOwner() {
        require(owner == msg.sender, "Must be owner");
        _;
    }
```

Si una dirección que no sea `owner` ejecuta un método que tenga este modificador recibirá el mensaje de error `"Must be owner"`.

Para el almacenamiento de la información en el log se utilizan los siguientes eventos:

Evento	Descripción

`NuevaOperacionGanadero(`
`uint256 indexed numLote,`
`string indexed idGanadero,`
`uint8 indexed tipoOperacion,`
`string info,`
`uint256 timestamp`
`)`
	Evento emitido siempre que se introduzca una operación de un operario tipo ganadero en la blockchain.

`NuevaOperacionIndustriaLote(`
`uint256 indexed numLote,`
`string indexed idIndustria,`
`uint8 indexed tipoOperacion,`
`string info,`
`uint256 timestamp`
`)`
	Evento emitido siempre que se introduzca una operación de un operario tipo industria relacionado con un lote en la blockchain.

`NuevaOperacionIndustriaPieza(`
`uint256 indexed numPieza,`
`string indexed idIndustria,`
`uint8 indexed tipoOperacion,`
`string info,`
`uint256 timestamp`
`)`
	Evento emitido siempre que se introduzca una operación de un operario tipo industria relacionado con una pieza en la blockchain.

Los atributos con el literal `indexed` indican que son parámetros por los que se puede buscar para recuperar el log, devolviendo los registros que coincidan con los parámetros introducidos o todos los registros si el parámetro es `null`. Esta búsqueda es muy potente, ya que no se limita a un único valor, es capaz de recibir, por ejemplo, un array de valores y devolver los registros que tengan alguno de los valores del array en el parámetro especificado.

Dado que se pueden almacenar diferentes tipos de operaciones, el campo `info` se rellenará con la información específica de cada operación en formato JSON.

Por último, se añade un campo `timestamp` para poder ordenar los registros cronológicamente cuando se recuperen.

Para la emisión de estos eventos el contrato cuenta con tres métodos:

Método	Tipo	Descripción

`registrarNuevaOperacionGanadero(`
`uint256 _idLote,`
`string memory _idGanadero,`
`uint8 _tipoOperacion,`
`string memory _info`
`)`
	`external`	Emite el evento `NuevaOperacionGanadero` para registrar la operación.

`registrarNuevaOperacionIndustriaLote(`
`uint256 _numLote,`
`string memory _idIndustria,`
`uint8 _tipoOperacion,`
`uint256 _numNuevasPiezas,`
`string memory _info`
`)`
	`external`	Emite el evento `NuevaOperacionIndustriaLote` para registrar la operación. Cuando es una operación de tipo sacrificio actualiza los atributos del contrato.

`registrarNuevaOperacionIndustriaPieza(`
`uint256 _idPieza,`
`string memory _idIndustria,`
`uint8 _tipoOperacion,`
`string memory _info`
`)`
	`external`	Emite el evento `NuevaOperacionIndustriaPieza` para registrar la operación.

Los métodos de tipo `external` son aquellos que únicamente se pueden ejecutar desde fuera del contrato inteligente.

Con este planteamiento donde únicamente el `owner` puede ejecutar los métodos de inserción de datos en la blockchain se está centralizando la aplicación. Se ha planteado de esta manera por sencillez, pero en un entorno real o de producción habría que proporcionar un método para dar permiso a ciertas direcciones aparte del `owner` y descentralizar el sistema.

6.3.2 Servidor intermediario para comunicación con blockchain

El servidor intermediario es el elemento central de la arquitectura ya que será el encargado de conectar la interfaz de usuario con las transacciones de la blockchain, por lo que contendrá toda la lógica de negocio. Las funciones del servidor serán las siguientes:

- Gestionar la interacción con el nodo de Ethereum, tanto para generar transacciones que carguen información como para recuperar la información de los eventos y poder presentársela al usuario a través de la interfaz gráfica.
- Gestionar la base de datos MongoDB, donde se almacenarán los datos de los usuarios.
- Securizar el acceso los recursos que ofrece, para que únicamente puedan acceder usuarios autenticados a la información.
- Atender las peticiones HTTP que reciba a través de la API o de la interfaz gráfica.

El servidor se ha creado utilizando nodeJS para permitir la utilización de Javascript en backend, con el framework ExpressJS. La conexión con la blockchain se realiza a través de la biblioteca de funciones Ethers.

El patrón de diseño de software utilizado para el servidor intermediario es el Modelo-Vista-Controlador.

6.3.2.1 Modelo-Vista-Controlador

Modelo-Vista-Controlador (MVC) es un patrón de diseño de software que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos [27].

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, sobre multitud de lenguajes y plataformas de desarrollo [27].

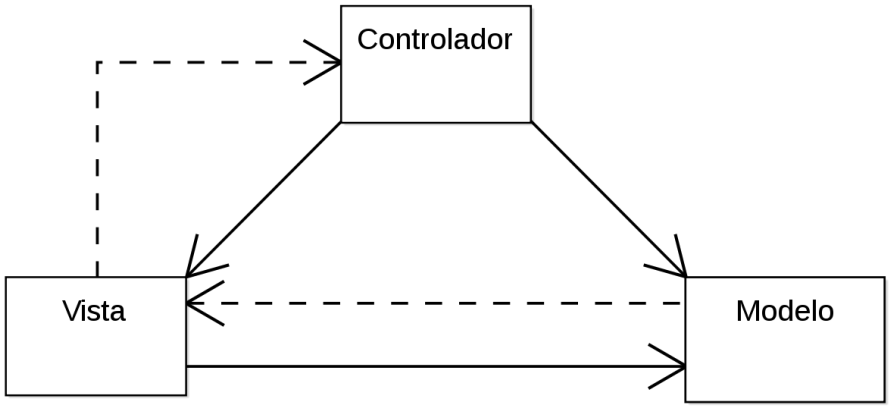

Ilustración 10. Patrón de diseño de software MVC.

El Modelo contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia [27].

La Vista (o interfaz de usuario) que compone la información que se envía al cliente y los mecanismos de interacción con este [27].

El controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno [27].

El flujo que se sigue con este tipo de patrones es generalmente el siguiente [27]:

1. El usuario interactúa con la interfaz gráfica (vista) de alguna forma.
2. El controlador recibe la notificación de acción solicitada por el usuario y la gestiona.
3. El controlador accede al modelo para llevar a cabo la acción solicitada por el usuario.
4. El controlador devuelve la información a la vista, que será la encargada de mostrarla de manera correcta.
5. La interfaz de usuario espera nuevas interacciones del usuario.

6.3.2.2 Modelo de Datos

Esta sección corresponde al Modelo del patrón MVC. Para este sistema el modelo de datos se ha definido en la sección "Requisitos de información" y se puede consultar la Ilustración 6 para ver el Diagrama de Entidad-Relación.

La información de la entidad Usuario se almacenará en la base de datos MongoDB, basada en documentos, con formato JSON. La información de las entidades Lote y Pieza se almacenará directamente en la blockchain, ya que es información sensible que debe ser extremadamente difícil de manipular de manera maliciosa.

6.3.2.3 API HTTP

Esta sección corresponde al Controlador del patrón MVC, donde se definirán las funcionalidades del servidor intermediario para poder comunicarse y brindar la información necesaria a la interfaz gráfica (Vista) a través de una API HTTP.

La estructura de la API se ha dividido por funcionalidades para una mejor organización, existiendo tres grupos diferenciados:

API de autenticación: Funciones para la autenticación de los usuarios. Si no están autenticados únicamente podrán acceder a las rutas públicas.

Método	Ámbito	Ruta	Descripción
POST	Publico	`/auth`	Inicia la sesión del usuario a través de nombre de usuario y contraseña. Si la información es correcta le devuelve un token de acceso y un token de refresco a través de una cookie.
GET	Publico	`/auth/refresh`	Se solicita cuando el token de acceso ha expirado. Devuelve un nuevo token de acceso si el token de refresco no ha expirado.
POST	Público	`/auth/logout`	Cierra la sesión del usuario. Elimina la cookie.

Para la implementación de la API de seguridad se han utilizado JSON Web Token (JWT): un estándar abierto basado en JSON para la creación de tokens de acceso que permiten la propagación de la identidad y privilegios. El token está firmado por la clave del servidor, por lo que tanto el cliente como el servidor son capaces de verificar que el token es legítimo [28].

Para la securización de las rutas privadas se ha implementado un middleware que comprueba si el solicitante manda la cookie con el token de acceso, en caso de no tener cookie se deniega el recurso. Adicionalmente se ha implementado otro middleware que limita el número de intentos de inicio de sesión de los usuarios y deberán esperar cierto tiempo antes de poder volver a intentarlo.

Todas las rutas son públicas para que todos los participantes puedan acceder a la aplicación.

API de usuarios: Funciones para la gestión de los usuarios del sistema.

Método	Ámbito	Ruta	Descripción
GET	Privado	`/users`	Devuelve una lista con todos los usuarios del sistema, excluyendo al administrador y excluyendo las contraseñas de los usuarios.
POST	Público	`/users`	Crea los usuarios que solicitan acceder a la aplicación, de tipo consumidor, ganadero o industria.
PATCH	Privado	`/users`	Modifica los datos de contacto de un usuario.
DELETE	Privado	`/users`	Elimina un usuario.
POST	Privado	`/users/auditor`	Crea un usuario de tipo auditor.
POST	Privado	`/users/getUser`	Devuelve un usuario concreto.
POST	Privado	`/users/changePassword`	Modifica la contraseña de un usuario.

API de trazabilidad: Funciones para interactuar con la blockchain.

Método	Ámbito	Ruta	Descripción
POST	PRIVADO	/trazabilidad/operation	Genera una operación en blockchain en función de los parámetros recibidos.
POST	PRIVADO	/trazabilidad/filter	Devuelve los eventos que satisfacen la búsqueda con los parámetros recibidos.

6.3.3 Interfaz gráfica de usuario

Esta sección corresponde a la Vista del patrón MVC, donde se captarán las solicitudes de los usuarios para hacérselas llegar al Controlador y se mostrará la información que este devuelva.

La Vista se ha creado utilizando el framework ReactJS, cuyos puntos fuertes son las páginas de tipo "single page" y la reutilización de los componentes. Con esta biblioteca no será necesario recargar la página cada vez que se quieran actualizar los datos a mostrar gracias a que controla el estado y muestra los componentes actualizados cuando cambia su valor, reacciona a los cambios. Además, se utiliza la biblioteca react-bootstrap que contiene componentes genéricos para React con diseño responsive, facilitando enormemente la creación de la Vista.

Debido a que esta interfaz de usuario debe ser utilizada por participantes que pueden no conocer el funcionamiento de esta, se deben tener en cuenta algunas pautas:

- Crear un diseño consistente, donde los componentes de la interfaz sean fáciles de localizar para el usuario y sigan el mismo patrón.
- Evitar los errores humanos: se deberán colocar comprobaciones en los campos donde el usuario introducirá la información para que no llegue información errónea a la API.
- Mostrar mensajes que informen al usuario de la ruta en la que se encuentra y del resultado de las acciones que realiza.
- La vista debe adaptarse a diferentes tamaños de pantalla para poder utilizarse desde cualquier dispositivo (diseño responsive).

Todas las pantallas siguen el mismo patrón: una barra de navegación en la parte superior que indica la ruta en la que se encuentra el usuario, una barra de información en la parte inferior que mostrará los datos del usuario cuando haya iniciado sesión y el contenido principal de la página que variará en función del usuario y la ruta.

A continuación, se mostrará el diseño de cada una de las pantallas, junto con la información que proporciona y el resultado final de la misma.

6.3.3.1 *Pantalla principal*

Esta pantalla es la toma de contacto con los usuarios: debe contener ciertas instrucciones sobre la aplicación y cómo utilizarla.

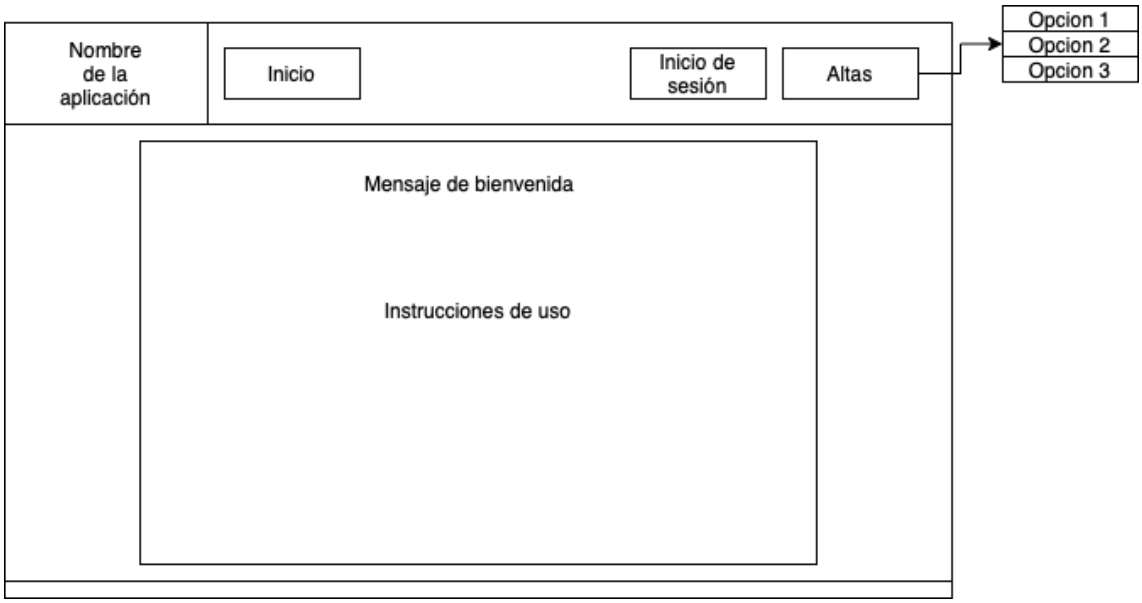

Ilustración 11. Diseño de la pantalla principal.

Se compone de un menú en la barra superior en el que se indican las opciones que tiene un usuario: ir a la página de inicio, iniciar sesión o darse de alta por alguna de las opciones.

Una zona principal que contendrá un mensaje de bienvenida y unas instrucciones para la correcta utilización de la aplicación.

Una barra inferior que estará vacía mientras el usuario no haya iniciado sesión.

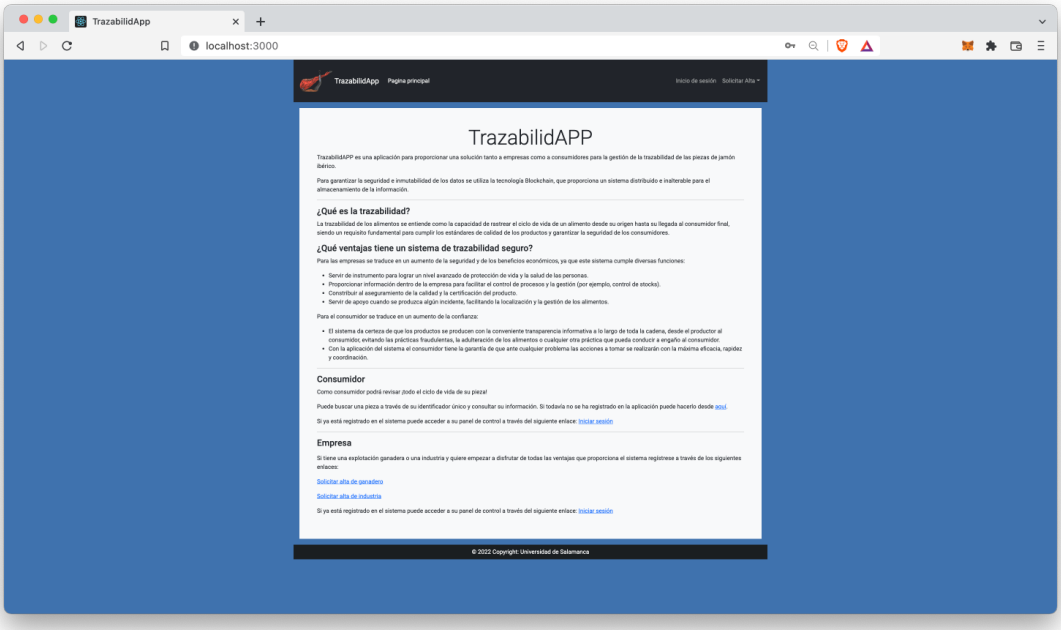

Ilustración 12. Captura de la pantalla principal.

6.3.3.2 *Pantalla de solicitud de alta de usuario*

A través de estas vistas se permitirá el alta de los usuarios en el sistema. Seguirá el mismo patrón que la pantalla principal, pero en la zona principal tendrá el formulario para la introducción de los datos del usuario.

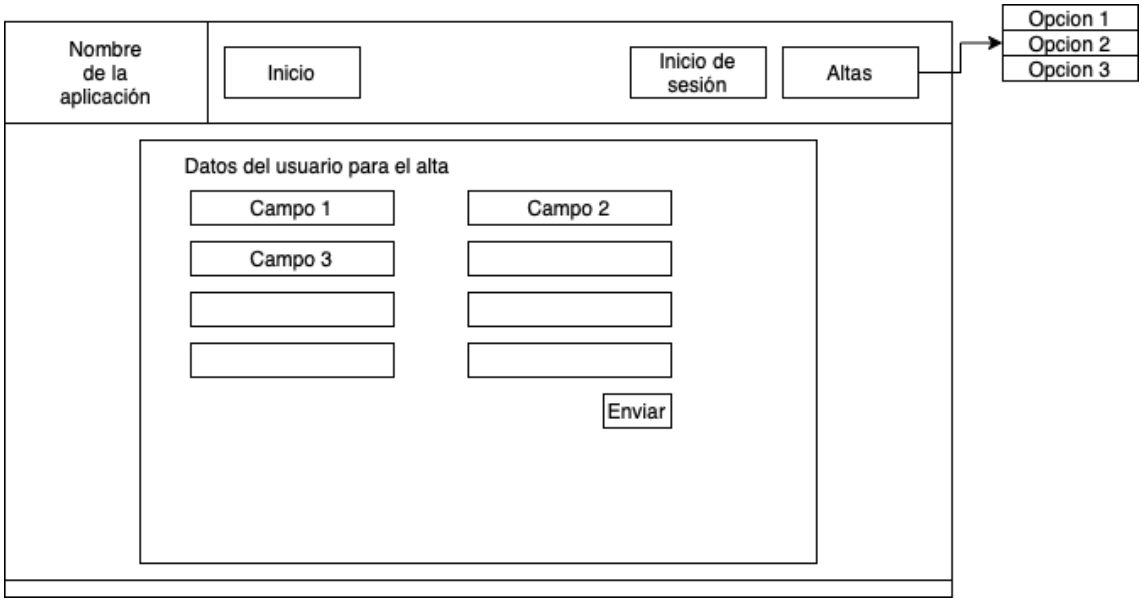

Ilustración 13. Diseño de las páginas de solicitud de alta.

El diseño de las tres pantallas de alta es el mismo, lo único que difieren es en los campos del formulario ya que se necesita información específica para cada tipo de usuario.

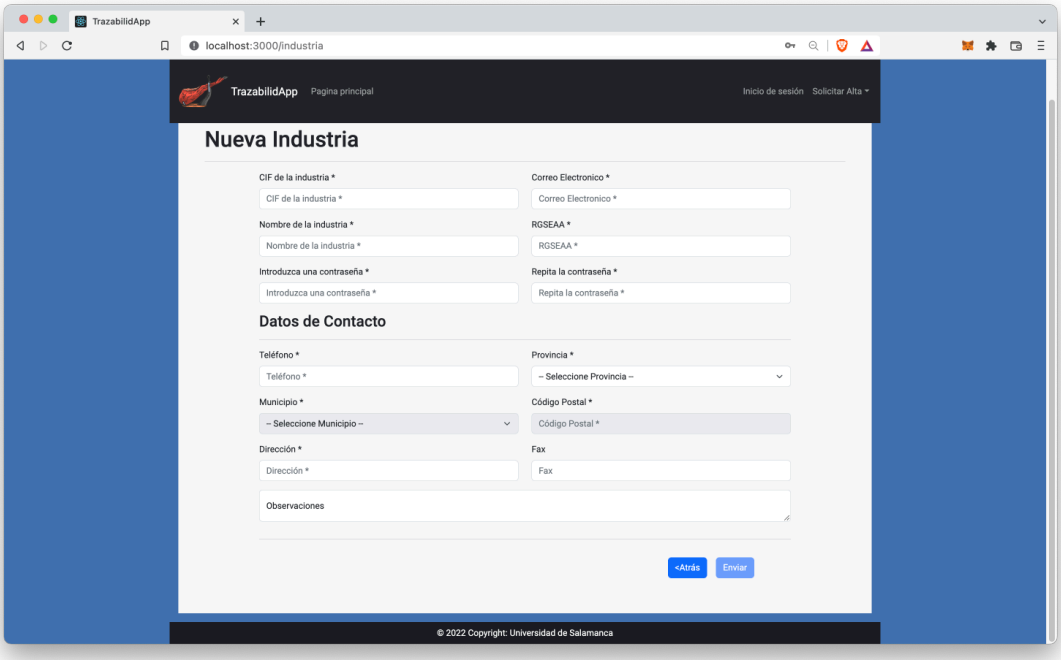

Ilustración 14. Captura de la pantalla de alta de industria.

6.3.3.3 *Pantalla de inicio de sesión*

Esta pantalla tendrá la misma estructura global que las anteriores, pero en la parte principal tendrá un formulario para introducir el nombre de usuario y la contraseña.

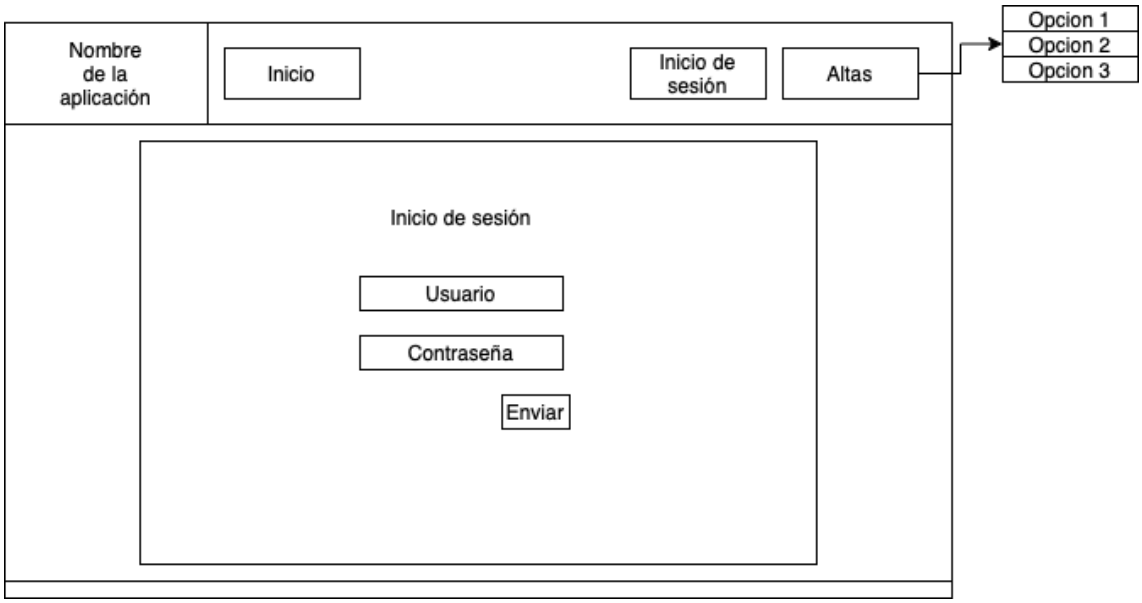

Ilustración 15. Diseño de pantalla de inicio de sesión.

Al haber discrepancias entre la información a introducir entre unos tipos de usuario y otros se ha decidido finalmente añadir unas instrucciones para el inicio de sesión.

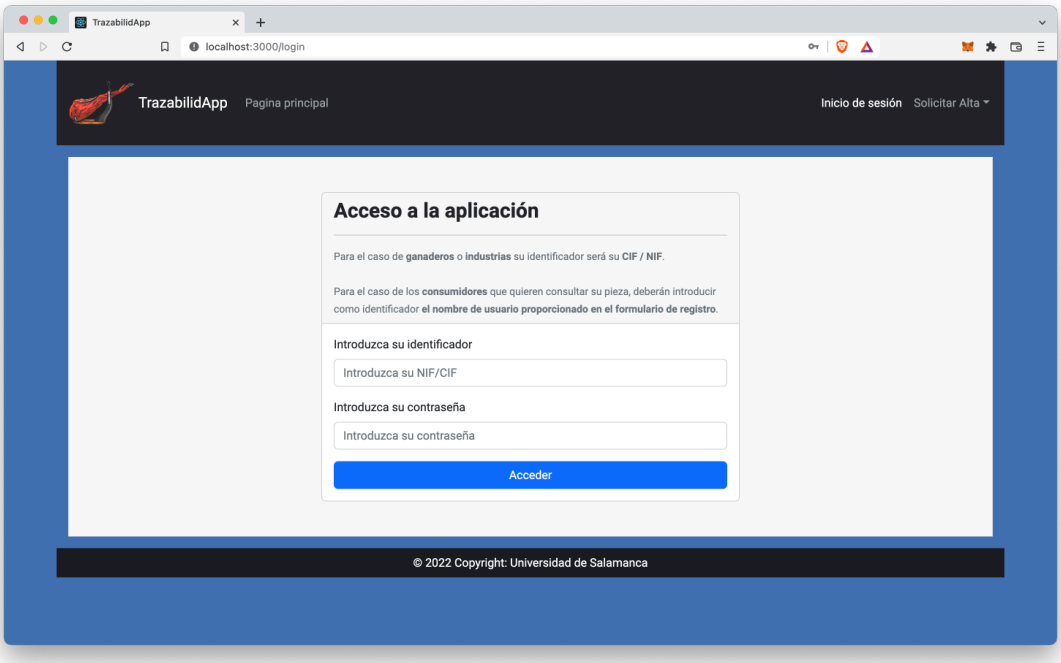

Ilustración 16. Captura de pantalla de inicio de sesión.

6.3.3.4 *Pantalla principal tras iniciar sesión*

Esta pantalla seguirá el mismo patrón, pero en la barra inferior aparecerá el nombre del usuario y su rol (consumidor, ganadero, industria, auditor o administrador).

En la barra superior encontraremos como opciones las operaciones que puede realizar el usuario y además opciones de gestión de su perfil como cambiar contraseña, editar información de contacto o cerrar sesión.

En la pantalla principal se encontrará un cuadro de diálogo que permitirá introducir los parámetros para buscar los eventos de las operaciones que ha creado y mostrarlas en la misma pantalla en una tabla.

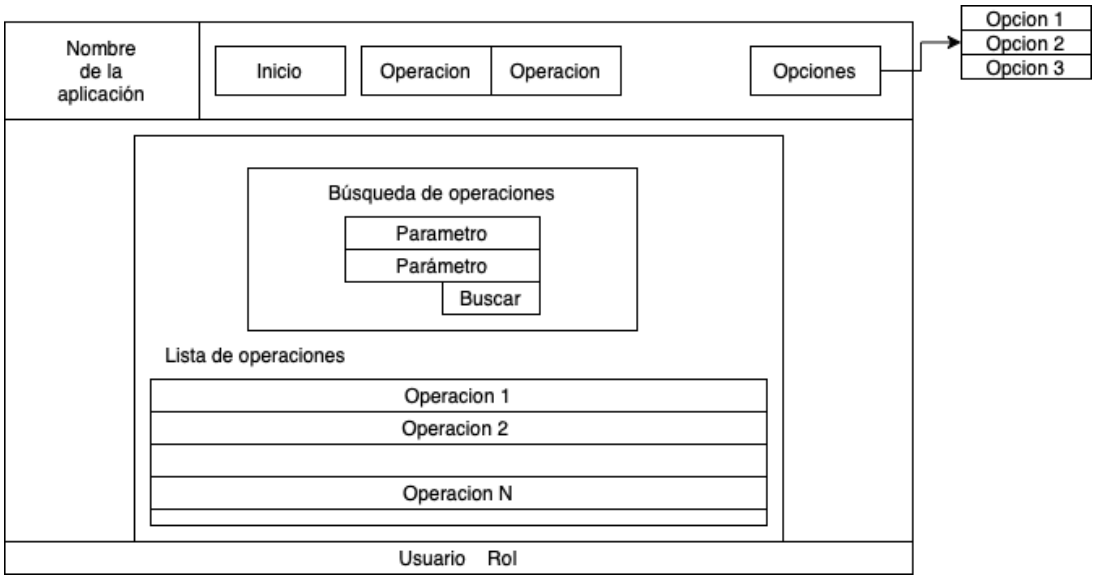

Ilustración 17. Diseño pantalla principal una vez autenticado.

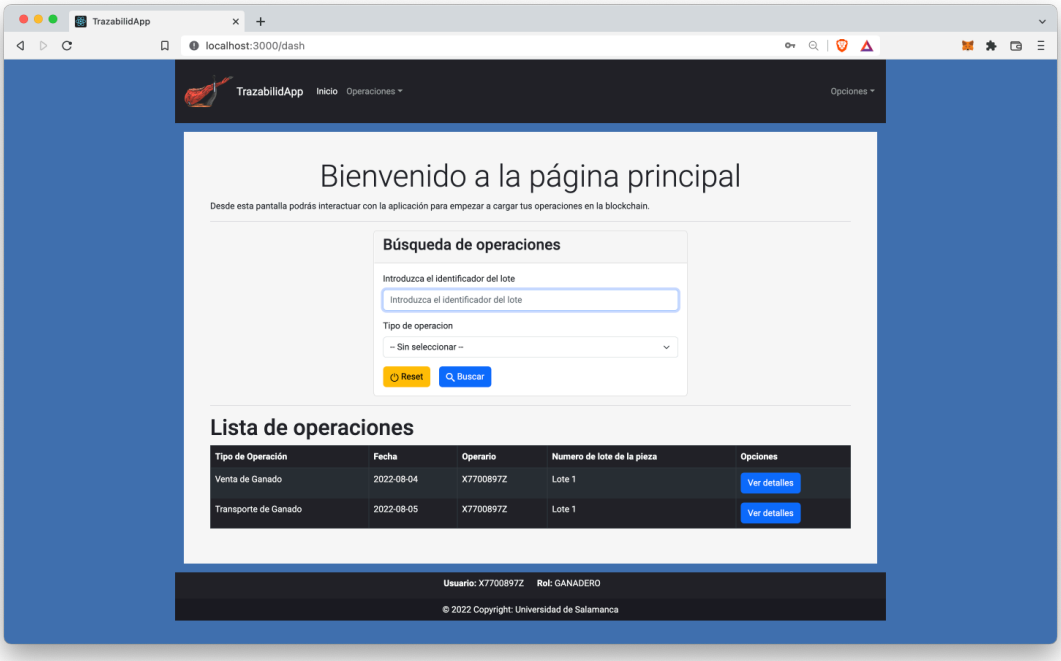

Ilustración 18. Captura de la pantalla principal de un usuario tipo ganadero.

6.3.3.5 *Pantalla de detalle de operaciones*

Todos los usuarios excepto los consumidores tendrán la posibilidad de consultar los detalles de las operaciones mediante un botón situado en cada una de las filas que forman la tabla de operaciones.

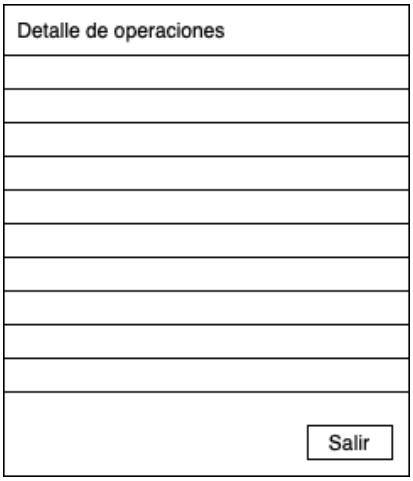

Ilustración 19. Diseño de pantalla de detalle de operaciones.

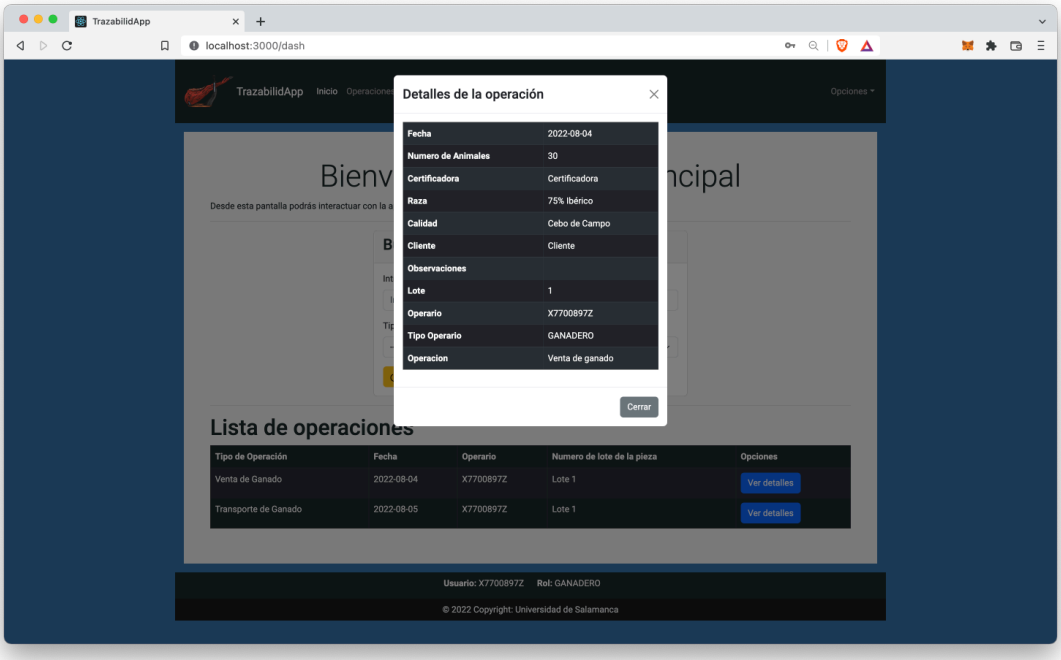

Ilustración 20. Captura de pantalla de detalle de operaciones.

6.3.3.6 *Pantallas de alta de operaciones*

Se muestra como ejemplo una pantalla para la creación de operaciones, ya que todas siguen el mismo diseño y lo único en lo que difieren es en la información solicitada en el formulario.

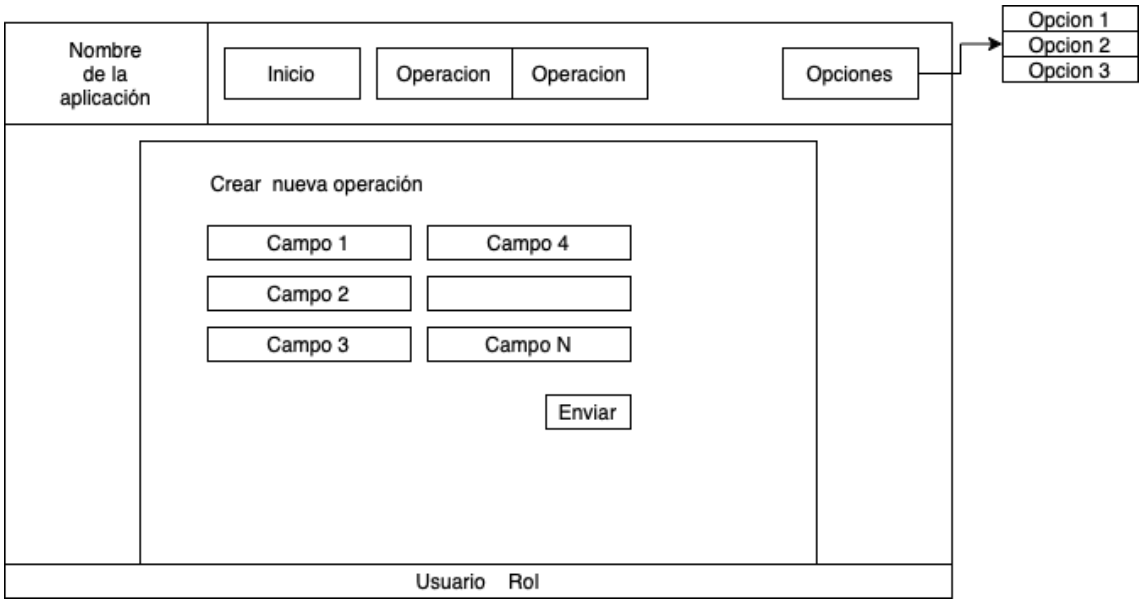

Ilustración 21. Diseño de pantalla de alta de operación.

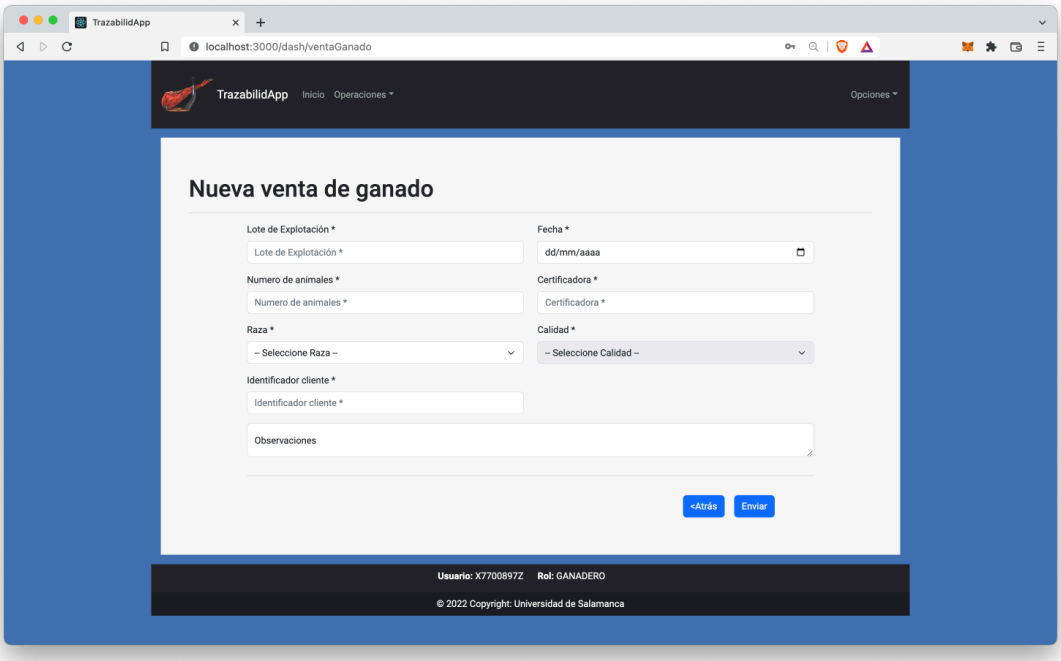

Ilustración 22. Captura de pantalla de alta de operación.

6.3.3.7 *Pantallas de modificación de datos del usuario*

Tanto para la modificación de los datos de contacto como para el cambio de contraseña del usuario el diseño será el mismo. Se mostrará únicamente una de las pantallas como ejemplo.

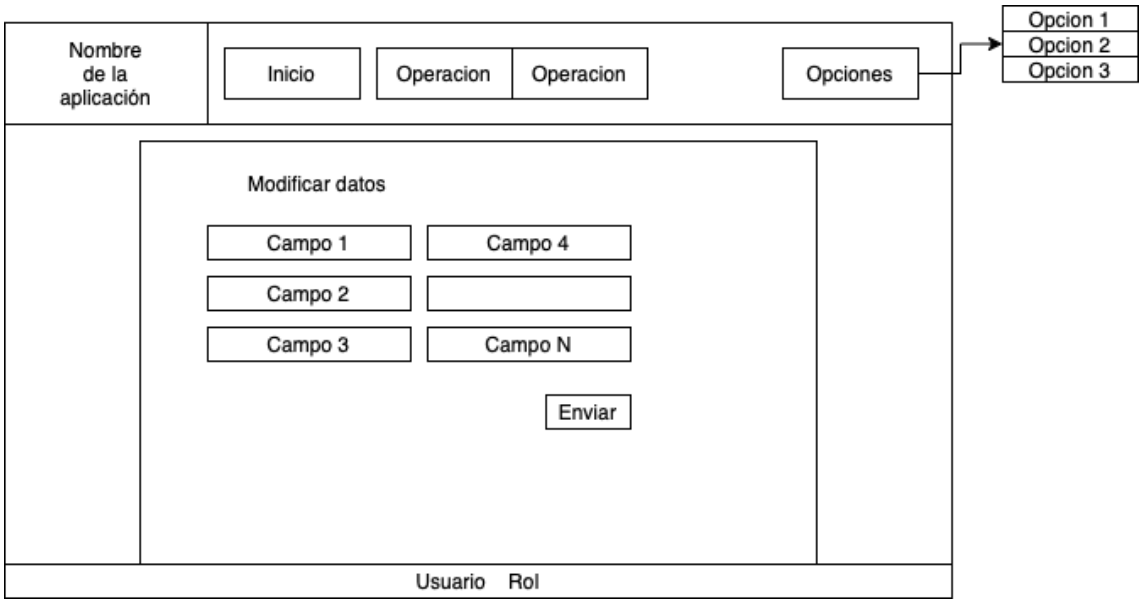

Ilustración 23. Diseño pantalla de modificación de datos.

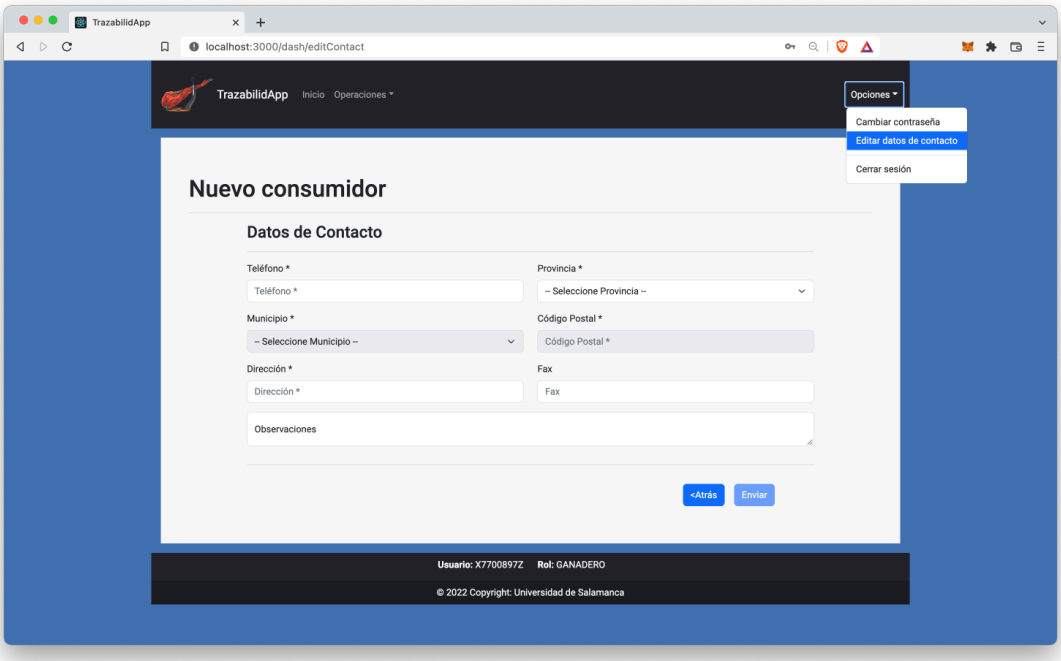

Ilustración 24. Captura de la pantalla de modificación de datos del usuario.

6.3.3.8 *Pantalla de lista de usuarios*

Esta pantalla será únicamente accesible para el administrador y le permitirá obtener un listado de todos los usuarios del sistema, junto con un botón para ver la información del usuario y otro botón para eliminarlo del sistema.

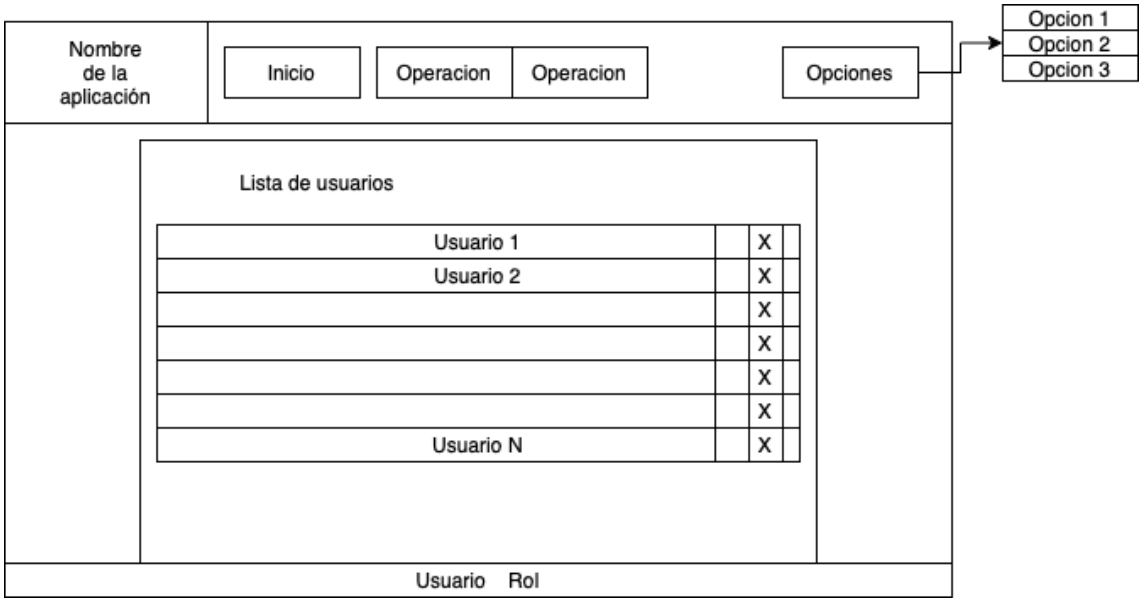

Ilustración 25. Diseño de pantalla de lista de usuarios.

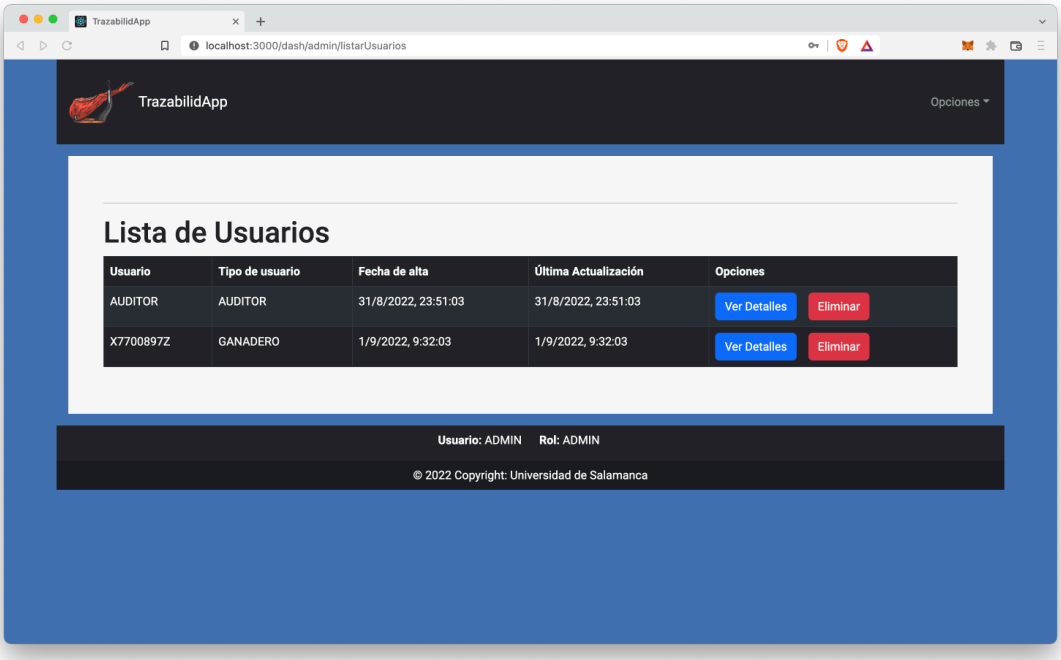

Ilustración 26. Captura de pantalla de lista de usuarios.

6.3.3.9 *Pantalla de detalle de usuario*

Esta pantalla es muy similar a la pantalla de detalle de operación.

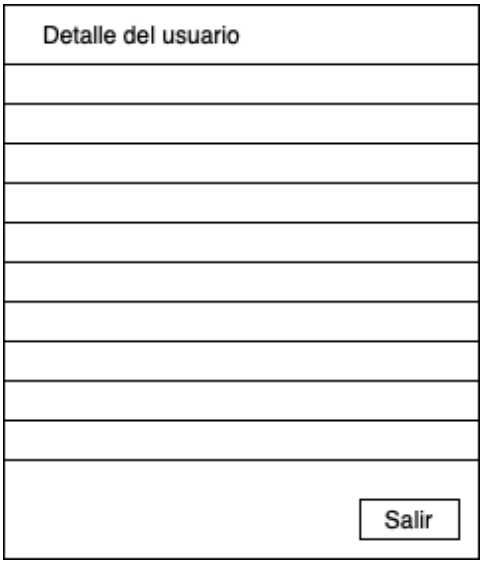

Ilustración 27. Diseño de pantalla de detalle de usuario.

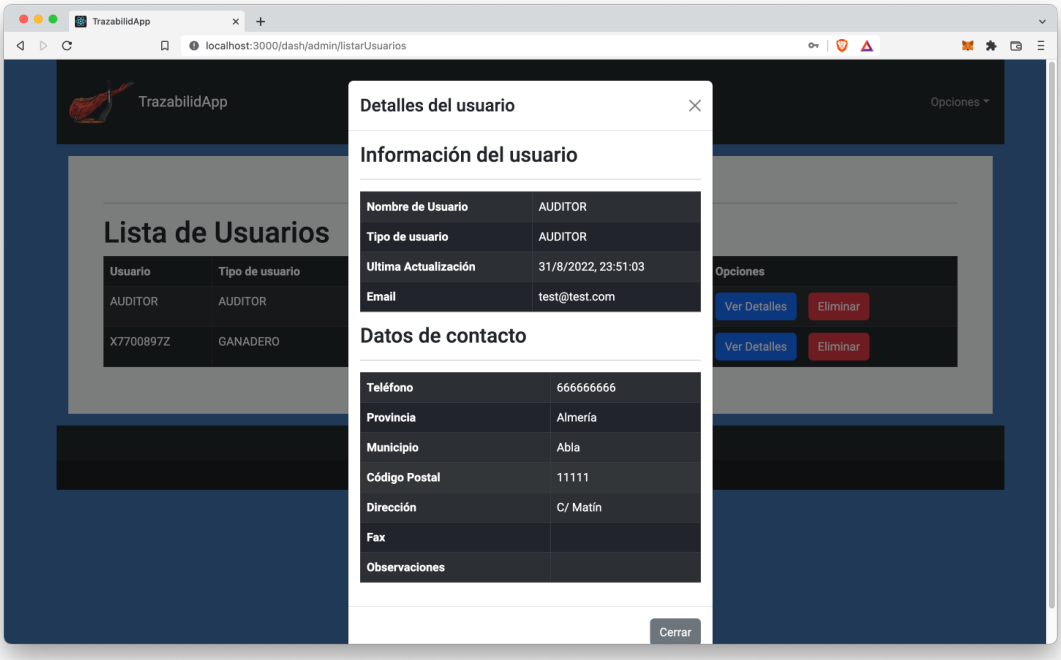

Ilustración 28. Captura de pantalla de detalle de usuario.

6.3.3.10 Securización del enrutamiento

De la misma manera que se ha aplicado una capa de seguridad en el controlador a través del token de acceso y refresco para que únicamente los usuarios con permiso puedan realizar peticiones a la API desde la interfaz gráfica, es necesario adaptarla para evitar que una persona maliciosa pueda acceder a rutas que no le corresponden. Esta seguridad se aplica teniendo en cuenta las rutas que tienen los usuarios en función de si han iniciado sesión o si su rol les permite acceder al recurso.

En caso de no haber iniciado sesión e intentar acceder a una ruta protegida, el sistema enviará directamente al usuario a la pantalla de inicio de sesión.

En caso de haber iniciado sesión e intentar acceder a un recurso que no existe o para el que no tiene permiso será redirigido a una pantalla de error.

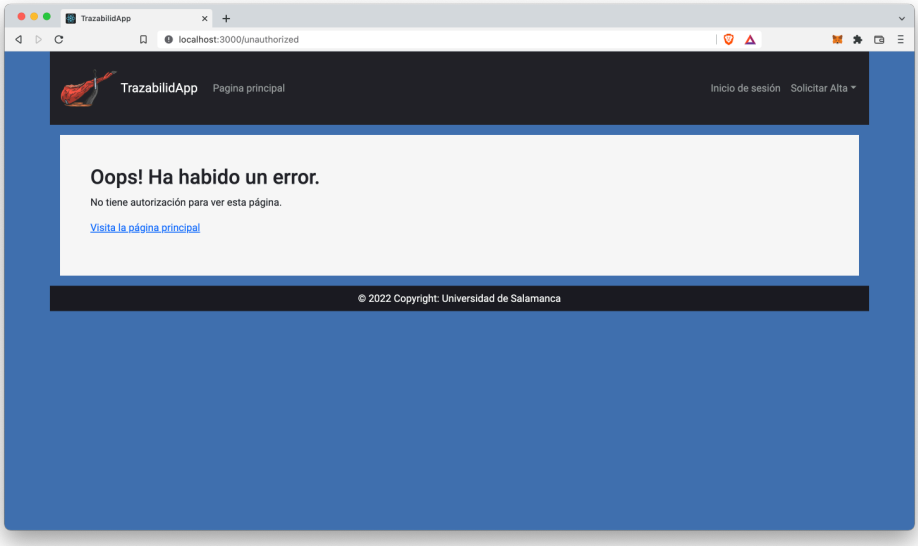

Ilustración 29. Captura de pantalla a una ruta sin autorización.

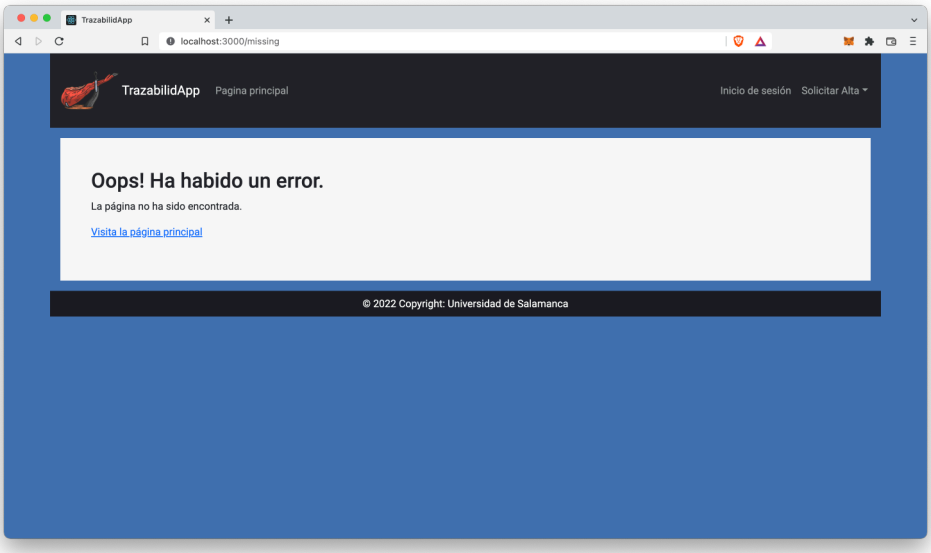

Ilustración 30. Captura de pantalla para ruta inexistente.

6.3.3.11 *Validación de la información*

Se han introducido comprobaciones en los formularios para asegurar que el usuario no introduce datos erróneos intencionalmente o por error.

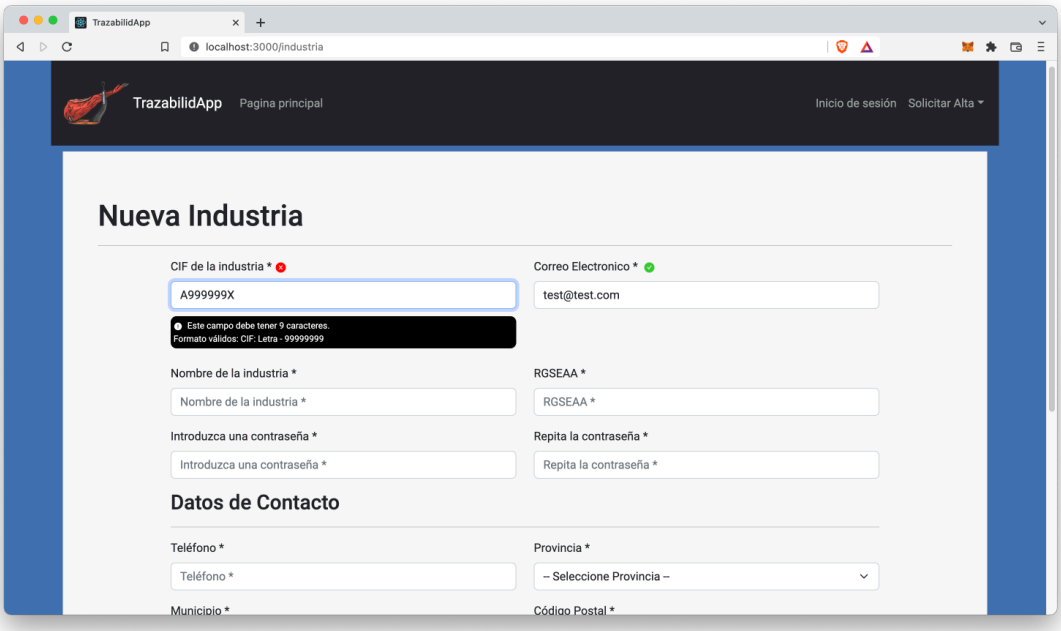

Ilustración 31. Validaciones de los campos de los formularios.

6.3.3.12 *Notificaciones para el usuario*

Siempre que el usuario realiza alguna acción recibirá un mensaje flotante indicando si el procesamiento ha sido correcto o erróneo.

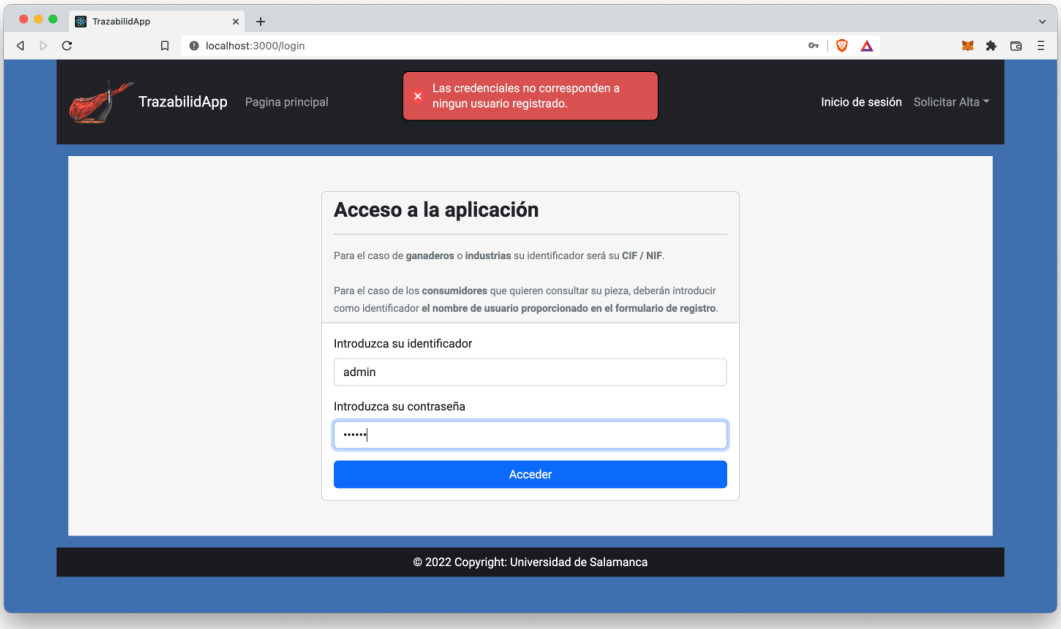

Ilustración 32. Captura de mensaje NOK.

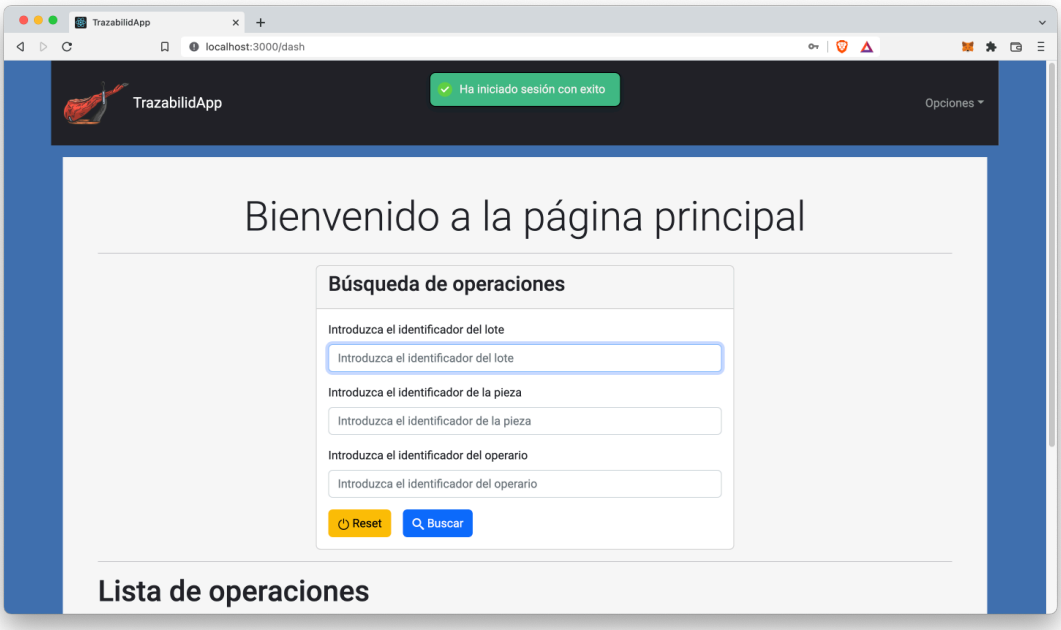

Ilustración 33. Captura de mensaje OK.

7 Conclusiones

Tras el desarrollo de este trabajo y la creación del prototipo del sistema se puede afirmar que la aplicación de blockchain a los sistemas de trazabilidad aporta muchas ventajas de cara a la seguridad e inmutabilidad de los datos, pero también tiene algunos inconvenientes.

El planteamiento que se ha seguido consistente en almacenar la información en el log de eventos no es una solución del todo válida, ya que para los casos donde es necesario que la información permanezca accesible durante un periodo prolongado de tiempo este planteamiento no servirá, debido a que los logs con el paso de los bloques se acaban perdiendo. Por otra parte, mantener toda la información en el almacenamiento del propio contrato inteligente sí aseguraría la persistencia de los datos, pero el coste por transacción aumentaría de manera desorbitada como consecuencia y en un sistema de trazabilidad el número de transacciones es elevado.

A pesar de existir mucha documentación sobre el funcionamiento de Ethereum, que actualmente es la blockchain más popular para realizar desarrollos y que cuenta con una gran comunidad, las operaciones que proporciona son muy básicas y es necesario realizar una gran adaptación para que funcione un sistema específico, lo que conlleva un gran coste para sistemas más complejos.

Se podría concluir que se han satisfecho los objetivos marcados para el trabajo, además de haber comprobado que un sistema de este tipo es viable, y se pueden aprovechar todas las ventajas que proporciona siempre que se esté dispuesto a asumir el gran coste que conlleva.

8 Líneas de trabajo futuras

Algunas de las mejoras que se podrían realizar a este sistema para que sea un prototipo viable en un entorno real o de producción serían las siguientes:

- Cada actor del sistema deberá tener su propia cartera y comunicarse directamente con la blockchain firmando él mismo sus transacciones, consiguiendo de esta manera un sistema descentralizado y distribuido.
- Realizar pruebas con usuarios reales para ajustar los requisitos y el sistema en base a las necesidades que puedan estar insatisfechas con el sistema actual.
- Mejorar la interfaz de usuario para que la introducción de información sea más segura.

9 Bibliografía

[1] «REGLAMENTO (CE) Nº178/2002 DEL PARLAMENTO EUROPEO Y DEL CONSEJO» 28 Enero 2002. https://www.boe.es/doue/2002/031/L00001-00024.pdf.

[2] «Guía para la aplicación del sistema de trazabilidad en la empresa agroalimentaria.,» 27 Julio 2019. https://www.aesan.gob.es/AECOSAN/docs/documentos/publicaciones/seguridad_alimentaria/guia_trazabilidad.pdf.

[3] H. Fernandez, «¿Qué es Blockchain? La tecnología que está revolucionando la economía global» https://economiatic.com/blockchain/.

[4] S. Nakamoto, «bitcoin.org,» 2008. https://bitcoin.org/bitcoin.pdf.

[5] «What is Ethereum?» https://ethereum.org/es/what-is-ethereum/.

[6] I. Bashir, Mastering Blockchain, Reino Unido: Packt Publishing Ltd., 2018.

[7] Z. Zheng, S. Xie, H.-N. Dai, X. Chen y H. Wang, An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends, 25/06/2017.

[8] «Intro to Ethereum» https://ethereum.org/en/developers/docs/intro-to-ethereum/#what-is-ethereum.

[9] «Consensus mechanisms: POS» https://ethereum.org/es/developers/docs/consensus-mechanisms/pos/.

[10] «Intro to Ether» https://ethereum.org/en/developers/docs/intro-to-ether/.

[11] «Coinmarketcap» https://coinmarketcap.com/currencies/ethereum/.

[12] «Solidity» https://solidity-es.readthedocs.io/es/latest/.

[13] «AltimFood» https://www.altimfood.com/sectores/.

[14] «JamonesIbericos» https://eligetuiberico.es.

[15] «MongoDB» https://www.mongodb.com.

[16] «GIT» https://git-scm.com.

[17] «Hardhat» https://hardhat.org.

[18] «NodeJs» https://nodejs.org/en/.

[19] «Ethers» https://docs.ethers.io/v5/.

[20] «ExpressJs» https://expressjs.com/es/.

[21] «ReactJs» https://es.reactjs.org.

[22] «React-bootstrap» https://react-bootstrap.github.io.

[23] «React-icons» https://react-icons.github.io/react-icons/.

[24] «React-hot-toast» https://react-hot-toast.com.

[25] «VisualStudioCode» https://code.visualstudio.com.

[26] «Diagrams» https://www.diagrams.net.

[27] «Universidad de Alicante - Modelo vista controlador (MVC)» https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.

[28] «Wikipedia: JSON Web Token» https://es.wikipedia.org/wiki/JSON_Web_Token.

[29] A. S. R. Guzmán, «TFM: Modelado y simulación de negocio eléctrico en los hogares inteligentes. Aplicación de tecnologías IoT y Blockchain».