UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de plataforma para la venta segura de entradas con blockchain

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Martí Valero, María

Tutor/a: Giménez Guzmán, José Manuel

Cotutor/a: García Valls, María Soledad

CURSO ACADÉMICO: 2023/2024

A mis padres,
por soportarme durante la realización de este trabajo.

Resumen

El fraude en las transacciones de compraventa online es, desgraciadamente, una realidad muy común en la actualidad, ocurriendo con más frecuencia de la que nos gustaría y causando pérdidas tanto a los consumidores como a los organizadores de eventos. Abordar este problema requiere la participación conjunta de diversas áreas tecnológicas, que van desde la criptografía avanzada y la autenticación digital hasta la seguridad web y la implementación de plataformas de confianza. Prevenir este tipo de fraudes es una prioridad que demanda soluciones innovadoras.

Este Trabajo Final de Grado se plantea con el objetivo de diseñar una solución para el problema actual de las estafas en la venta de entradas por internet para distintos espectáculos, abarcando tanto las ventas falsas como los posibles duplicados, mediante el uso de nuevas tecnologías ciberseguras y de desarrollo web. En concreto, se empleará la tecnología blockchain, una red descentralizada compuesta por múltiples nodos donde la información se almacena de manera inmutable. Esta característica clave del blockchain impide cualquier manipulación o modificación de los datos, proporcionando así un registro seguro y confiable de todas las transacciones. Gracias a esta tecnología, será posible registrar todas las ventas de entradas, verificar su autenticidad y posesión, y realizar transferencias de titularidad de forma segura y transparente.

Uno de los objetivos fundamentales es entender el funcionamiento de esta blockchain que nos sirve de base para el correcto funcionamiento del sistema. Para ello crearemos una cadena de bloques propia para poder controlar sus operaciones. Al crear la plataforma desde cero es posible modificar el funcionamiento y los protocolos utilizados, así como la gestión de las transacciones y la implementación de contratos inteligentes, lo que sería más complejo de realizar en plataformas de terceros.

Otro de los objetivos principales es simular un entorno real a pequeña escala donde se recrean las transacciones entre compradores y vendedores. Se creará una interfaz intuitiva y funcional utilizando HTML5, CSS y JavaScript junto con las bibliotecas necesarias para ilustrar de manera clara la operatividad del sistema. Esta plataforma permitirá la interacción del usuario con la cadena de bloques, facilitando las transacciones necesarias para operar con las entradas, y simulando un portal de venta común.

Por último, se estudiará el consumo energético del sistema desarrollado, analizando su impacto y asegurando que la tecnología sea sostenible y eficiente.

Resum

El frau en les transaccions de compravenda en línia és, desgraciadament, una realitat molt comuna en l'actualitat, i ocorre amb més freqüència de la que ens agradaria, causant pèrdues tant als consumidors com als organitzadors d'esdeveniments. Abordar aquest problema requereix la participació conjunta de diverses àrees tecnològiques, que van des de la criptografia avançada i l'autenticació digital fins a la seguretat web i la implementació de plataformes de confiança. Prevenir aquest tipus de fraus és una prioritat que demanda solucions innovadores.

Aquest Treball Final de Grau té com a objectiu resoldre el problema actual de les estafes en la venda d'entrades per internet per a diferents espectacles, abastant tant les vendes falses com els possibles duplicats, mitjançant l'ús de noves tecnologies cibersures i de desenvolupament web. En concret, s'emprarà la tecnologia blockchain, una xarxa descentralitzada composta per múltiples nodes on la informació s'emmagatzema de manera immutable. Aquesta característica clau del blockchain impedeix qualsevol manipulació o modificació de les dades, proporcionant així un registre segur i fiable de totes les transaccions. Gràcies a aquesta tecnologia, serà possible registrar totes les vendes d'entrades, verificar la seua autenticitat i possessió, i realitzar transferències de titularitat de manera segura i transparent.

Un dels objectius fonamentals és entendre el funcionament d'aquesta blockchain que ens serveix de base per al correcte funcionament del sistema. Per a això crearem una cadena de blocs pròpia per a poder controlar les seues operacions. En crear la plataforma des de zero és possible modificar el funcionament i els protocols utilitzats, així com la gestió de les transaccions i la implementació de contractes intel·ligents, cosa que seria més complex de realitzar en plataformes de tercers.

Un altre dels objectius principals és simular un entorn real a petita escala on es recreen les transaccions entre compradors i venedors. Es crearà una interfície intuïtiva i funcional utilitzant HTML5, CSS i JavaScript juntament amb les biblioteques necessàries per a il·lustrar de manera clara l'operativitat del sistema. Aquesta plataforma permetrà la interacció de l'usuari amb la cadena de blocs, facilitant les transaccions necessàries per a operar amb les entrades, i simulant un portal de venda comú.

Finalment, s'estudiarà el consum energètic del sistema desenvolupat, analitzant el seu impacte i assegurant que la tecnologia siga sostenible i eficient.

Abstract

Fraud in online buying and selling transactions is, unfortunately, a very common reality today, occurring more frequently than we would like and causing losses for both consumers and event organizers. Addressing this issue requires the joint participation of various technological areas, ranging from advanced cryptography and digital authentication to web security and the implementation of trustworthy platforms. Preventing such fraud is a priority that demands innovative solutions.

This Final Degree Project aims to address the current problem of scams in online ticket sales for various events, covering both fake sales and possible duplicates, using new cybersecurity and web development technologies. Specifically, blockchain technology will be used, a decentralized network made up of multiple nodes where information is stored immutably. This key feature of blockchain prevents any manipulation or modification of data, thus providing a secure and reliable record of all transactions. Thanks to this technology, it will be possible to record all ticket sales, verify their authenticity and ownership, and securely and transparently transfer ownership.

One of the fundamental objectives is to understand the operation of this blockchain that serves as the basis for the proper functioning of the system. To this end, we will create our own blockchain to control its operations. By creating the platform from scratch, it is possible to modify the operation and protocols used, such as transaction management and the implementation of smart contracts, which would be more complex to achieve on third-party platforms.

Another main objective is to simulate a real small-scale environment where transactions between buyers and sellers are recreated. An intuitive and functional interface will be created using HTML5, CSS, and JavaScript, along with the necessary libraries to clearly illustrate the system's operation. This platform will allow user interaction with the blockchain, facilitating the necessary transactions to operate with tickets, and simulating a common sales portal.

Finally, the energy consumption of the developed system will be studied, analyzing its impact and ensuring that the technology is sustainable and efficient.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Tecnología Digital y Multimedia debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la tecnologías digitales y multimedia

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	5
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	9-16
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	6
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	7 y 16
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	16 y 34
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	52
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	49-52

Índice

Índice de tablas

Índice de ilustraciones

Capítulo 1. Introducción

1.1 Contexto y motivación

El espectacular avance de las tecnologías de la información y las comunicaciones ha facilitado la inclusión de las actividades "online" en la mayoría de los ámbitos de nuestra vida cotidiana sin la necesidad del desplazamiento por parte del usuario para llevar a cabo la acción, por ejemplo, en las relaciones interpersonales mediante mensajes y llamadas telefónicas, el teletrabajo, el ocio gracias a contenido bajo demanda, o las compras a través de dispositivos electrónicos. Focalizando nuestra atención en este último caso, el avance tecnológico ha permitido el acceso a una amplia variedad de productos y servicios, democratizando el mercado y ofreciendo un amplio abanico de opciones tanto para los consumidores como para los vendedores.

La compra de bienes digitales tales como entradas, libros electrónicos o software, entre otros, es ahora un proceso rápido y eficiente gracias a plataformas online y sistemas de pago electrónicos. Además, por la naturaleza de estos bienes no es ni siquiera necesario ningún tipo de interacción más que la pura transacción económica de la compraventa.

Sin embargo, estas nuevas implementaciones, que a priori son una ayuda, han traído consigo un aumento en la ciberdelincuencia y las estafas. Mediante estos engaños, los estafadores pretenden obtener dinero o datos personales de sus víctimas, este último caso es denominado "phishing". El "phishing" consiste en engañar a los usuarios para que revelen información confidencial, y son muy comunes en este ámbito. Además, la venta online ha propiciado el uso de "bots" para comprar grandes cantidades de entradas de eventos, creando desigualdad de condiciones y evitando que un gran número de interesados pueda hacerse con una de las entradas originales, teniendo que recurrir a una reventa muchas veces insegura y con toda certeza a un mayor precio.

En ocasiones, estos engaños se producen por la imprudencia del comprador, quien accede por desconocimiento a páginas falsificadas que imitan portales de venta oficiales, o que realiza el proceso de compra a particulares sin garantías de obtener un producto real. Otro de los principales problemas es la facilidad con la que se pueden falsificar o duplicar las entradas, siendo numerosos los casos en los que la entrada adquirida ha sido distribuida a más de un usuario, por lo que, únicamente el primero en validar la entrada podrá garantizar su acceso. En otros casos, la entrada adquirida nunca llega al comprador.

Actualmente existen plataformas de reventa de entradas que ofrecen garantías, como por ejemplo "TicketSwap" [1], que gracias a su colaboración con los portales y establecimientos oficiales asegura la generación de una nueva entrada única y original para el comprador, además de que limita el incremento del precio en un 20% para evitar que el vendedor se aproveche de la situación de desigualdad con la persona interesada. Estos son avances muy significativos que permiten una interacción justa entre los usuarios, sin embargo, no evita que las entradas sean vendidas posteriormente por otras vías que hacen que ya no sean seguras.

Acreditar la identidad junto a la propiedad de los bienes en estas transacciones es uno de los retos para garantizar la originalidad y validez de la entrada. Las tecnologías web pueden jugar un papel crucial en la solución de estos problemas, incorporando los mecanismos necesarios para asegurar la identidad de los participantes en la transacción, la singularidad de la entrada mediante la validez en tiempo real empleando el uso de sistemas de verificación digital y forzando reglas que garanticen una justa distribución de los bienes, por ejemplo, la implementación de límites de compra por usuario, basados en la identificación única de cada comprador o el establecimiento de un registro fiable de las transacciones.

1.2 Objetivos

En esta sección se establecen los objetivos del Trabajo Fin de Grado acerca de la venta segura de entradas mediante blockchain. La definición precisa de los objetivos permite enfocar el esfuerzo hacia los aspectos más importantes del diseño y la implementación, ayudando a lograr los resultados deseados.

O1. Implementar un método para contribuir a evitar el fraude en la venta y reventa de entradas mediante nuevas técnicas de seguridad como las cadenas de bloques asociándolas a las tecnologías web necesarias.

O2. Profundizar en el funcionamiento de los principales protocolos de blockchain, entendiendo sus mecanismos de consenso y evaluando su aplicabilidad y eficiencia en el contexto de la venta de entradas.

O3. Aplicar el funcionamiento de las cadenas de bloques en un lenguaje de alto nivel, en auge y de fácil aprendizaje como es Python.

O4. Garantizar la transparencia y autenticidad en las transacciones de entradas mediante el uso de contratos inteligentes que regulen las condiciones de venta y reventa, evitando así el fraude y la falsificación.

O5. Ampliar los conocimientos teóricos y prácticos sobre programación de servidores en Python y su integración con diversas tecnologías web como HTML5 y Flask.

O6. Implementar una plataforma web intuitiva y fácil de usar que proporcione todas las funcionalidades necesarias para la gestión de entradas.

O7. Realizar simulaciones en un entorno real a pequeña escala que reproduzcan las transacciones entre compradores y vendedores, así como el comportamiento general del sistema.

O8. Registrar en la cadena de bloques las transacciones de compra que realice un usuario a través de la interfaz mediante la interacción entre la plataforma web y el servidor blockchain.

O9. Estudiar el impacto energético mediante análisis y mediciones del consumo de la plataforma web y la blockchain empleada en el proyecto.

O10. Establecer una planificación detallada del trabajo que incluya todas las etapas de desarrollo del proyecto, desde la fase de investigación hasta la implementación y prueba del sistema.

O11. Desarrollar habilidades de resolución de problemas para identificar y solucionar posibles desafíos técnicos que surjan durante la realización del proyecto.

1.3 Vista general de la solución propuesta

Para regular el problema de la identificación y validación de la entrada, así como el registro fiable de las transacciones se pretende hacer uso de la "Blockchain", una tecnología cuya principal característica es llevar el registro de las transacciones de los usuarios y no requiere ser gestionada por un tercero. Permitirá registrar cada entrada vendida en la cadena y regular el proceso de reventa mediante un contrato implementado en la misma.

La solución global que se propone para tratar de solucionar este problema en la venta y reventa de entradas es la implementación de una plataforma web que registre todas las transacciones de compra de entradas de los usuarios, junto con su identificación, y además la blockchain regulará los contratos de reventa, garantizando el cumplimiento de los requisitos y la seguridad en el proceso.

La compra podrá ser tanto de entradas originales lanzadas a la venta por la organización como la venta particular de alguno de los usuarios de esas mismas entradas posteriormente. Las entradas para los eventos únicamente podrán distribuirse a través de la aplicación y no podrían adquirirse por ningún otro canal, por lo que cualquier entrada que se ofrezca por otro medio nunca será válida. Mediante la blockchain y su registro de transacciones se garantiza la compra y la pertenencia de cada entrada. Las entradas únicamente se mostrarán a su usuario correspondiente cuando todos los procesos en la cadena de bloques hayan sido validados de manera exitosa, por lo que, la posesión de la entrada garantiza su validez y autenticidad.

Para garantizar la integridad y evitar la posibilidad de reventa por medios no fiables, las entradas no poseerán ningún código visible que permita su distribución y las prácticas negligentes y fraudulentas. El registro de la transacción en la cadena y la interacción de la plataforma con el servidor serán suficientes para garantizar la veracidad de la entrada y la fiabilidad del proceso de adquisición, y el identificador del registro en la blockchain, que se gestionará de manera interna, permite la gestión por parte del usuario sin necesidad de ser conocedor de este.

Para poder certificar su uso una vez llegado el momento en el recinto del evento, sería necesario una aplicación con sistema de verificación personal. Unos minutos antes de la apertura de puertas el sistema debe desbloquear la opción de validar las entradas. En el acceso al recinto, el responsable situado en la puerta deslizará, en la pantalla del dispositivo móvil personal, la entrada adquirida, teniendo que hacer una acción para validar la entrada, similar a la validación de transacciones en las aplicaciones bancarias. Esta acción no se podrá realizar por segunda vez y es imprescindible que ocurra para poder entrar, pues capturas de pantalla no serían validas.

Además, para evitar la venta de contraseñas y de accesos, la sesión de cada usuario en la aplicación solo podrá estar activa en un único dispositivo. Si se pretende cambiar de dispositivo, la sesión en el previo ha de estar cerrada o se podrá cerrar remotamente desde la nueva ubicación. Esta metodología de registro de IPs ha comenzado a emplearse recientemente en las plataformas de "streaming" para evitar que diferentes usuarios puedan acceder a una misma cuenta y disfrutar del contenido simultáneamente.

1.4 Fases de realización

En esta sección se exponen las diferentes fases que componen el desarrollo del proyecto presentado. Cada fase representa un conjunto específico de tareas que han sido llevadas a cabo en un orden determinado para lograr una implementación exitosa del sistema.

Fase 1. Especificación de los requisitos del sistema.

Fase 2. Diseño de alto nivel de los componentes del sistema.

Fase 3. Desarrollo de un prototipo de una plataforma blockchain.

Fase 4. Desarrollo del método para registrar transacciones desde el Front-End.

Fase 5. Desarrollo de la interfaz de usuario.

Fase 6. Desarrollo del Back-End de la interfaz.

Fase 7. Implementación del contrato inteligente.

Fase 8. Integración de la cadena de bloques con la plataforma web.

Fase 9. Pruebas de ejecución una vez realizada la integración.

Fase 10. Evaluación del consumo energético.

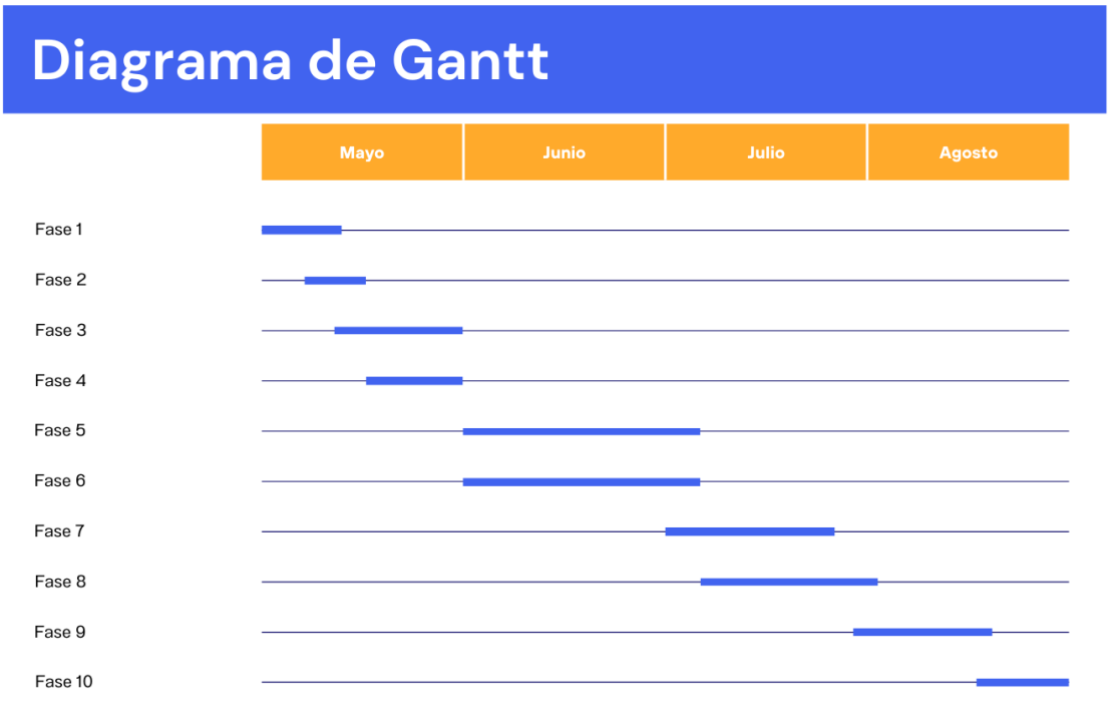

Ilustración 1. Diagrama de Gantt de las fases de realización.

Capítulo 2. Tecnologías relacionadas

2.1 Tecnologías Blockchain

2.1.1 ¿Qué es una cadena de bloques?

Una cadena de bloques (en inglés blockchain) [2], es una red descentralizada cuya información se almacena en distintos nodos ligados entre sí formando una línea temporal, dificultando la modificación de la información, ya que para llevar a cabo la edición o falsificación de los datos almacenados haría falta modificar todos los bloques anteriores. Se puede entender tal y como cita la Cámara de Comercio de Valencia [3]: "Es un libro electrónico público que se puede compartir abiertamente entre usuarios dispares y que crea un registro inmutable de sus transacciones."

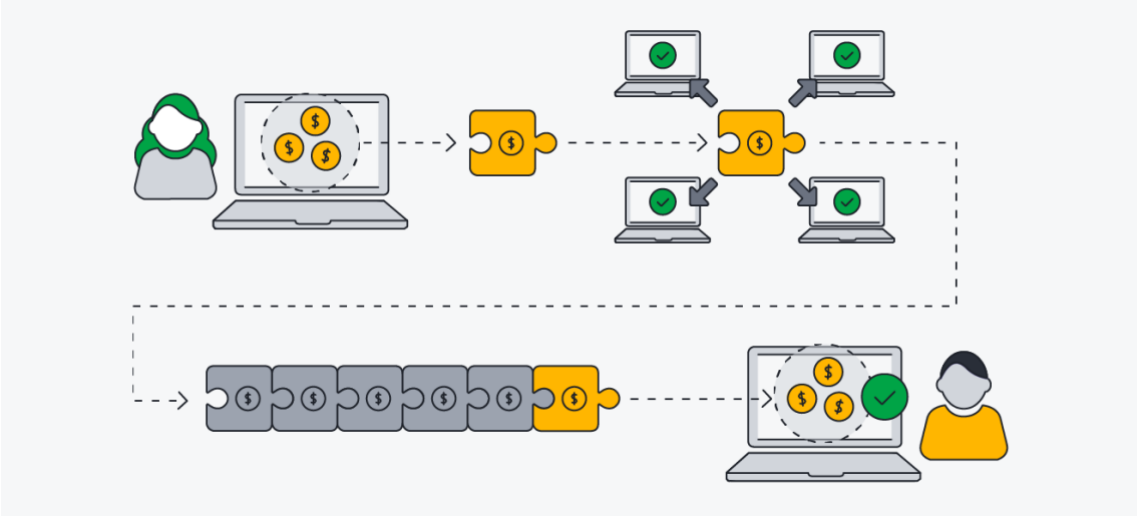

Ilustración 2. Dibujo del funcionamiento de una Blockchain.

Fuente: AVG [4]

2.1.2 Características de una blockchain

Una cadena de bloques es considerada segura debido a sus características principales, que son las siguientes:

- Red distribuida: La información es compartida y cada participante posee una copia en su dispositivo que se actualiza en tiempo real.
- Inmutabilidad: Una vez se introduce información en la cadena, no puede ser alterada ni modificada.
- Trazabilidad: Los bloques de la cadena están relacionados entre sí, por lo que se puede rastrear toda la información registrada en la blockchain.
- Consenso: Para llevar a cabo las transacciones se establecen unas reglas sobre el consentimiento de los participantes, solo se puede realizar un nuevo registro cuando la mayoría de sus participantes dan su consentimiento.

2.1.3 Tipos de blockchain

Existen diferentes tipos [5] de cadenas de bloques dependiendo de su estructura y características:

- Blockchain pública: Se trata de una cadena abierta y descentralizada en la que cualquier usuario puede unirse a la red, participar en la validación de transacciones y acceder a su historial.
- Blockchain privada: Está controlada por una organización y restringida a un grupo específico de participantes que necesita autorización para poder acceder.
- Blockchain de consorcio: En este caso son varias organizaciones las que controlan la cadena de bloques y las decisiones las toman los miembros de estas entidades. Este tipo de blockchain es más flexible en términos de descentralización manteniendo a su vez un cierto control entre los participantes.
- Blockchain híbrida: Está compuesta tanto por áreas públicas, abiertas a todos los usuarios como por áreas de acceso restringido, lo que las convierte en una mezcla entre una blockchain de tipo pública y otra privada.
- Blockchain permisiva: Los usuarios que quieran acceder para participar y validar transacciones necesitan obtener permiso mediante sistemas de identificación y autenticación.

2.1.4 Protocolos de consenso

Los protocolos de consenso, mencionados en [6], son un mecanismo para validar transacciones velando por la integridad de la cadena. Mediante este método se pretende brindar transparencia detectando actitudes fraudulentas por parte de algún usuario y beneficiar la honestidad.

Los protocolos más conocidos son PoW (Proof of Work) y PoS (Proof of Stake).

Proof of Work [7] fue el primer protocolo, propuesto por Satoshi Nakamoto para bitcoin, aunque fue creado en los años 90 para evitar los ataques de spam en el correo electrónico. En este protocolo, la validez de un bloque se determinará mediante un cálculo matemático basado en la prueba y error. Este cálculo es resuelto por los mineros mediante la función hash utilizando una gran cantidad de poder computacional al realizar múltiples intentos hasta lograr la operación requerida. Una vez se resuelve el problema se obtiene el resultado, la creación del bloque y la validación de la transacción correspondiente.

Por su parte, en el protocolo Proof of Stake [8], no se realizan operaciones matemáticas para validar un bloque. En este caso se emplean “Validadores”, nombre que se le da a los nodos participantes en este protocolo. La elección del nodo que validará el bloque se realiza dando prioridad a aquellos que tengan más cantidad de moneda reservada o a quienes más participen. Si un validador actúa de manera deshonesta puede perder su participación como castigo. Este protocolo consume menos energía en comparación con PoW, mejorando la eficiencia energética y la escalabilidad de la red.

2.1.5 Hash

Un hash [9] es una función criptográfica que crea una cadena de identificación única e irrepetible a partir de los datos que son proporcionados. Son utilizadas en la tecnología blockchain para garantizar la autenticidad de los datos. Mediante el procesamiento matemático y lógico de la información se pretende obtener una cadena de longitud fija cuya peculiaridad es su proceso de creación unidireccional, es decir, es simple de crear, pero realizar el proceso inverso y obtener los datos correspondientes a partir del hash es imposible pues requeriría miles de años empleando supercomputadoras. Además, la similitud de la información introducida no resulta en la creación de hashes parecidos, cualquier mínimo cambio da lugar a un hash totalmente distinto.

2.1.6 Minado

Minar [10] consiste en obtener un hash que cumpla con los requisitos de dificultad establecidos por la cadena de bloques. Para generar una nueva transacción, tanto el índice del bloque, como el contenido y el hash anterior son inmutables, por ello se ha de utilizar un valor modificable que permita lograr un hash válido. Este valor se llama Nonce y únicamente se utiliza para este proceso de minado.

Este proceso es complejo y costoso, sin embargo, comprobar posteriormente la validez del hash es sencillo ya que basta con comprobar que el inicio coincide con el número de ceros establecido.

2.1.7 Limitaciones

La operación descentralizada de la blockchain ayuda a que cualquier tipo de ataque sea difícil de ejecutar, sin embargo, no es imposible que suceda. A continuación, se explican algunos de los posibles ataques [11]:

- Ataque Sybil: Este ataque consiste en la creación de múltiples identidades falsas por parte de una entidad para manipular los resultados de las votaciones, bloquear transacciones legitimas o controlar recursos. Para prevenir este ataque se utilizan diferentes técnicas como la verificación de identidad o la implementación de consensos basados en pruebas de trabajo.
- Ataque de denegación de servicio (DoS): Los atacantes sobrecargan los servidores con numerosas solicitudes inválidas para consumir todos sus recursos de procesamiento e impedir que se ofrezca el servicio a los usuarios.
- Ataque del 51%: El atacante controla más del 51% de la potencia de procesamiento de la red y le permite revertir transacciones previas o añadir transacciones nuevas.

Otra de las limitaciones de una cadena de bloques recae en el rendimiento y los riesgos y consecuencias a los que pueda dar lugar. La escalabilidad es uno de los problemas principales, cuando más grande es una cadena, más compleja de gestionar es. Esto puede provocar que el tiempo para registrar una transacción aumente. También se ha de tener en cuenta el elevado coste energético y computacional que genera la tecnología blockchain, sobre todo en el proceso de la prueba de trabajo, como se menciona en el estudio [12].

2.1.8 Ethereum

Ethereum [13] es una plataforma digital basada en una blockchain descentralizada pública cuya característica principal es que permite almacenar código, desarrollar aplicaciones y ejecutar programas que utilicen una cadena de bloques junto con las transacciones, a diferencia de otras cadenas de bloques cuya única finalidad es la transacción de criptomonedas. Ethereum está sustentado por una red de nodos que descargan la cadena y hacen cumplir las reglas de consenso. Ethereum permite a los desarrolladores programar contratos inteligentes, que se ejecutan en la máquina virtual de Ethereum, aislada de la red principal. Permite la ejecución del contrato sin interferir con otros programas. Para medir el coste computacional utilizado al ejecutar operaciones se introduce el concepto de “gas”. Ethereum tiene su propia criptomoneda llamada “ETH”, que se emplea no solo como medio de intercambio sino como incentivo para los usuarios participantes.

2.1.9 Blockchain en Python

Con el auge de las tecnologías blockchain, el lenguaje Python se ha postulado como una herramienta fundamental en el desarrollo de aplicaciones sobre cadenas de bloques [14], debido a su versatilidad y popularidad. Python es conocido por su capacidad de trabajar juntamente con otros lenguajes, y su facilidad de aprendizaje permite un rápido desarrollo y la aportación de mejoras por parte de la comunidad. Además, cuenta con una amplia variedad de frameworks sobre los que operar, por ejemplo web3.py [15]. Web3.py es una librería de Python para interactuar con Ethereum y los contratos inteligentes programados en Solidity. Python también permite la

creación de nodos Ethereum con librerías "pyethereum" o crear tu propia cadena de bloques personalizada.

2.1.10 Contratos inteligentes

Los contratos inteligentes, tal y como se menciona en [16], son programas que se ejecutan automáticamente, sin la necesidad de terceros, según las partes involucradas cumplan ciertas condiciones predefinidas, y los cuales se almacenan en la cadena de bloques. Estos contratos eliminan la necesidad de intermediarios garantizando la ejecución precisa y segura de las transacciones, reduciendo el riesgo de errores o su manipulación. Su implementación en la blockchain asegura que los contratos sean inmutables y verificables por todos los usuarios involucrados. Los contratos permiten llevar a cabo procesos como transacciones bancarias o la gestión de cadenas de suministro.

2.1.11 Solidity

Solidity [17] es un lenguaje de programación orientado a objetos similar a C++, JavaScript o Python especialmente diseñado para implementar contratos inteligentes en plataformas blockchain, como Ethereum. Su principal característica es su capacidad para interactuar directamente con la Máquina Virtual de Ethereum, facilitando el desarrollo de aplicaciones descentralizadas.

2.1.12 Contrato en Python

Aunque Solidity es el lenguaje más común para escribir contratos inteligentes existen otras alternativas, como por ejemplo JavaScript o Python. Python no es un lenguaje nativo para contratos, pero se puede emplear junto con web3.py. Para escribir scripts y realizar pruebas de estos contratos en lenguaje Python se pueden emplear frameworks como Brownie.

2.2 Tecnologías web y la web 3.0

2.2.1 HTML5

HTML5 [18] es la quinta versión del lenguaje de marcado de hipertexto (HyperText Markup Language), utilizado para definir la estructura, diseño y contenido de páginas web y aplicaciones. La evolución y principal diferencia respecto a su anterior versión, HTML4, es la inclusión de nuevas etiquetas semánticas que facilitan la estructura de los documentos, ayudando a la indexación de los buscadores. Además, permite añadir audio y video a través de etiquetas multimedia, así como fórmulas matemáticas.

HTML5 utiliza el protocolo de transferencia de hipertexto, más conocido como HTTP, para la comunicación entre clientes y servidores. Se trata de un protocolo de solicitud/respuesta en el que el cliente envía una solicitud HTTP sobre una conexión TCP entre cliente y servidor. Así pues, cuando cliente y servidor tienen una conexión TCP establecida, intercambien mensajes según el protocolo HTTP.

La creación de páginas web empleando HTML5 tiene múltiples ventajas:

- Es compatible con la mayoría de los navegadores más utilizados y mejora la experiencia de usuario, adaptándose a distintos dispositivos.
- Es un lenguaje interpretado por numerosos sistemas informáticos.
- Es posible introducir elementos complejos como mapas interactivos o dibujos 3D gracias al sistema API de JavaScript.
- Ofrece una mayor seguridad a las páginas que cuentan con este sistema.

2.2.2 CSS

CSS [19], cuyas siglas hacen referencia a Cascading Style Sheets, es un lenguaje de estilos en cascada, de base OpenWeb utilizado en documentos HTML o XML y que ha sido estandarizado por W3C. Este lenguaje describe el diseño y la apariencia de una página web y su atributo "en cascada" refiere a que las hojas de estilos pueden heredar propiedades de otras.

El uso del lenguaje CSS permite separar los estilos del código HTML, manteniendo este limpio y estructurado, mejorando la legibilidad y la accesibilidad a la hora de realizar modificaciones. Se pueden encontrar en el mismo archivo HTML en la etiqueta "styles", o bien importarlo desde un archivo externo con la extensión ".css" También permite aplicar estilos, que, una vez definidos, se aplican a múltiples elementos del proyecto.

Las posibilidades que ofrece este lenguaje son muy amplias permitiendo un control preciso sobre los elementos. Todas las reglas CCS requieren de un selector, este puede ser la propia etiqueta del elemento, su clase o su identificador. Tras el selector, se indican las declaraciones entre dos corchetes y cada declaración consta de una propiedad y un valor.

2.2.3 JavaScript

JavaScript [20] es un lenguaje de programación interpretado y orientado a objetos. Su particularidad es que opera en el lado del cliente, es decir, se ejecuta en el navegador del usuario permitiendo la creación de páginas web dinámicas mediante la implementación de funciones interactivas, así como la validación de formularios, la manipulación de la interfaz de plataforma conocida como DOM (Document Object Mode) y la gestión de eventos. El propio lenguaje de JavaScript también cuenta con un framework llamado web3.js [21] para poder interactuar con Ethereum y los contratos inteligentes de Solidity.

Debido a su popularidad se creó Node.js [22], un entorno de JavaScript que se ejecuta en la parte del servidor en lugar de funcionar en el navegador del cliente. Esto proporciona una serie de ventajas en el ámbito del desarrollo de aplicaciones web. El uso de un servidor en JavaScript facilita el desarrollo y mantenimiento de las aplicaciones, pues Front-End y Back-End utilizan el mismo lenguaje. Además, permite manejar diferentes conexiones de manera simultánea para servicios en tiempo real y su entorno de ejecución es más amplio y flexible, pues no está restringido por el navegador.

2.2.4 Python

Python [23] es un lenguaje de programación de código abierto y orientado a objetos, creado por Guido van Rossum en 1991. Se trata de un lenguaje interpretado ya que se convierte en bytecode para ser posteriormente ejecutado. Es uno de los lenguajes preferidos para comenzar la introducción en la programación debido a su sencillez y versatilidad en comparación a otros lenguajes y se utiliza tanto en la programación de software, aplicaciones o desarrollo web ya que su funcionalidad abarca numerosos ámbitos. Una de las principales ventajas de Python es su extensa colección de bibliotecas y frameworks, como por ejemplo Django [24] y Flask [25] para el desarrollo web. Además, en la actualidad prácticamente todos los sistemas operativos tienen su propio intérprete de Python.

Python se considera un lenguaje de alto nivel debido a que tiene en cuenta las capacidades cognitivas humanas en lugar de las del procesador de datos. Está orientado a objetos, es decir, prioriza su atención en objetos o conjuntos de datos en lugar de en funciones, aunque estas también se utilizan y son importantes. Otra de sus principales características es que las variables pueden tomar valores de diferentes tipos.

2.2.5 JSON

JavaScript Object Notation, más conocido como JSON [26], es un formato ligero de intercambio de datos, ya que es simple de escribir para los humanos y simple de interpretar para las máquinas. Está basado en la sintaxis de JavaScript, pero no se emplea para el desarrollo de programas sino para el almacenamiento e intercambio de datos de manera organizada para facilitar su búsqueda y acceso.

Un objeto JSON está constituido por una lista ordenada de valores y los correspondientes pares nombre-valor. El nombre ha de ser un string mientras que el valor puede tomar puede ser string, número, objeto, array, booleano o null.

Debido a la creciente popularidad del uso de archivos JSON gracias a su simplicidad, muchos de los lenguajes de programación son compatibles con este formato.

Capítulo 3. Diseño del sistema propuesto

3.1 Requisitos del sistema

En esta sección se lista el conjunto de los requisitos funcionales que el sistema a implementar debe cumplir para un correcto funcionamiento del Trabajo Fin de Grado presentado, asegurando que todas las necesidades del usuario final sean atendidas de manera eficiente. Cada requisito ha sido definido en base a los objetivos del proyecto y las necesidades a satisfacer, y son los siguientes:

R1. El sistema debe permitir que se vendan entradas para eventos proporcionadas por un distribuidor oficial autorizado.

R2. La plataforma web debe permitir que un usuario realice la compra al distribuidor oficial de entradas, así como a usuarios que la quieran revender.

R3. La venta de entradas oficial debe estar distinguida de la compra de entradas de reventa.

R4. Solo se puede comprar una entrada por transacción, sin perjuicio de que en un futuro el sistema permita compras múltiples por un usuario para varias personas.

R5. El sistema debe permitir que un usuario adquiera una entrada siempre y cuando haya disponibilidad de entradas.

R6. Solo aparecerán en la opción de reventa aquellas entradas que hayan sido puestas en venta por algún usuario.

R7. Cuando un usuario ha solicitado una entrada de reventa, se bloquea la posibilidad de compra de esa entrada para el resto de los usuarios.

R8. El importe de la entrada de reventa no puede aumentar. El precio deberá ser el mismo que el de la compra original.

R9. La reventa de entradas será regulada por un Contrato Inteligente implementado en la blockchain.

R10. Las entradas no deben poseer ningún código visible que pueda distinguir la entrada y permitir su distribución y su validación por un medio diferente a la plataforma implementada.

R11. Cada usuario debe poder registrarse una única vez en la plataforma e iniciar sesión cada vez que sea necesario.

R12. El sistema debe estar diseñado para ser escalable, de modo que pueda manejar un gran número de usuarios y transacciones simultáneamente sin comprometer el rendimiento.

R13. La plataforma únicamente debe mostrar al usuario las entradas que hayan sido registradas en el sistema a su nombre.

R14. Cualquier transacción de compra de entradas quedará registrada en la cadena de bloques y será trazable.

R15. La interfaz será intuitiva y tendrá un menú para poder acceder a todas las pestañas en cualquier momento.

3.2 Arquitectura de alto nivel

El sistema propuesto consta de una plataforma web que interactúa con un servidor blockchain. El servidor gestiona las transacciones que se añaden a la blockchain y la plataforma web actúa como la interfaz a través de la cual los usuarios interactúan con sus entradas y su registro en la blockchain. El servidor está implementado en Python, y la plataforma web utiliza HTML, CSS, y JavaScript con el motor de plantillas Jinja2 para integrar Python con HTML.

El sistema implementado está diseñado para facilitar la interacción entre los usuarios y la blockchain. Los usuarios interactúan con el sistema a través de una interfaz web intuitiva, que envía solicitudes a los contratos inteligentes desplegados en la cadena de bloques. Estos contratos son responsables de la ejecución de la lógica, asegurando que todas las transacciones sean verificadas y procesadas de manera segura.

Los componentes principales del sistema son los siguientes:

- Servidor Blockchain: La blockchain, implementada en Python, es el núcleo del sistema. Proporciona un entorno descentralizado y seguro para la ejecución de contratos inteligentes, garantizando la inmutabilidad y la transparencia de las transacciones.
- Contratos Inteligentes: Los contratos inteligentes en este sistema han sido implementados utilizando Python, y se utilizan para automatizar la ejecución de acuerdos al verificar y aplicar las condiciones previamente establecidas, asegurando que las transacciones en la blockchain se realicen de manera segura y sin intermediarios.
- Front-End (Interfaz de Usuario): La interfaz de usuario del sistema ha sido desarrollada utilizando tecnologías web como HTML5, CSS y JavaScript. Esta interfaz permite a los usuarios interactuar con la blockchain de manera sencilla, proporcionando formularios para la entrada de datos y botones para la ejecución de transacciones. La interfaz se comunica con la cadena de bloques y los contratos inteligentes a través del framework de Flask, facilitando la interacción desde el navegador del usuario.
- Back-End (Servidor de la Interfaz): El back-end del sistema, implementado también en Python, actúa como un intermediario entre el Front-End y la blockchain. Aunque gran parte de la lógica de negocio reside en los contratos inteligentes, el Back-End gestiona tareas adicionales como la autenticación de usuarios, la validación de datos antes de enviarlos a la blockchain, y la gestión de respuestas del sistema.
- Base de datos: La base de datos del sistema se gestiona utilizando documentos JSON, facilitando el almacenamiento y la manipulación de datos de manera flexible y eficiente. Los documentos JSON se integran perfectamente con el resto de los componentes.

Capítulo 4. Diseño de la plataforma web

El Front-End contiene los archivos HTML que presentan el diseño visual de la interfaz. Es la parte visible de la aplicación, codificada en archivos de lenguaje HTML y que utiliza Jinja2 como su motor de plantillas, lo que permite la integración sencilla de Python con HTML. Además, mantiene la lógica de la aplicación separada del diseño de la interfaz de usuario, haciendo que el código sea más limpio. Cada pestaña de la aplicación web tiene su propio archivo con los elementos que la componen, así como los estilos CSS y el código en JavaScript para introducir funciones.

En el Back-End de la plataforma web se encuentran las funciones con las que interactúa la interfaz para recibir datos, redirigir pestañas o comunicarse con el servidor. En caso de que sea necesario registrar una transacción o se cumpla alguna de las condiciones del contrato, las rutas del Back-End de la interfaz son los que llamarán a los métodos correspondientes del servidor mediante el marco Flask para que se lleve a cabo la acción correspondiente.

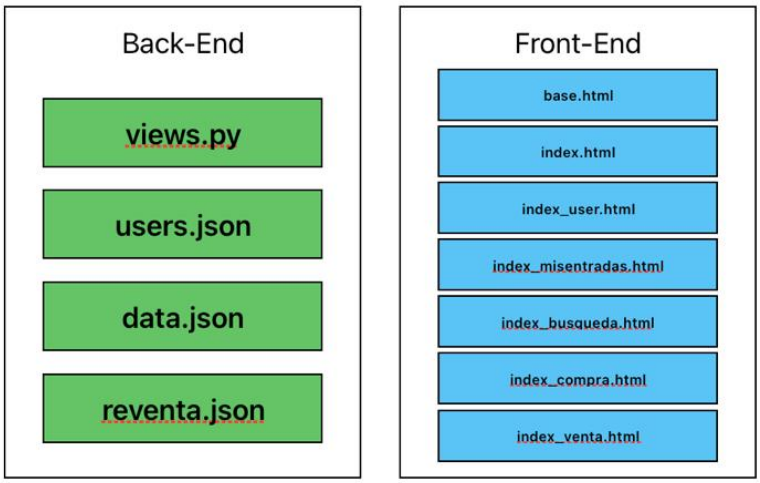

Ilustración 3. Componentes de la Plataforma Web con sus correspondientes archivos.

Ilustración 4. Rutas del Back-End de la Plataforma.

4.1 Archivos de plantillas HTML

A continuación, se explica brevemente qué podremos observar y qué funcionalidades tiene cada una de las plantillas de la interfaz:

Nombre del archivo	Descripción
base.html	Documento que incluye el menú lateral y la barra de búsqueda. Sirve como plantilla para el resto de las pestañas en el espacio reservado para éstas.
index.html	Pestaña que se recoge y permite el despliegue completo del menú. Sirve como índice de la plataforma.
index_user.html	Página de inicio de sesión donde el usuario puede registrarse o acceder con un perfil ya existente.
index_misentradas.html	Pestaña que muestra las entradas del usuario.
index_busqueda.html	Ventana que se despliega en la que aparecerán entradas a la venta. Tiene dos botones que permiten seleccionar entre entradas oficiales o entradas puestas en reventa por otros usuarios. Si por el contrario no hay resultados o no se ha introducido ningún valor en la barra de búsqueda, también se indicará por pantalla.
index_compra.html	Pantalla en la que se presenta un formulario para que el usuario rellene sus datos a la hora de comprar la entrada, tanto para la venta como para la reventa. Los formularios serán distintos en función de si se trata de compra original o de reventa.
index_venta.html	Pestaña que tiene en su parte superior un botón que despliega las entradas que el usuario posee y puede añadir a la venta. Si ya existiera alguna puesta en reventa aparecerá también junto con su estado y permitirá aceptar la solicitud de compra.

Tabla 1. Archivos de plantillas de la interfaz.

4.2 Variables del Back-End de la plataforma web

El Back-End de la interfaz cuenta con variables globales comunes en todas las funciones que a continuación se explican:

Nombre de la variable	Columna 2
user	Almacena el usuario que ha iniciado sesión en la plataforma web.
previo	Almacena el estado de la pestaña de la que viene el usuario. Puede ser 1 o 0.
actual	Almacena el estado de la pestaña en la que se encuentra el usuario. Puede ser 1 o 0.
template	Almacena la ruta de la pestaña en la que se encuentra el usuario
last_entry	Almacena el valor de la última búsqueda del usuario y el tipo de compra.

Tabla 2. Variables del Back-End de la plataforma web.

4.3 Funciones del Back-End de la plataforma web

Para el funcionamiento de los métodos de la interfaz y su correcto despliegue y apariencia necesitamos hacer uso de dos funciones:

Nombre de la función	Explicación
estado(previo, actual)	La función "estado" la utilizamos para realizar una animación en la apertura del índice y saber si ha de estar recogido o desplegado ocupando toda la pantalla. Funciona con una serie de condiciones if dependiendo de los valores de previo y actual.
search_word(users, user_form, path="")	Esta función se utiliza para buscar el usuario introducido en el archivo users.json.

Tabla 3. Funciones del Back-End de la plataforma web.

4.4 Métodos del Back-End de la plataforma web

También tenemos una breve explicación de cada una de las rutas que contiene el Back-End de la plataforma y para qué sirve cada una de ellas:

Nombre de la ruta	Descripción
/	Devuelve al usuario al índice de la interfaz.
/sesion	Lleva al usuario a la página de inicio de sesión, donde se puede registrar o acceder con un usuario previamente creado.

/user	Recibe el valor del formulario de la página de inicio de sesión. Dependiendo de si el usuario se registra, inicia sesión o cierra sesión, realiza las acciones pertinentes.
/mis_entradas	Si el usuario no ha iniciado sesión, redirige al usuario a la página de inicio de sesión. En caso contrario, obtiene las entradas asociadas al perfil y las muestra por pantalla.
/search	Obtiene el valor introducido en la barra de búsqueda y muestra los resultados existentes que coinciden en el archivo "data.json", si es de compra, o en el archivo "reventa.json" si es de reventa.
/compra	Una vez se selecciona una entrada para ejecutar la compra, redirige al usuario a la pestaña para realizar la transacción.
/submit	Envía los valores del usuario introducido y la entrada que se desea adquirir a la blockchain para registrar la transacción. Si la respuesta es válida asocia la entrada al usuario que ha realizado la compra y crea el contrato para realizar la transacción de reventa en caso necesario.
/venta	Redirige al usuario a la ventana de reventa de entradas, donde se pueden añadir entradas a la venta y gestionar las solicitudes del resto de usuarios.
/puesta_venta	Una vez el usuario selecciona una entrada para ponerla en venta, envía la solicitud a la blockchain que modifica el estado del parámetro "is_for_sale" del contrato a True.
/solicitar	Si se ha solicitado una entrada en la pestaña de compra de entradas de reventa, el formulario no registra la transacción en la blockchain sino que realiza una solicitud al contrato respectivo de la entrada para cambiar el valor "buyer" al nombre del solicitante y que el vendedor acepte la transacción.
/aceptar	Cuando el usuario que vende la entrada acepta la transacción, se modifican los atributos del contrato y se da la titularidad al nuevo usuario. Se registra la entrada en la blockchain con el nuevo nombre correspondiente y se le asigna al comprador para que le aparezca en su sesión.

Tabla 4. Rutas del Back-End de la plataforma web.

4.5 Desarrollo de la plataforma web

4.5.1 Nombre y logo

El nombre escogido para la página web es “Seshat Secure Tickets”, inspirado en la cultura egipcia. Seshat es el nombre de la diosa de la escritura, la historia y los libros y protectora de las bibliotecas, asociada con preservación y el registro del conocimiento. Ya que la blockchain es un registro inmutable de los datos, es un nombre apropiado pues transmite la idea del cuidado y la fiabilidad de las transacciones registradas.

Para la creación del logo, se ha fusionado la imagen común de una entrada y un candado, simbolizando la seguridad y la protección de la integridad de cada transacción.

Ilustración 5. Logo de la Plataforma Web.

La elección del nombre y el diseño se alinea perfectamente con los valores de la plataforma utilizando elementos visuales que evocan tanto la cultura antigua y la seguridad aplicados a la modernidad y la tecnología.

4.5.2 Menú de navegación

Para navegar por la interfaz precisamos de un menú de navegación y una barra de búsqueda desplegada por un archivo llamado “base.html”, sirviéndonos de plantilla para el resto de las pestañas. El menú lateral contiene cinco botones: el logo de la plataforma, “Mis entradas”, “Comprar Entradas”, “Vender Entradas” e “Iniciar Sesión”/”Cerrar Sesión”. Estos botones, así como la barra de búsqueda redirigirán a diferentes páginas que se mostrarán en el espacio restante de la plantilla.

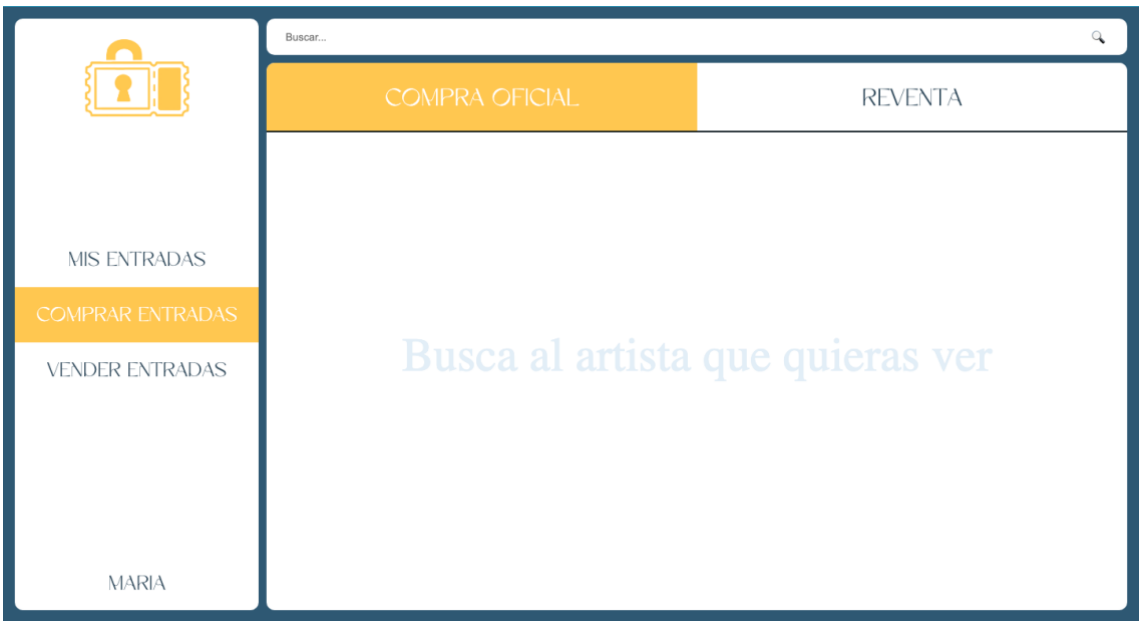

Ilustración 6. Captura en la que se muestra el menú lateral y la barra de búsqueda.

El último de los botones, además, muestra el valor la de variable "user" cuando la sesión está iniciada, la cual se pasa al ejecutar cualquiera de las funciones de redirección en views.py. Además, cuando se pase el ratón por encima dará la opción de cerrar sesión, y si es pulsado, se modificará el valor de "user" a "Inicia sesión" para volver al inicio del funcionamiento.

4.5.3 Página de inicio

La primera entrada que aparece en el menú de navegación es el propio logo de la página web. Al pulsarlo nos llevará a la página de inicio de la interfaz, ejecutada por el método "/" del Back-End. En esta función se establece el valor de la variable "actual" a "0", correspondiente con la página de inicio, y se ejecuta la función "estado", devolviendo los parámetros correspondientes. Si entramos a la función "estado" podemos observar distintas condiciones que determinarán la salida de los parámetros:

- Si "previo" y "actual" son = 0: significa que la página anterior era el índice y se ha vuelto a pulsar el botón que nos lleve a esta página. Por lo que no tiene transición y los nombres de la clase de los elementos del menú son aquellos que permiten que ocupe toda la página.
- Si "previo" y "actual" son = 1: es el caso contrario al anterior. El estado del menú en la página anterior era recogido a la izquierda y en la pestaña que se ha pulsado el menú ha de ocupar la misma posición.
- Si los valores de "previo" y "actual" son distintos entre sí: dependerá de si la pestaña que se ha pulsado es la del índice u otra para determinar la posición del menú. En este caso sí que se realiza una transición por lo que el valor de la clase de los elementos es el contrario al que sería el correcto para su resultado final y se realiza la animación mediante el script de JavaScript en el documento HTML de la página seleccionada.

La función estado se ejecuta en todos los métodos. Las pestañas que no sean la de inicio tendrán como valor "actual" "1", y en "previo" se almacenará el valor de "actual" antes de ser modificado.

Una vez se despliega la pantalla de inicio, los botones del menú ocuparán toda la pantalla, habiendo realizado o no la transición correspondiente con relación al valor de "previo". Con los botones del menú ocupando toda la pantalla, una vez se pulse alguno distinto al logo, el menú se recoge al lateral de la pantalla y la despeja para cargar el contenido gracias a la función explicada previamente.

Ilustración 7. Captura del menú desplegado tras pulsar el botón del logo.

4.5.4 Página Mis Entradas

Como podemos observar en el menú, la opción inmediatamente inferior al botón del logo de la interfaz es “Mis Entradas”, desplegada por el archivo “index_misentradas.html”. Una vez pulsado el botón, se envía una redirección a la ruta “/mis_entradas” del Back-End de la interfaz.

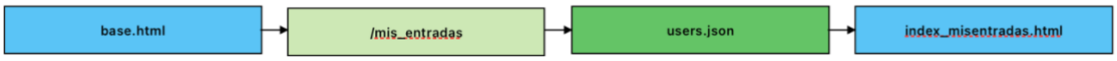

Ilustración 8. Diagrama del funcionamiento de la pestaña Mis Entradas.

```
<div class="{{ div_header }}" id="div_header_1">
    <a href="/mis_entradas"><p class="p_div">MIS ENTRADAS</p></a>
```

Ilustración 9. Código que muestra el funcionamiento del botón "Mis Entradas" del menú.

Una vez en el método se comprueba que la sesión esté iniciada. En caso contrario se enviará al usuario a la página de inicio de sesión.

```
if user == "Inicia sesión":
    return render_template('index_user.html',
                           user = user,
                           header = header,
                           resto = resto,
                           div_header = div_header,
                           transition = transition)
```

Ilustración 10. Código que comprueba si no se ha iniciado sesión.

Lo primero que se hace es crear una lista vacía y abrir el documento “users.json” que almacena todos los nombres de usuario registrados y sus correspondientes entradas, haciendo el funcionamiento de una base de datos. En la lista, buscará el usuario que coincida con el nombre de inicio de sesión y añadirá a la lista vacía las entradas que este tiene asociadas. Esta lista se devuelve como parámetro en el despliegue de la plantilla que muestra las entradas.

```
tickets = []

with open(DATA_FILE_PATH_USERS, 'r', encoding='utf-8') as file:
        user_tickets = json.load(file)

for user_iter in user_tickets['users']:
    if user_iter['name'] == user :
        for ticket in user_iter['tickets']:
            tickets.append(ticket['ticket'])
```

Ilustración 11. Código que almacena las entradas asociadas al usuario.

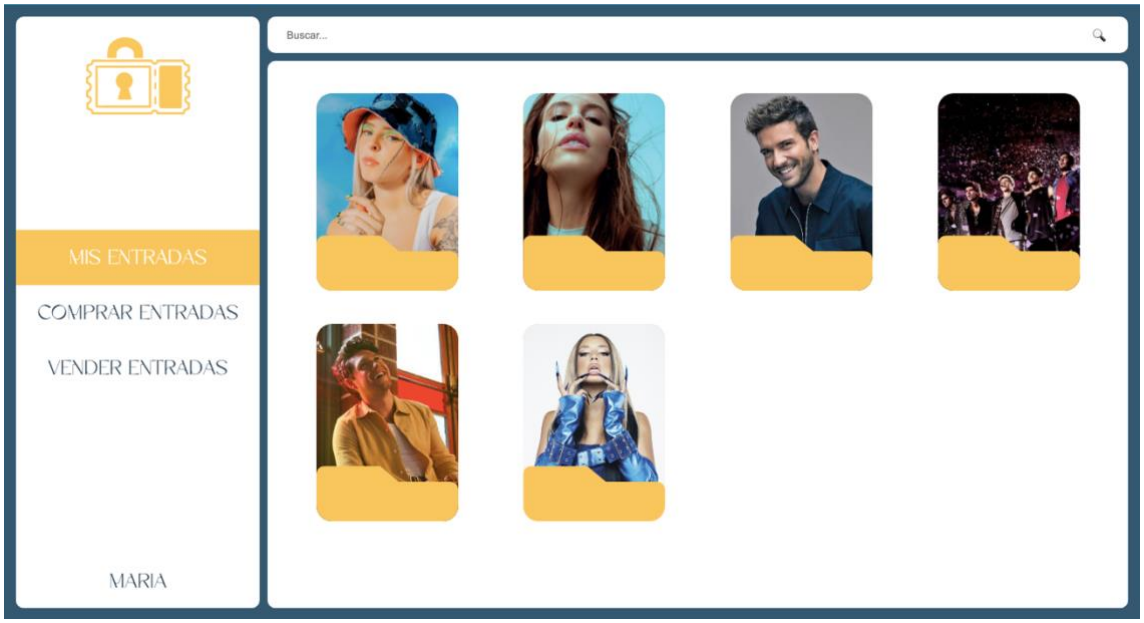

Ilustración 12. Captura de la pestaña "Mis Entradas" con las entradas del usuario “MARIA”.

En el documento “index_misentradas.html” que despliega la función, encontramos un bucle, que crea un elemento por cada entrada en la lista enviada como parámetro a la interfaz. Cada entrada comprada aparecerá recogida y únicamente se apreciará la foto del artista. Cada entrada tiene unas dimensiones establecidas en los estilos CSS, con los bordes redondeados y dos elementos que simulan la pestaña de una carpeta. Si situamos el ratón encima de la entrada, esta solapa subirá para que pulsemos y podamos ver toda la información de la entrada.

```
{% for ticket in tickets %}

<div class="ticket"><div class="ticket3">
    <div class="ticket_info">
        <p class="search-input">{{ticket.artist}}</p>
        <p class="entry-city">Ciudad: {{ticket.city}}</p>
        <p class="entry-country">País: {{ticket.country}}</p>
        <p class="entry-day">Fecha: {{ticket.day}}</p>
    </div>
</div><div class="ticket2">
    <img class="img" src="{{ ticket.photo }}">
    <div class="carpeta" id="solapa_alta"></div>
    <div class="carpeta" id="solapa_baja"></div>
</div></div>

{% endfor %}
```

Ilustración 13. Código del Front-End que crea un elemento por cada entrada.

Si pulsamos en una de ellas, esta se ampliará mostrando sus datos más importantes: nombre artista, fecha y lugar, mediante una función ejecutada en el script de JavaScript. Esta función tiene un método “toggle” que añade la clase “active” al elemento pulsado y cambia sus estilos para poder desplegar la entrada. Cuando una entrada esté desplegada, el resto no se podrá pulsar, habrá que volver a recoger la desplegada para poder ver la información de una de las otras.

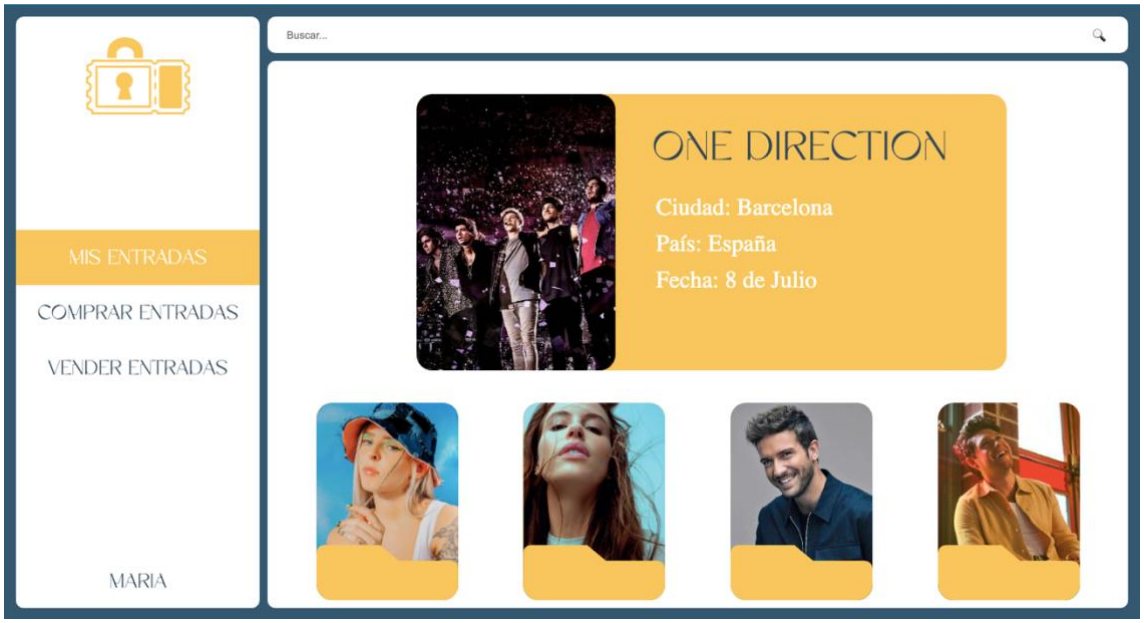

Ilustración 14. Captura de una entrada desplegada en la pestaña "Mis Entradas" y el resto sin desplegar.

```
div.addEventListener('click', function() {
    const activeElement = document.querySelector('.ticket.active');
    if (activeElement && activeElement !== this) {
        return;
    }
    this.classList.toggle('active');
```

Ilustración 15. Código JavaScript que activa un elemento y evita la interacción con el resto.

4.5.5 Barra de búsqueda y Página Comprar entradas

Para que se muestren las entradas a la venta podemos lograrlo de dos maneras diferentes, y en todas depende del valor de la barra de búsqueda. El botón “Comprar Entradas” devuelve al usuario a la página de búsqueda, pero con un valor de entrada vacío por lo que en la interfaz aparece un mensaje de “Introduce el artista que quieras ver”. Si se introduce un nombre no válido o un artista que no tiene resultados que ofrecer aparecerá un mensaje de “No se han encontrado resultados”.

La acción de introducir un artista en la barra de búsqueda se puede realizar desde cualquier pestaña. Tanto si introducimos el nombre del artista que elijamos en la barra de búsqueda desde la pestaña de “Comprar Entradas” como cuando nos encontremos en otra ventana, nos mostrará directamente los resultados del artista buscado.

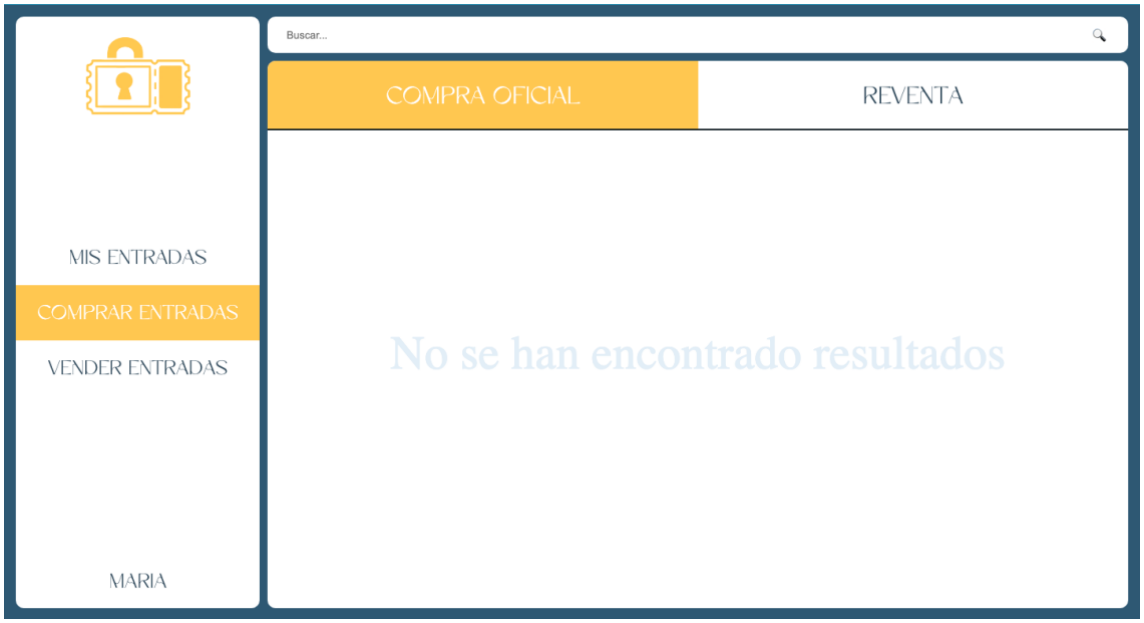

Ilustración 16. Captura de la pestaña "Comprar Entradas" mostrando el mensaje "No se han encontrado resultados".

Una vez introducimos un valor en la barra de búsqueda, implementada como un formulario para poder obtener el valor, lo almacenamos en una variable. Abrimos el documento "data.json" donde previamente se habrán registrado los conciertos de los que se pueden comprar entradas, y buscaremos el valor introducido en la barra de búsqueda entre las opciones de compra. Se almacenan todas las opciones posibles y se pasan a la plantilla HTML para que las muestre. El proceso es el mismo para las entradas de reventa, pero tanto el documento de búsqueda como la lista en la que se almacenan las entradas cambia, siendo "data.json" el archivo para entradas oficiales y "reventa.json" para entradas de reventa. La estructura del archivo JSON es la siguiente, por ello, una vez se encuentra el artista, obtenemos todos los datos necesarios de la entrada y es el resultado que almacenamos.

```
"One Direction": [
    {
        "city": "Barcelona",
        "country": "España",
        "limit": 100,
        "sold": 19,
        "day": "8 de Julio",
        "photo": "onedirection.jpeg"
    },
```

Ilustración 17. Estructura de la información en el archivo "data.json".

La función "/search" del back-end almacena tanto las entradas de venta original como las de reventa. Para ello busca el artista introducido en ambos documentos JSON con entradas. Para las entradas originales creamos una variable a la que se le da el valor "true" si se han encontrado entradas del artista y "false" so no se encuentra ninguna. Lo mismo se hace con las entradas de reventa en su documento respectivo. Estas variables se le pasan en forma de argumento a la plantilla para determinar el resultado a mostrar por pantalla, así como las listas de entradas encontradas, en diferentes variables para cada opción.

```
with open(DATA_FILE_PATH_DATA) as f:
    data = json.load(f)

if search_input in data:
    found = 'true'
    entries = data[search_input]
else:
    found = 'false'
    entries = 'No se han encontrado resultados'
```

Ilustración 18. Código de almacenamiento de las entradas oficiales.

```
with open(DATA_FILE_PATH_REVENTA) as f:
    data_r = json.load(f)

if search_input in data_r:
    found_r = 'true'
    entries_r = data_r[search_input]
else:
    found_r = 'false'
    entries_r = 'No se han encontrado resultados'
```

Ilustración 19. Código de almacenamiento de las entradas de reventa.

Una vez en "index_busqueda.html", se crea un elemento por cada resultado almacenado con los datos de la entrada gracias a un bucle "for". Además, se comprueba si el límite de entradas es el mismo valor que el número de entradas vendidas. Si ya no quedan más por vender, se cambia el botón por uno de "Sold Out" que ya no permite la compra. La ventana contiene animaciones de cambio de color cuando el ratón entra en cada uno de los botones exceptuando el de "Sold Out" que no permite realizar ninguna acción.

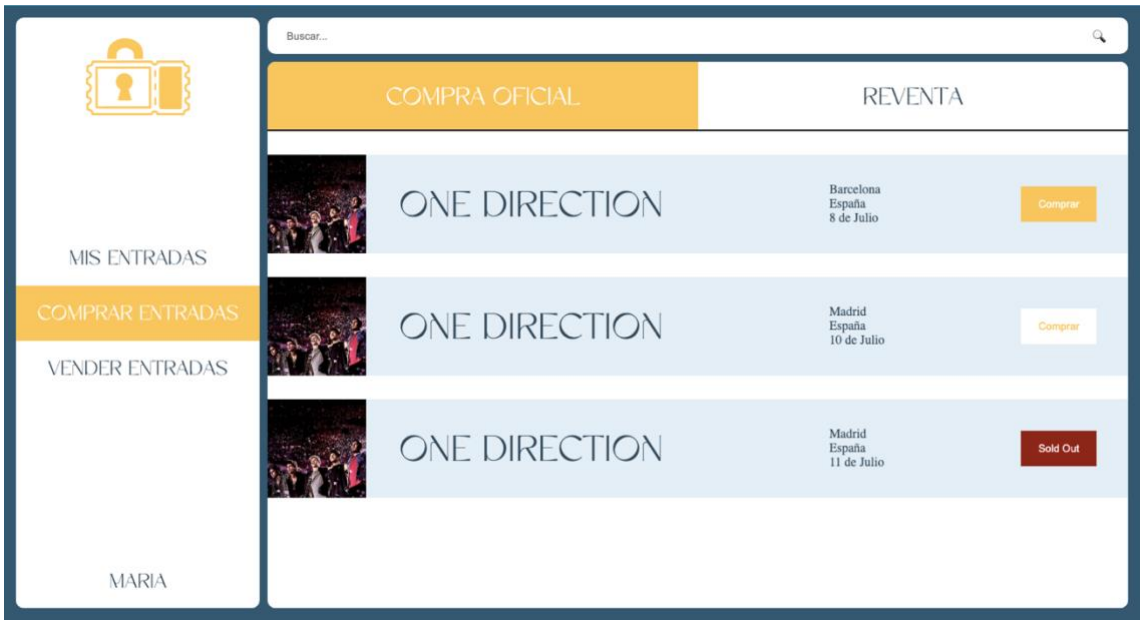

Ilustración 20. Captura de la pestaña "Comprar Entradas" una vez hemos introducido un artista válido.

Una vez desplegada la pestaña podemos observar que en la parte superior aparecen dos botones, en el primero está escrito "Compra Oficial" y en el segundo "Reventa". Según pulsemos uno y otro, se activará una función escrita en JavaScript que mostrará las entradas correspondientes al seleccionado y ocultará el alterno, según queramos ver las entradas oficiales o las de reventa.

```
function change_reventa(){
    div_oficial.style.display = 'none';
```

Ilustración 21. Código JavaScript que oculta una de las dos opciones, en este caso la venta oficial.

Si pulsamos en comprar o solicitar compra, dependiendo del tipo de entrada que queramos comprar, el script de JavaScript obtiene la información del ticket que hemos seleccionado para comprar y lo añade a la url para poder pasar la información al archivo "views.py" y en consecuencia a la siguiente página HTML, para introducir los datos y adquirir la entrada.

```
window.location.href = "/compra?search_input=" + encodeURIComponent(searchInput) +
"&oficial=" + encodeURIComponent(oficial) +
"&entry_city=" + encodeURIComponent(entryCity) +
"&entry_country=" + encodeURIComponent(entryCountry) +
"&entry_day=" + encodeURIComponent(entryDay) +
"&entry_photo=" + encodeURIComponent(entryPhoto) +
"&entryHidden=" + encodeURIComponent(entryHidden);
```

Ilustración 22. Código JavaScript que añade parámetros a la ruta de compra que se envía al Back-End.

4.5.6 Formulario de Comprar Entradas

Una vez pulsado el botón de compra, se llama al método "/compra" en "views.py". Si no se ha iniciado sesión, se redirigirá a la página de inicio al igual que en la página de "Mis Entradas". Si la sesión se ha iniciado, se obtienen los datos introducidos en la url y se añaden como variables al desplegar la página "index_compra.html". Se ha añadido un valor en la url que indica si la entrada pulsada para realizar la compra es de venta oficial o reventa. En el segundo caso, el formulario de solicitud de compra será diferente ya que llama a otro endpoint, por lo que, al pulsar dicho botón, en lugar de finalizar la transacción, realizar el supuesto pago y añadir la entrada a la blockchain, se enviará la solicitud al contrato inteligente para que modifique su estado y se pueda proceder con la validación del vendedor. Esta función la explicaremos más adelante.

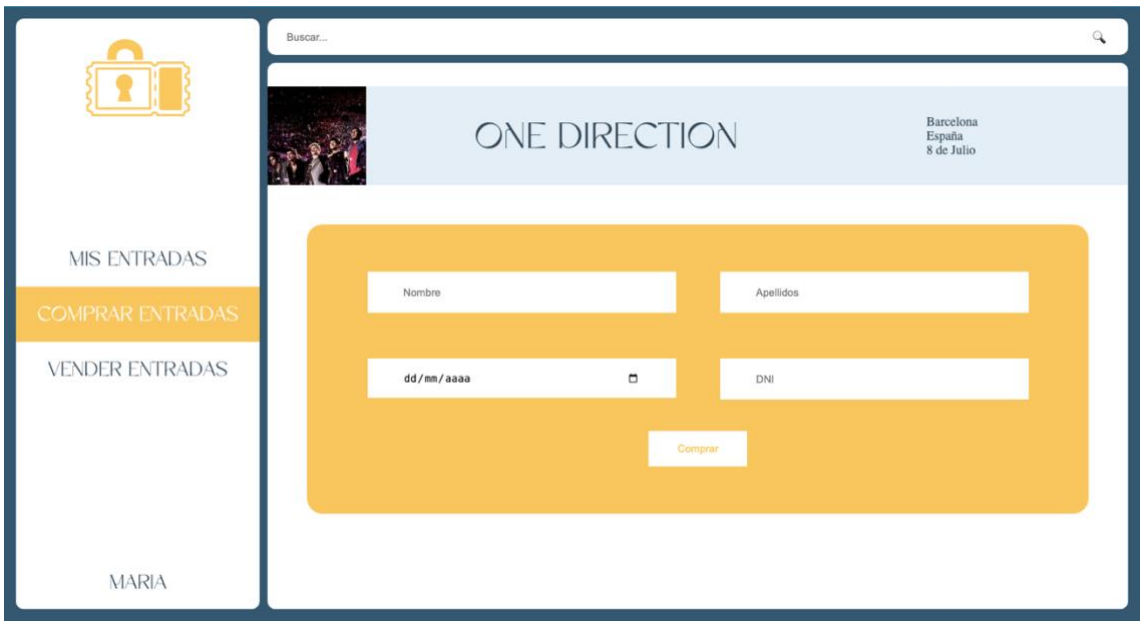

Ilustración 23. Captura de la pestaña del formulario de compra.

4.5.7 Página Vender Entradas

Para desplegar la interfaz de venta de entradas, tenemos el endpoint "/venta" que devuelve la plantilla y almacena en una lista las entradas que ha comprado el usuario y están asignadas en el documento "users.json" asociadas al usuario correspondiente. Estas entradas tienen una variable llamada "state". El valor de esta variable puede estar vacío, ser "puesta a la venta" o "compra solicitada". Dependiendo del valor, la entrada se mostrará de un modo u otro en nuestra interfaz.

Ilustración 24. Captura de la pestaña "Vender Entradas".

Vamos a ponernos en el caso inicial, es decir, todavía no hemos puesto una entrada a la venta, así podemos ir explicando las opciones conforme sea pertinente.

Para poner entradas a la venta disponemos de un botón en la parte superior de la interfaz. Si se pulsa, se despliegan las entradas que el usuario ha comprado y no están puestas a la venta, es decir, el valor de "state" está vacío. Estas entradas se muestran gracias a una función en JavaScript, que modifica el estilo "height" del elemento donde se encuentran las entradas.

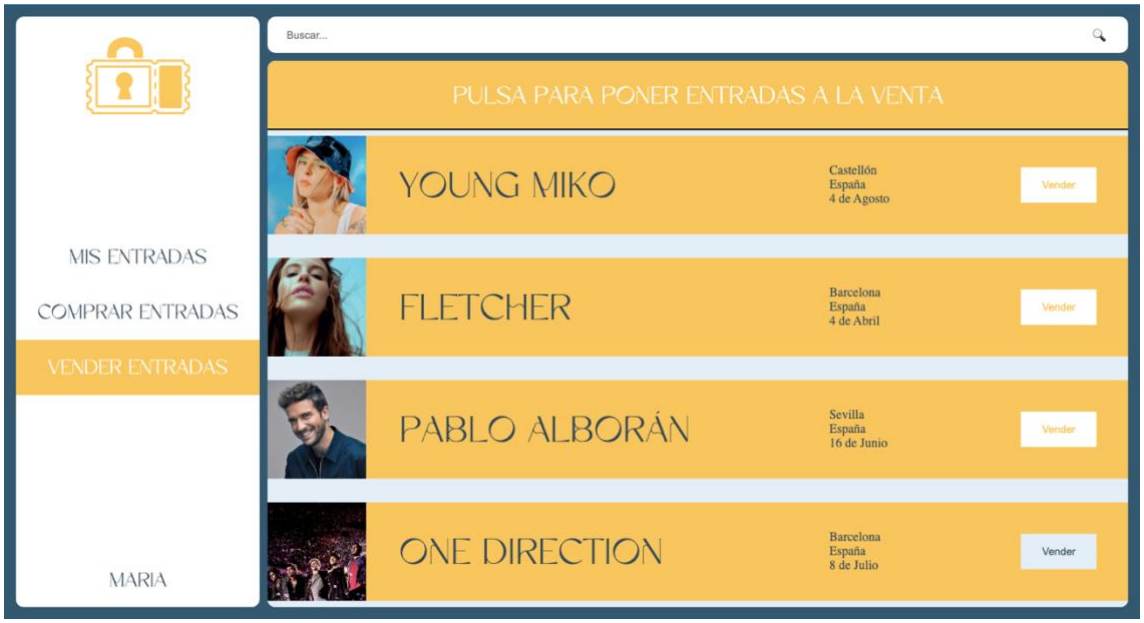

Ilustración 25. Captura de la pestaña "Vender Entradas" cuando se pulsa el botón de poner en venta entradas.

Si se pulsa el botón “vender” en alguna de ellas, se pondrá a la venta la entrada respectiva llamando a la función correspondiente del Back-End de la interfaz, en este caso al endpoint “/puesta_venta” y se pasan como argumentos los datos de la entrada. En la interfaz, la entrada ahora aparecerá en la ventana principal con un botón indicando que la entrada ha sido puesta a la venta.

```
window.location.href = "/puesta_venta?ticket_artist=" + encodeURIComponent(ticketArtist) +
"&ticket_city=" + encodeURIComponent(ticketCity) +
"&ticket_country=" + encodeURIComponent(ticketCountry) +
"&ticket_day=" + encodeURIComponent(ticketDay) +
"&ticket_photo=" + encodeURIComponent(ticketPhoto) +
"&ticket_contract=" + encodeURIComponent(ticketHidden);
```

Ilustración 26. Código JavaScript que añade parámetros a la ruta de puesta en venta que se envía al Back-End.

Ilustración 27. Captura de la pestaña "Vender Entradas" con una entrada puesta en venta.

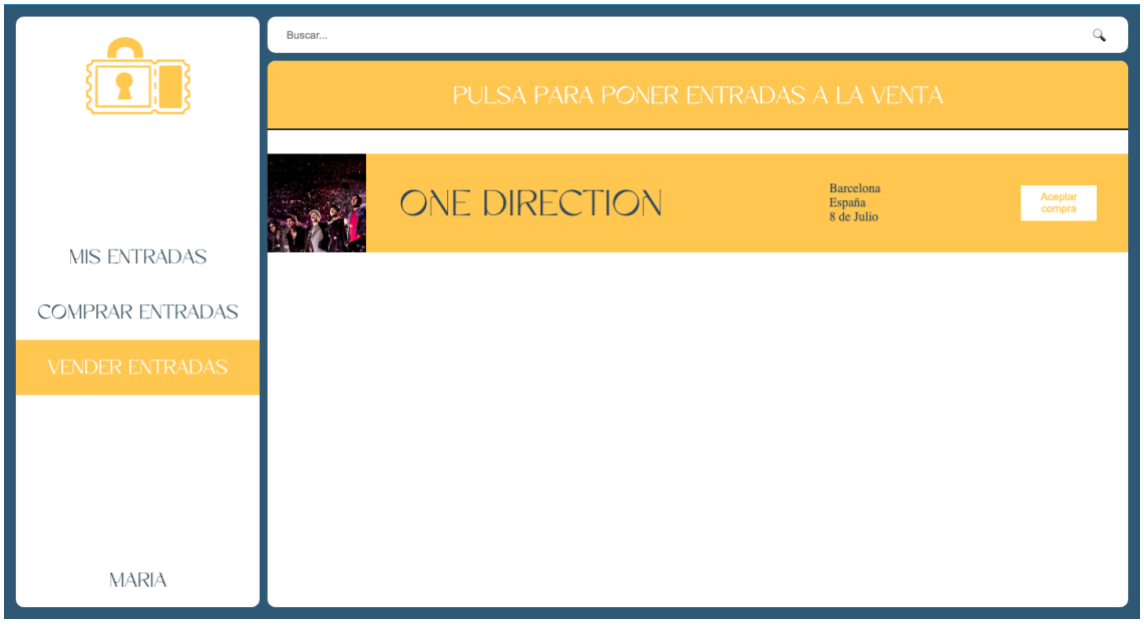

Ilustración 28. Captura de la pestaña "Vender Entradas" con la compra de entrada solicitada.

Si se solicita la compra de una de las entradas, el valor “state” de la entrada en cuestión habrá sido modificado y ahora su valor será “Compra solicitada”. En la interfaz de “Vender Entradas”, el botón correspondiente a la entrada solicitada habrá cambiado de “Puesta en venta” a “Aceptar compra”. Si pulsamos el botón nos lleva al método “/aceptar” del back-end de la interfaz.

4.5.8 Página Iniciar Sesión

Ilustración 29. Diagrama del funcionamiento del inicio de sesión.

Tal y como ocurre en la mayoría de las aplicaciones actualmente, para poder personalizar la experiencia del usuario y almacenar el contenido propio, es necesario realizar un registro y navegar con la sesión iniciada. En la plataforma web implementada esto no es distinto. Para poder funcionar con total libertad, es necesario que el usuario que esté interactuando con la aplicación haya iniciado sesión para poder registrar sus compras y mostrar las entradas que únicamente él ha adquirido.

En la parte inferior del menú encontramos un botón en el que aparece escrito “Inicia sesión”, si es pulsado, nos llevará a la página de registro o inicio de sesión. Esto también ocurrirá en el caso en el que se vaya a dar un proceso que requiera acceder a la información del usuario, más concretamente la muestra de entradas, la compra o la venta.

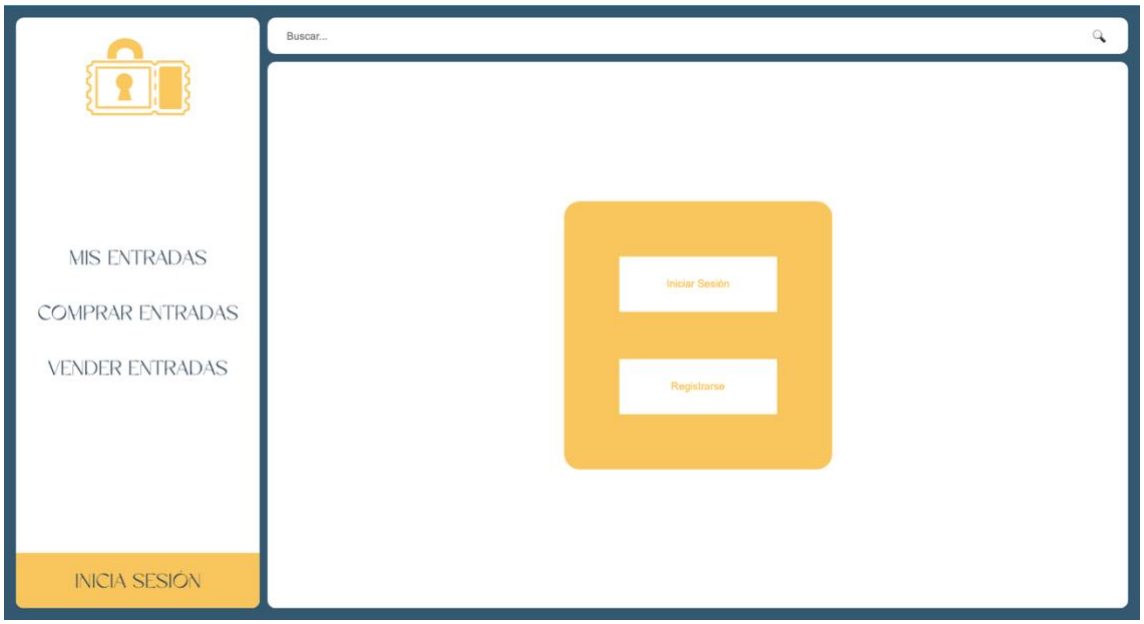

Ilustración 30. Captura de la pestaña "Inicia Sesión".

Una vez nos encontramos en la pestaña desplegada por “index_user.html”, vemos que tenemos dos opciones: “Iniciar sesión” o “Registrarse”. En el caso de que pulsemos “Iniciar Sesión”, el botón “Registrarse” desaparecerá, el pulsado se desplazará hacia abajo y se mostrarán dos campos de texto para introducir el nombre de usuario con el que se ha obtenido el acceso a la aplicación y la contraseña para poder llevarlo a cabo. Estos campos forman parte de un formulario, cuando se pulsa al botón para finalizar la acción, las respuestas de este formulario se envían a la ruta “/user” que una vez dentro de la función del método leerá el valor del botón pulsado, en este caso “log_in”, y accederá a la condición que cumpla el valor que es el mencionado.

Ilustración 31. Captura de la pestaña "Inicia Sesión" si pulsamos "Iniciar Sesión"

Como en este caso se trata del inicio de sesión de un usuario previamente registrado, la función del Back-End de la plataforma busca en el documento “users.json” el valor introducido mediante la función search_word(). Si el valor no es encontrado, la ventana de inicio de sesión volverá a su estado inicial. Si el valor es encontrado en el archivo, se le atribuye a la variable global “user” para que sea utilizado en el resto de las redirecciones manteniéndose la sesión del usuario abierta. Finalmente, la función devuelve al usuario a la pestaña previa a realizar el inicio de sesión gracias a la variable “template”.

```
if value == "log_in":
    user_form = request.form["user"]

    with open(DATA_FILE_PATH_USERS, 'r', encoding='utf-8') as file:
        users = json.load(file)

    search_results = search_word(users, user_form)

    if search_results:
        user = user_form
    else:
        user = "Inicia sesión"
```

Ilustración 32. Código que busca a un usuario ya registrado en el archivo "users.json".

Si por el contrario se pulsa al botón “Registrarse” en lugar del de “Iniciar Sesión” el proceso es muy similar. El formulario que aparece en pantalla sigue la misma estructura, pero se añade un campo de texto adicional, en el que se inserta el nombre y los apellidos del usuario. En este caso, el botón para enviar los resultados al Back-End de la plataforma tiene como valor “Register”, por lo que la condición en la que entra en la función del método también cambia. En este caso busca si el nombre de usuario introducido se encuentra en el documento JSON, pero para comprobar que no se realiza el registro de dos nombres de usuario iguales. Una vez esto se cumple, se añade

el usuario al archivo JSON, se modifica la variable global “user” y se redirige a la página en la que se encontraba el usuario previamente a realizar el registro.

Ilustración 33. Captura de la pestaña "Inicia Sesión" si se pulsa la opción "Registrarse".

En caso de que la sesión ya esté iniciada, el botón de inicio de sesión mostrará en su lugar el nombre de usuario que tenga la sesión iniciada. Si desplazamos el ratón por encima, el texto cambia a “Cerrar sesión”, que llevará a cabo esta acción si es pulsado ya que se trata de otro formulario que redirige al método “/user” pero en este caso con valor “log_out”.

```
{% if (user == "Inicia sesión") %}
    <div id="div_header_user" class="{{ div_header }}">
        <a href="/sesion"><p id="text_user" class="p_div">{{ user }}</p></a></div>
{% else %}
    <div id="div_header_user" class="{{ div_header }}"><p id="text_dhu">{{ user }}</p>
        <form action="/user" id="textform3" method="post">
            <button type="submit" name="Action" class="p_div" value="log_out" id="text_user">{{ user }}</button></form></div>
{% endif %}
```

Ilustración 34. Código del Front-End que muestra el valor de "user" en el botón del menú y cierra sesión.

La condición “log_out” vuelve a reestablecer el valor de la variable global “user” a “Inicia Sesión” lo que provocará que no se pueda acceder al resto de funcionalidades si no se vuelve a introducir un usuario válido, repitiendo así este proceso.

Capítulo 5. Diseño del servidor Blockchain

Para lograr el correcto funcionamiento de la plataforma, se precisa de un servidor que permita la interacción y el almacenamiento de datos de todos los usuarios. Además, en este caso, para el registro de los datos se emplea una cadena de bloques, también almacenada en este servidor.

Anteriormente hemos explicado el funcionamiento de una Blockchain descentralizada ya existente basada en Ethereum, permitiendo implementar aplicaciones de código y con contratos implementados en el lenguaje Solidity.

Sin embargo, en lugar de utilizar de base una blockchain existente y aplicarle una plataforma web para interactuar con esta, en este trabajo se pretende explicar el funcionamiento de esta nueva tecnología de ciberseguridad de manera visual y poder entender sus métodos paso a paso. Es por ello por lo que se ha implementado la simulación del nodo de una blockchain creada desde cero en lenguaje Python, simulando mediante clases y funciones todos los procesos que se llevan a cabo para registrar una nueva transacción. En Python se trata de imitar el comportamiento de la cadena de bloques desplegando el contrato en forma de objeto en la lógica del servidor. Las transacciones en Python también se almacenan y se minan, registrándolas en un bloque de nuestra blockchain local, siendo inmutables, ya que no se pueden modificar sin invalidar la cadena, lo que imita la principal característica de una blockchain real.

Este servidor está formado únicamente por el archivo “node_server.py” escrito en lenguaje Python. En este archivo se encuentran las clases que crean los bloques de la blockchain y la propia cadena, así como los contratos inteligentes. Además, contiene las rutas que interactúan con la plataforma web para almacenar las transacciones que se requieran por las acciones del usuario en la interfaz.

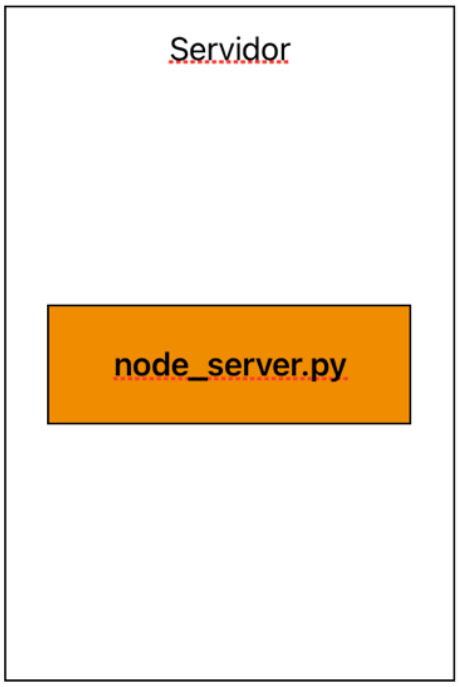

Ilustración 35. Archivos del servidor blockchain.

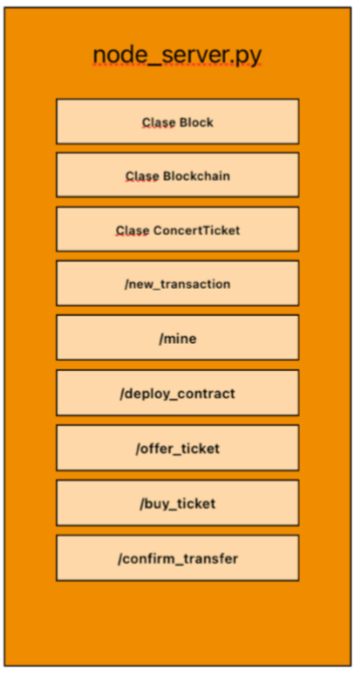

Ilustración 36. Clases y rutas del archivo node_server.py.

5.1 Clases del servidor Blockchain

Para añadir transacciones en forma de bloques a la cadena hemos de crear la propia cadena de bloques a la que estos se irán sumando y un nuevo objeto que será el añadido y contendrá la información.

En primer lugar, explicamos la clase "Block", encargada de crear cada objeto que se añadirá a la cadena. Cada bloque contiene una serie de atributos que definen su contenido y su relación con otros bloques en la cadena y estos atributos se definen en la propia clase, y son los siguientes:

Nombre del atributo	Explicación
index	Representa la posición del bloque en la cadena.
transactions	Alberga la información que se quiere almacenar en el bloque.
timestamp	Registra el instante de tiempo en el que se crea el bloque.
previous_hash	Almacena el hash del anterior bloque de la cadena para que los bloques se enlacen secuencialmente.
nonce	Se trata de un número utilizado en el proceso de minería para encontrar un hash válido. Este número se ajusta hasta que se encuentra un hash que cumple con ciertos requisitos establecidos previamente como, por ejemplo, un número específico de ceros al inicio.

Tabla 5. Atributos de la clase Block.

La clase Block contiene además una función llamada "compute_hash" que se utiliza para calcular el hash de cada bloque.

Para crear la cadena de bloques empleamos la clase "Blockchain", y lo primero que se realiza al crear el objeto es definir la dificultad de la prueba de trabajo, que determina cuántos ceros iniciales debe tener el hash de un bloque para que sea considerado válido.

La clase "Blockchain" cuenta con los siguientes atributos:

Nombre del atributo	Explicación
unconfirmed_transactions	Lista de transacciones no confirmadas
chain	Cadena de bloques con las transacciones registradas existentes
contracts	Diccionario que almacena las direcciones de los contratos creados y su objeto correspondiente.

Tabla 6. Atributos de la clase Blockchain.

La clase "Blockchain" contiene a su vez una serie de funciones para poder modificar los valores de sus atributos desde los métodos del servidor en base a lo que la plataforma web requiera. Las funciones de la clase "Blockchain" son las siguientes:

Nombre de la función	Explicación
create_genesis_block(self)	En el caso de que la cadena todavía no tenga ninguna transacción registrada se crea un bloque de inicio de la blockchain con todos sus parámetros inicializados a cero. Para crear el bloque génesis, se crea un elemento de la clase Block, cuyo índice será 0, una lista de transacciones vacía, tiempo 0, y sin hash previo, por lo que se le asigna el valor "0", ya que no tiene ningún bloque anterior.
last_block(self)	Obtiene el último bloque de la cadena.
add_block(self, block, proof)	Añade un nuevo bloque a la cadena. Se comprueba que el hash previo del nuevo bloque coincide con el hash del bloque anterior y que la prueba de trabajo es válida. Si es válida, el bloque se añade a la cadena.
proof_of_work(block)	Este método encuentra un valor de "nonce" del bloque para que el hash del bloque comience con un número de ceros igual a la dificultad establecida.
add_new_transaction(self, transaction)	Añade la transacción que se está gestionando a la lista de transacciones no confirmadas
is_valid_proof(cls, block, block_hash)	Verifica que el hash de un bloque es válido y cumple con la dificultad establecida.
check_chain_validity(cls_chain)	Verifica la relación entre el hash del bloque y el hash previo, así como la prueba de trabajo, pero en este caso a lo largo de todos los bloques de la cadena para comprobar la validez de esta.
mine(self)	Esta función añade las transacciones no confirmadas a un nuevo bloque, realiza la prueba de trabajo y llama a la función "add_block", para que añada el bloque a la cadena realizando las comprobaciones pertinentes. Si el bloque se añade de manera exitosa, la lista de transacciones no confirmadas se vacía.
add_contract(self, contract_address, contract)	Almacena el contrato en el diccionario "contracts" de la blockchain.
get_contract(self, contract_address)	Obtiene el contrato en el diccionario de la blockchain utilizando su dirección.

Tabla 7. Funciones de la clase Blockchain.

Por último, contamos con la clase ConcertTicket que equivale al contrato desplegado en la blockchain para regular la reventa.

Los atributos de la clase ConcertTIcket son los siguientes:

Nombre del atributo	Explicación
ticket_holder	Valor del propietario actual de la entrada.
ticket	Diccionario con la entrada puesta en venta.
buyer_data	Datos del comprador de la entrada
is_for_sale	Valor booleano que indica si la entrada ha sido puesta en venta o no.
pending_buyer	Indica el comprador que ha solicitado comprar la entrada.

Tabla 8. Atributos de la clase ConcertTicket.

La clase ConcertTicket posee a su vez 3 funciones además de la que define los atributos de la clase.

Nombre de la función	Explicación
offer_ticket_for_sale(self)	Cambia el valor del atributo “is_for_sale” a “True” ya que la entrada ha sido puesta en venta cuando esta se ejecuta.
buy_ticket(self, buyer, data)	Se asegura de que el comprador no es el mismo que el propietario de la entrada. Solo entonces cambia el valor al comprador en espera y el atributo que almacena sus datos, para que en caso de ser válida la transacción se puedan registrar en la blockchain.
confirm_transfer(self)	Se asegura de que hay un comprador introducido en los parámetros de la función. Almacena al anterior propietario de la entrada en la variable “old_holder”. Se establece el comprador como nuevo propietario y se reinician los valores de entrada puesta a la venta y comprador, para el caso de que la entrada vuelva a ponerse en reventa.

Tabla 9. Funciones de la clase ConcertTicket.

5.2 Métodos del servidor Blockchain

Al igual que en el servidor Back-End de la interfaz, el servidor blockchain cuenta con métodos que interactúan con las clases de la cadena de bloques modificando sus parámetros, registran transacciones y devuelven datos a la plataforma web para su correcto funcionamiento. Las rutas implementadas son las siguientes:

Nombre de la ruta	Explicación
/new_transaction	Añade la nueva transacción a la lista de transacciones no confirmadas de la clase "Blockchain"
/mine	Llama a la función de minado de la clase "Blockchain" para realizar la prueba de trabajo y validar el bloque.
/deploy_contract	Crea una nueva dirección de contrato para una entrada.
/offer_ticket/<contract_address>	Llama a la función "offer_ticket_for_sale()" de la clase ConcertTicket que cambia el valor de "is_for_sale" a True para añadir las entradas a las que se encuentran en venta.
/buy_ticket/<contract_address>	Llama a la función "buy_ticket" de la clase ConcertTicket para añadir el nombre del comprador a los atributos del contrato.
/confirm_transfer/<contract_address>	El vendedor acepta la compra y se llama a la función "confirm_transfer()" de la clase ConcertTicket para actualizar el dueño de la entrada y reiniciar los valores del contrato.

Tabla 10. Rutas del servidor blockchain.

Capítulo 6. Interacción de la blockchain con la plataforma web

6.1 Proceso de compra de entradas oficiales

En primer lugar, vamos a explicar el proceso de una compra original. El proceso de compra de reventa se explicará más adelante, una vez entendamos el funcionamiento de su puesta en venta y la participación del contrato en la gestión.

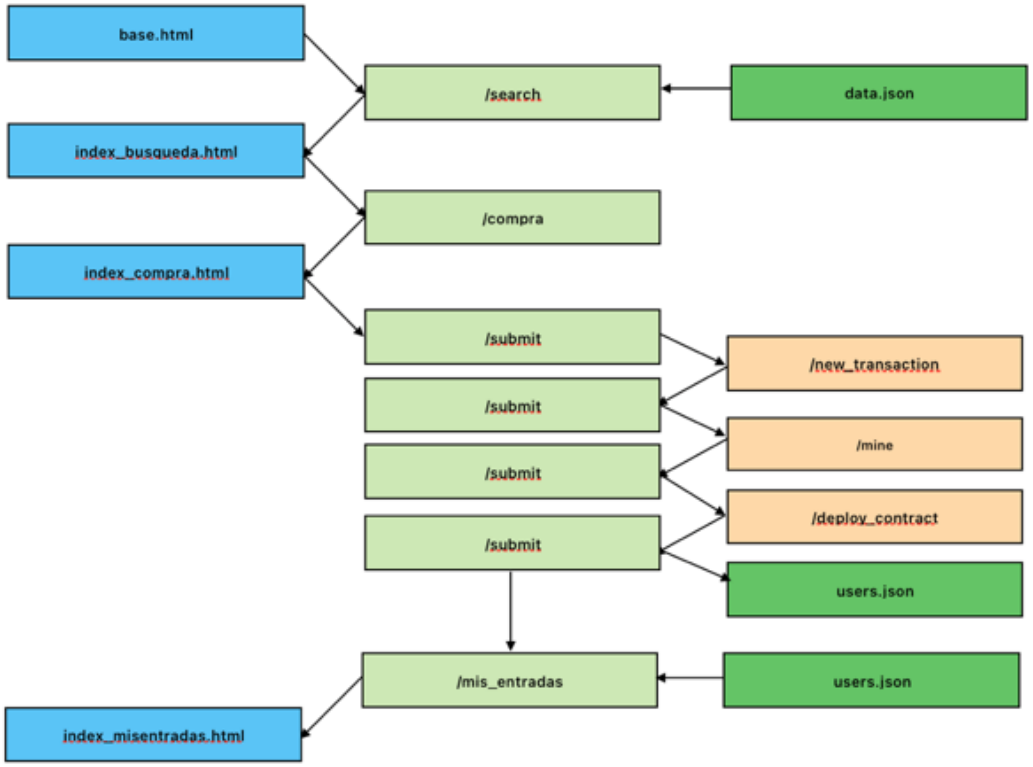

Ilustración 37. Diagrama del funcionamiento de la compra de entradas oficiales.

Para comprar una entrada oficial hemos de acceder a la página que muestra las entradas a la venta, explicada en el apartado 4.5.5 Barra de búsqueda y Página Comprar Entradas, y para este caso tener seleccionado el botón superior de entradas oficiales. Una vez se han desplegado las entradas, seleccionamos la que queramos comprar y pulsamos el botón “Comprar”.

Tal y como hemos visto en la explicación de la interfaz, en la plantilla 4.5.6 Formulario de Comprar Entradas, observamos un formulario con campos de texto a rellenar. El botón del formulario redirecciona a la función “/submit”. Una vez en la función se recogen los datos de los campos del formulario, tanto los introducidos por el usuario como los ocultos que incluyen la información de la entrada. Se almacenan en variables y se crea el objeto que se va a registrar en la blockchain mediante la transacción.

```
post_object = {
    'name': name,
    'surname': surname,
    'owner': user,
    'born': born,
    'dni': dni,
    'ticket': { "artist" : artist,'city': city,'country': country,'day': day,'photo': path_photo}
}

new_tx_address = "{}/new_transaction".format(CONNECTED_NODE_ADDRESS)
response_new_tx = requests.post(new_tx_address,
                  json=post_object,
                  headers={'Content-type': 'application/json'})
```

Ilustración 38. Código de creación del objeto y llamada al servidor blockchain.

Una vez creado el objeto, se escribe la ruta de la función de la blockchain que crea la nueva transacción empleando la dirección del servidor y el endpoint que contiene la función necesaria en el servidor blockchain. Una vez creada la dirección se realiza una solicitud post en la que se incluye el objeto a introducir en la transacción mediante el método "/new_transaction" de la cadena de bloques.

El método "/new_transaction" recibe solicitudes post de información que se pretende añadir a la blockchain. Una vez en la función, se obtiene el objeto JSON, enviado a través de la solicitud, y se comprueba que los campos requeridos han sido introducidos para finalmente añadir los datos a la clase "blockchain" mediante la función "add_new_transaction()" del objeto.

```
@app.route('/new_transaction', methods=['POST'])
def new_transaction():
    tx_data = request.get_json()
    required_fields = ["name", "surname"] #### añadir fields

    for field in required_fields:
        if not tx_data.get(field):
            return "Invalid transaction data", 404

    tx_data["timestamp"] = time.time()

    blockchain.add_new_transaction(tx_data)
```

Ilustración 39. Función "new_transaction" que crea una nueva transacción en la blockchain.

Si la respuesta que el Back-End de la plataforma recibe del servidor es positiva entonces se realiza el mismo proceso de contacto con el servidor, pero esta vez con el endpoint de minado para añadir el nuevo bloque a la cadena.

```
if response_new_tx.status_code == 201:
    mined_block_address = "{}/mine".format(CONNECTED_NODE_ADDRESS)
    response_mine = requests.get(mined_block_address)
```

Ilustración 40. Solicitud de contacto con el servidor blockchain para realizar el minado.

La ruta "/mine" implementa las funciones para el proceso de minado de las transacciones y el consenso, cuya finalidad es verificar la cadena más larga entre los nodos de la red. La primera función es "mine_unconfirmed_transactions", que mina las transacciones de la blockchain por confirmar. Para ello se llama a la función "mine" de la clase "blockchain", que se encargará de realizar la prueba de trabajo en otra función llamada "proof_of_work". Este método encuentra un valor de "nonce" para que el hash del bloque comience con un número de ceros igual a la dificultad establecida.

```
def mine(self):

    if not self.unconfirmed_transactions:
        return False
    last_block = self.last_block
    new_block = Block(index=last_block.index + 1,
                      transactions=self.unconfirmed_transactions,
                      timestamp=time.time(),
                      previous_hash=last_block.hash)

    proof = self.proof_of_work(new_block)
    self.add_block(new_block, proof)
```

Ilustración 41. Función de minado de la blockchain.

```
def proof_of_work(block):

    block.nonce = 0
    computed_hash = block.compute_hash()
    while not computed_hash.startswith('0' * Blockchain.difficulty):
        block.nonce += 1
        computed_hash = block.compute_hash()

    return computed_hash
```

Ilustración 42. Código de la función de la prueba de trabajo de la blockchain.

Cada hash se genera mediante la función "compute_hash()". En su interior se obtiene un diccionario con atributos del objeto para generar un hash único para la transacción a minar. Una vez tenemos el objeto, creamos el hash mediante la función sha256.

```
def compute_hash(self):
    block_string = json.dumps(self.__dict__, sort_keys=True)
    return sha256(block_string.encode()).hexdigest()
```

Ilustración 43. Código de la función compute_hash().

A continuación, se ejecuta la función "consensus()", que implementa un algoritmo de consenso simple en la que se asegura que los nodos tienen la cadena más larga y válida. Se obtiene una lista de los nodos en la red y se hace una solicitud a cada uno para obtener su cadena de bloques. Si una cadena recibida es más larga y válida que la anterior se guarda como la cadena más larga. Esa cadena se compara con la cadena local y si es más larga se reemplaza.

```python
def consensus():

    global blockchain

    longest_chain = None
    current_len = len(blockchain.chain)

    for node in peers:
        response = requests.get('{}chain'.format(node))
        length = response.json()['length']
        chain = response.json()['chain']
        if length > current_len and blockchain.check_chain_validity(chain):
            current_len = length
            longest_chain = chain

    if longest_chain:
        blockchain = longest_chain
        return True

    return False
```

Ilustración 44. Función del protocolo de consenso de la blockchain.

Si el Back-End de la función recibe una respuesta positiva de que la cadena de bloques ha minado adecuadamente la nueva transacción, se vuelve a contactar con el servidor para crear un contrato y disponer de él en caso de que el usuario quiera vender la entrada, y cuya dirección servirá de identificador para gestionar la entrada. Por ello, volvemos a escribir la dirección de contacto con la blockchain, en este caso con la ruta "/deploy_contract" con el método post en la que obtenemos la dirección del contrato.

```python
contract_address = "{}/deploy_contract".format(CONNECTED_NODE_ADDRESS)
response = requests.post(contract_address,
                         json=contract_object,
                         headers={'Content-type': 'application/json'})
```

Ilustración 45. Solicitud de contacto con el servidor blockchain para crear una dirección de contrato.

En su función se obtiene un objeto JSON con los datos del vendedor y la entrada correspondiente. Se genera una dirección de contrato única utilizando el nombre de usuario del propietario y el instante de creación, y con esta información se calcula el hash empleando la función sha256. Se añade la información obtenida a un nuevo objeto de la clase ConcertTicket y se almacena el contrato creado en la blockchain con la función "add_contract". Se devuelve al Back-End de la plataforma la dirección del contrato para poder seguir operando con él.

```python
unique_string = f"{ticket_holder}-{time.time()}"
contract_address = sha256(unique_string.encode()).hexdigest()

contract = ConcertTicket(ticket_holder, ticket)
blockchain.add_contract(contract_address, contract)
```

Ilustración 46. Código de la creación de la dirección de contrato.

Una vez el Back-End de la plataforma recoge el identificador del contrato se actualiza el número de entradas vendidas en el JSON con los datos de las entradas a la venta y se introduce la entrada comprada en el JSON que contiene los usuarios y sus respectivas entradas. La función termina con la redirección del usuario a la ruta "/mis_entradas" que llevará al usuario a la pestaña que permite visualizar las entradas que éste posee con la última adquisición realizada.

6.2 Puesta en venta y compra de entradas de reventa

6.2.1 Puesta en venta de entradas de reventa

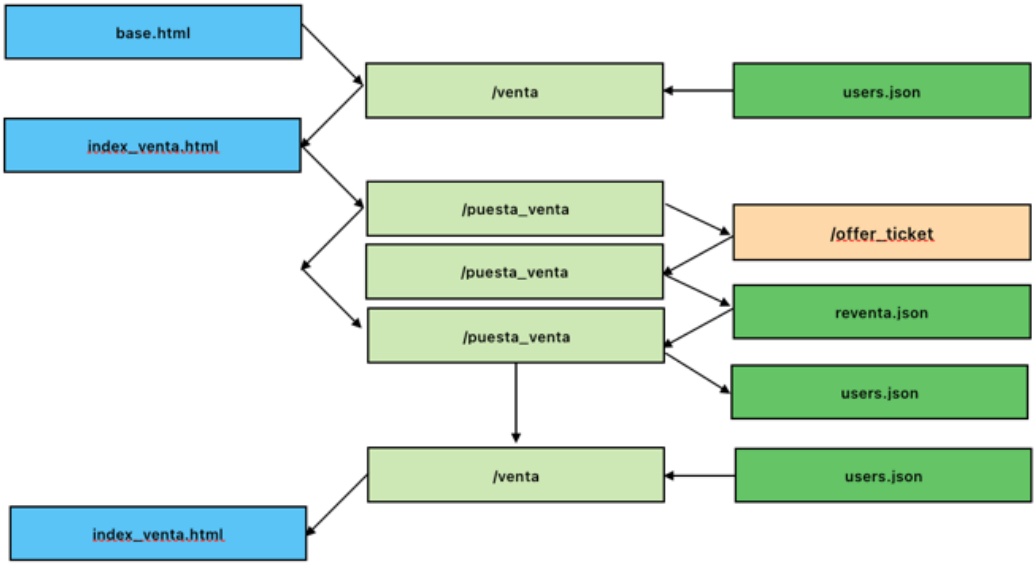

Ilustración 47. Diagrama del proceso de puesta de entradas en reventa.

Para poner entradas a la venta hemos de pulsar el botón "Vender Entradas" y seleccionar la que deseemos añadir. Cuando pulsemos el botón, llamaremos al método "/puesta_venta"

Al igual que en el proceso de compra de una entrada oficial, se crea el objeto que se va a enviar al servidor, se escribe la ruta de la función del servidor que se ha de ejecutar y se le añade la dirección de conexión del nodo. En este caso, el primer método al que hemos de llamar es "/offer_ticket".

```
sell_object = {
    'owner': user,
    'ticket' : { artist: {
    'city': city,
    'country': country,
    'day': day,
    'photo': photo
    }}
}

offer_address = "{}/offer_ticket/{}".format(CONNECTED_NODE_ADDRESS, contract)
response_2 = requests.post(offer_address, headers={'Content-type': 'application/json'})
```

Ilustración 48. Solicitud de contacto con la blockchain para poner una entrada a la venta

La función del método "/offer_ticket" se llama en el momento en el que se pone la entrada a la venta. Se comprueba que el contrato del cual se ha introducido la dirección existe y se llama a la función que actualizará el estado del atributo del contrato "is_for_sale" a verdadero.

```python
@app.route('/offer_ticket/<contract_address>', methods=['POST'])
def offer_ticket(contract_address):
    contract = blockchain.get_contract(contract_address)
    if not contract:
        return "Contract not found", 404

    try:
        contract.offer_ticket_for_sale()
    except ValueError as e:
        return str(e), 400

    return "Ticket offered for sale", 200
```

Ilustración 49. Código de la función "offer_ticket" del contrato en la blockchain.

Si la respuesta es positiva añadimos la entrada al documento "reventa.json" que almacena las entradas de reventa para que otros usuarios las puedan comprar y se modifica el estado de la entrada del usuario a "puesta en venta". La función nos devuelve a la pestaña de vender entradas y entonces podremos observar que nuestra entrada ya ha cambiado de estado y se encuentra puesta a la venta.

6.2.2 Compra de entradas de reventa

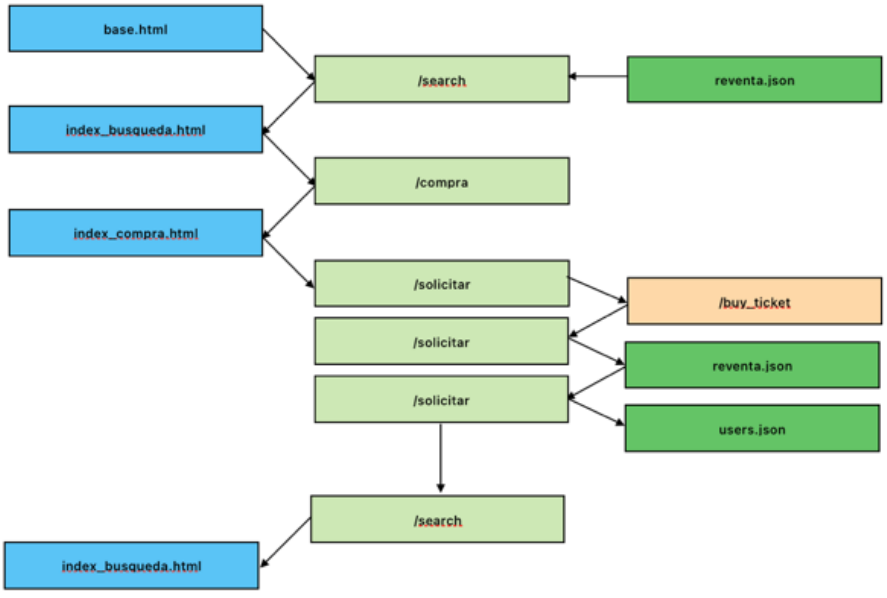

Ilustración 50. Diagrama del funcionamiento de la solicitud de compra de entradas de reventa.

Para el proceso de compra de entradas de reventa, un usuario diferente al que ha puesto la entrada en venta inicia el proceso de compra de la misma manera que en la compra de entradas oficial, a diferencia de que, en la parte superior ha de estar seleccionado el botón de "Reventa" en lugar del de "Compra Oficial", y, una vez pulsemos en la entrada que queramos adquirir seremos redirigidos a la pestaña de compra previamente explicada.

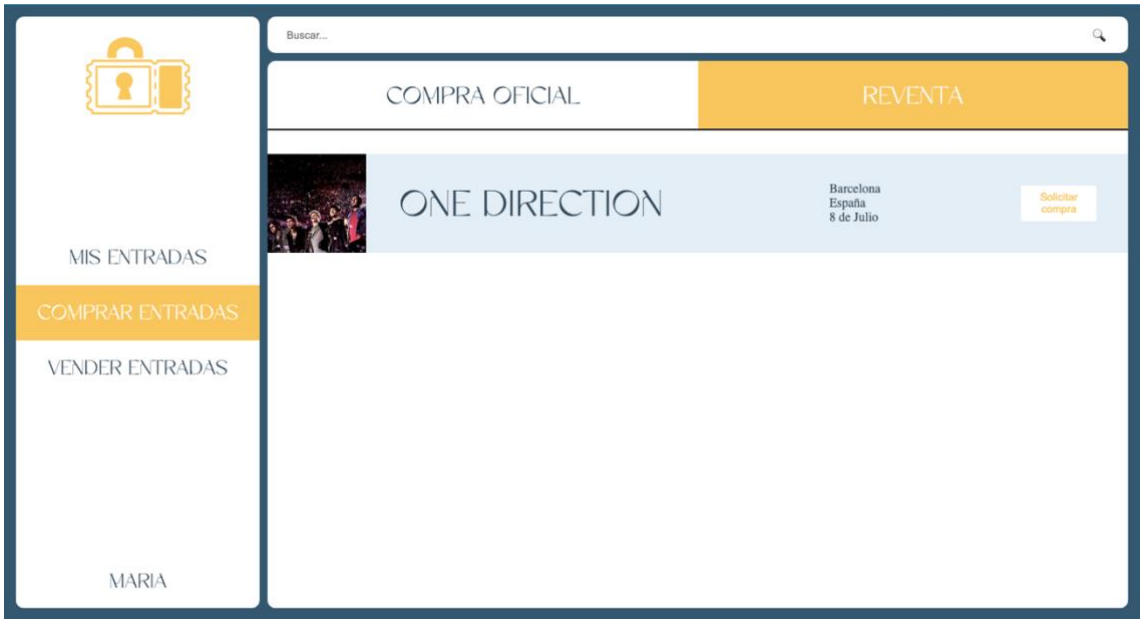

Ilustración 51. Captura de la plantilla "Comprar Entradas" en la opción reventa.

La diferencia que se presenta, y que permite que la entrada se compre en reventa a otro usuario es la presentación de un formulario distinto al de la opción de compra oficial. En este caso, el método "/compra" ha recogido el valor que indica que la compra es de reventa, por lo que, una vez se pulse el botón para solicitar la compra, el usuario será redirigido al método "/solicitar".

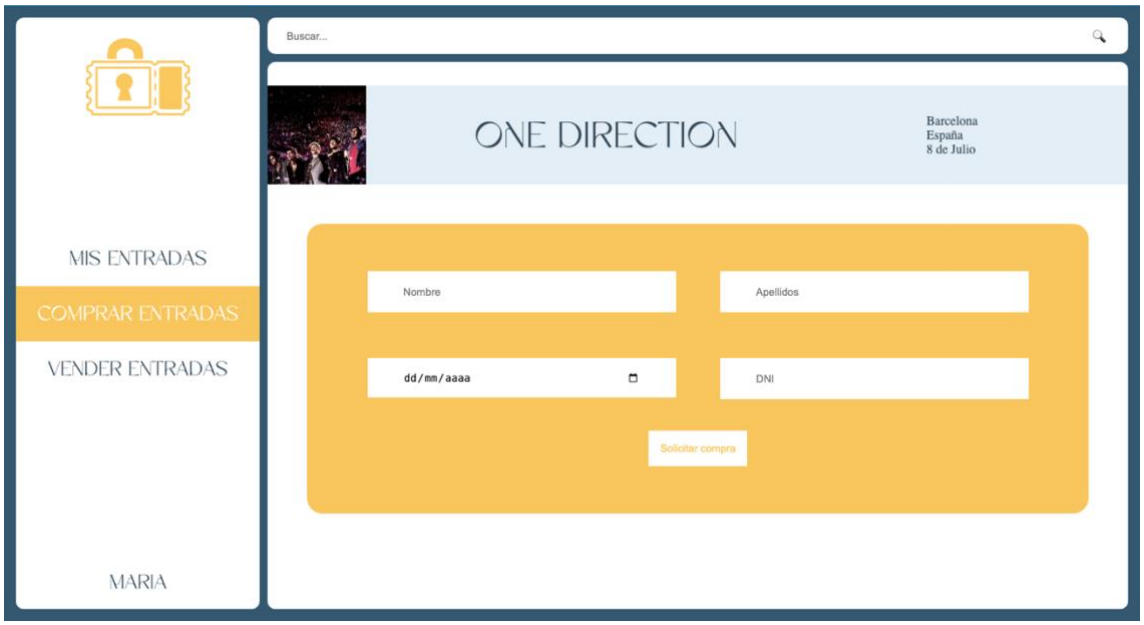

Ilustración 52. Formulario de compra de una entrada de reventa.

Al igual que en la compra oficial, obtenemos los valores del formulario, y creamos un objeto que enviaremos al servidor para que lo introduzca en el atributo del contrato correspondiente. Creamos la ruta, en este caso al endpoint del servidor "/buy_ticket" introduciendo como parámetro la dirección del contrato y realizamos la solicitud post.

```
sol_object = {
    'name': name,
    'surname': surname,
    'buyer': user,
    'born': born,
    'dni': dni
}

sol_address = "{}/buy_ticket/{}".format(CONNECTED_NODE_ADDRESS, contract)
response = requests.post(sol_address,
                         json=sol_object,
                         headers={'Content-type': 'application/json'})
```

Ilustración 53. Solicitud de contacto con la blockchain para solicitar la compra de una entrada de reventa.

La función de "/buy_ticket" también tiene como parámetro la dirección del contrato. Se obtiene el objeto JSON enviado a través de la solicitud post y se comprueba que se encuentra el campo requerido "buyer", es decir, el comprador de la entrada. Se obtiene el contrato con la función de la clase "blockchain" "get_contract" y se llama a la función de la clase del contrato "buy_ticket" para actualizar los valores del comprador y sus datos.

```
@app.route('/buy_ticket/<contract_address>', methods=['POST'])
def buy_ticket(contract_address):
    data = request.get_json()
    required_fields = ['buyer']
    for field in required_fields:
        if not data.get(field):
            return "Missing fields", 400

    contract = blockchain.get_contract(contract_address)
    if not contract:
        return "Contract not found", 404

    try:
        contract.buy_ticket(data['buyer'], data)
    except ValueError as e:
        return str(e), 400

    return "Ticket purchased, pending confirmation", 200
```

Ilustración 54. Código de la función "buy_ticket" del contrato en la blockchain.

Si el resultado es exitoso, insertamos en el nombre del comprador en la entrada correspondiente del documento de reventa y modificamos el estado en la entrada del vendedor a "Compra solicitada".

6.2.3 Aceptar compra de entradas de reventa

Una vez algún usuario ha solicitado la compra de nuestra entrada solo faltará validar la transacción.

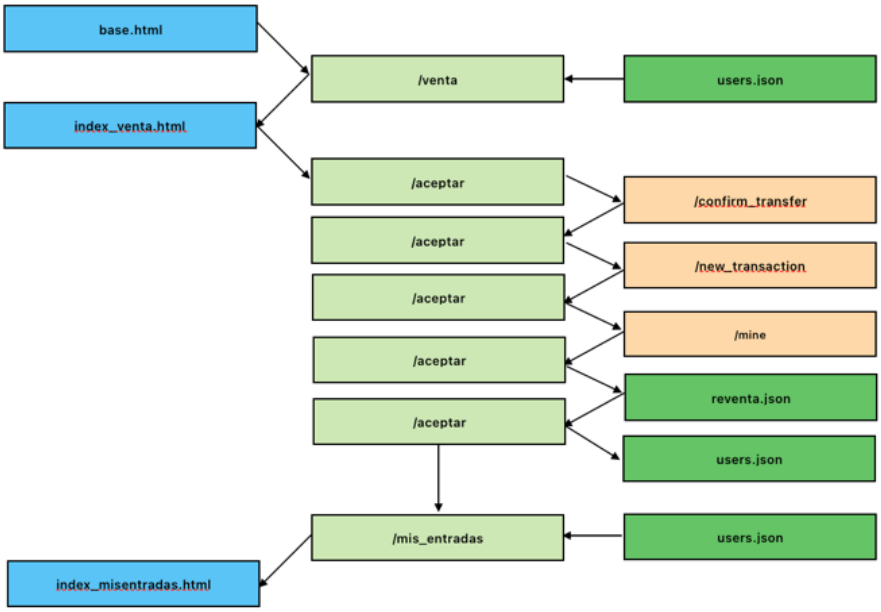

Ilustración 55. Diagrama del funcionamiento de aceptar una compra de reventa.

Ilustración 56. Captura de la pestaña de "Vender Entradas" con una compra por aceptar.

En la pestaña de vender entradas del usuario que la ha puesto a la venta habrá cambiado el botón de “Puesta en venta” a “Aceptar compra”. Si pulsamos el botón nos lleva a la función “/aceptar” del back-end de la interfaz. En esta función se obtiene la dirección de contrato insertada en uno de los argumentos de la url. Se crea la dirección de ruta al servidor, en concreto al endpoint “/confirm_transfer” y se realiza una solicitud get, también empleando la dirección del contrato correspondiente.

```
contract = request.args.get('ticketHidden', '')
accept_address = "{}/confirm_transfer/{}".format(CONNECTED_NODE_ADDRESS, contract)
response = requests.get(accept_address)
response_json = response.json()
```

Ilustración 57. Solicitud de contacto con la blockchain para aceptar la transferencia.

Una vez nos encontramos en el método del contrato de la cadena de bloques, se obtiene el contrato deseado de la clase "blockchain" y se llama al endpoint "confirm_transfer" para confirmar la transferencia. En esta función se actualiza el valor del dueño de la entrada y se reinician los valores del contrato.

```
@app.route('/confirm_transfer/<contract_address>', methods=['GET'])
def confirm_transfer(contract_address):

    contract = blockchain.get_contract(contract_address)
    if not contract:
        return "Contract not found", 404

    try:
        old_holder, data, ticket = contract.confirm_transfer()
    except ValueError as e:
        return str(e), 400
```

Ilustración 58. Función "confirm_transfer" del contrato en la blockchain.

Tras confirmar la transferencia se crea un objeto nuevo que será registrado en la blockchain como una nueva transacción para que quede constancia de la compra del nuevo propietario, al igual que en la compra de entradas original. Ese registro se llamará desde el Back-End de la interfaz una vez se obtenga como válido el resultado de la operación de aceptación de la compra.

Una vez se ha realizado la operación de traspaso con éxito, se obtiene del contrato del servidor la entrada con los nuevos datos del comprador. Desde el Back-End de la interfaz se añade esta entrada a una nueva transacción de la blockchain tal y como realizamos al llevar a cabo una compra normal, se mina este nuevo bloque y se almacena en la lista de entradas del comprador.

El usuario que ha adquirido esta entrada de reventa puede ejercer el derecho de reventa de la misma, y así sucesivamente de forma ilimitada. De esta forma, la gestión de la entrada es libre y siempre está garantizada la trazabilidad y transparencia en la sucesión de titulares de la entrada.

Capítulo 7. Consumo energético

La eficiencia energética se ha convertido en un factor crucial en el desarrollo tecnológico actual y el consumo de energía y de recursos es uno de los aspectos más debatidos y criticados a la hora de implementar nuevos avances. Uno de estos casos en los que se ha focalizado la atención del consumo energético son cadenas de bloques, especialmente aquellas que utilizan como método de consenso la Prueba de Trabajo (PoW) debido a su elevado coste computacional, mencionado en [12]. Por ello, se va a realizar en este capítulo un estudio del consumo energético de la blockchain implementada que utiliza como protocolo de consenso la prueba de trabajo.

Tal y como muestra el trabajo [27], para comprender y optimizar el consumo energético es necesario conocer y analizar el comportamiento del procesador durante la realización de las tareas. Para ello vamos a emplear la ecuación (1) que define la energía consumida por ciclo de procesamiento y nos ayudará a calcular la energía consumida.

$$E_{\mathbf{cycle}}^{\mathbf{proc}}(f_{A_i}) = C_{\mathbf{eff}} \cdot f_{A_i}^2 = \frac{P_{\mathbf{proc}}(f_{\mathbf{CPU}})}{f_{\mathbf{CPU}}^3} f_{A_i}^2 \qquad \textbf{(1)}$$

Los términos de la ecuación explicados con más detalle son los siguientes:

- $E_{\mathbf{cycle}}^{\mathbf{proc}}(f_{A_i})$: es la energía consumida por la CPU durante un ciclo y dependiendo de su frecuencia de trabajo. Se mide en Julios (J).
- $C_{\mathbf{eff}}$: es la capacitancia efectiva de la CPU, que bajo ciertas condiciones del procesador es un valor constante.
- $f_{A_i}^2$: se trata de la frecuencia de la CPU durante la ejecución de una instrucción. Se mide en hercios (Hz).
- $P_{\mathbf{proc}}(f_{\mathbf{CPU}})$: es la potencia consumida por el procesador cuando opera a su máxima frecuencia y se mide en vatios (W).
- $f_{\mathbf{CPU}}$: es la frecuencia máxima de la CPU y también se mide en hercios (Hz).

La Potencia de Diseño Térmico (TDP) se utiliza como una aproximación de la potencia máxima $P_{\mathbf{proc}}(f_{\mathbf{CPU}})$ a la frecuencia máxima ($f_{\mathbf{CPU}}$). Para realizar el cálculo del consumo energético de las tareas realizadas por el programa, necesitamos saber algunas de las características del procesador del ordenador que las ejecuta, en concreto la frecuencia máxima del procesador y la Potencia de Diseño Térmico (TDP).

El ordenador empleado para realizar este trabajo es un MacBook Air de 2015 con un procesador 1,6 GHz Intel Core i5-5250U de doble núcleo [28]. Si consultamos las especificaciones de este procesador podemos obtener los valores necesarios para realizar el cálculo. En este caso, el valor de la de la frecuencia máxima del procesador son 1600 MHz y el de TDP 15 W.

En este caso, y debido a que el proyecto se ejecuta de manera local en el dispositivo, es únicamente un usuario el que interactúa con la interfaz. No es posible que dos usuarios accedan al sistema y realicen gestiones de manera simultánea. Por ello, la misma interfaz, el Back-End respectivo de la plataforma y el servidor blockchain con un único minero son los tres componentes que van a proporcionar la información del consumo que supone la ejecución de sus métodos.

Una vez contamos con estos datos, implementamos el Back-End de la interfaz una función que calcula el consumo energético de cada método, de esta manera calculamos cuál genera más gasto.

Para implementar la función primero definimos como variables globales la Frecuencia Base y el valor de TDP, así como la frecuencia máxima de la CPU que calculamos empleando el módulo de Python psutil. Una vez hemos definido las variables, creamos una función "calcular_energia(elapsed)" que realiza el cálculo matemático definido en la ecuación previa dados los ciclos de CPU necesarios para realizar la tarea.

```
def calcular_energia(elapsed):
    cpu_current_freq = psutil.cpu_freq().current
    C_eff = TDP / (MAX_FREQ ** 3)
    E_cycle_proc = C_eff * (cpu_current_freq ** 2)
    total_energy = E_cycle_proc * elapsed
    return total_energy
```

Ilustración 59. Función de cálculo de energía de una tarea.

Por último, en cada uno de los métodos hemos de contar los ciclos empleados. Nada más entrar comenzamos la cuenta con la función "count()" del módulo "x" y, previo a devolver la plantilla correspondiente se finaliza la cuenta con la función "count_end()" y se calcula la diferencia restándole el inicio a este valor. Por último, se llama a la función "calcular_energia", que nos dará el resultado final de la energía consumida en la ejecución del método.

Tras completar la implementación iniciamos el programa con normalidad e interactuamos con la plataforma utilizando todas las pestañas y gestionando las entradas de todas las formas posibles para recopilar los datos. Una vez hemos recopilado múltiples datos de todos los métodos calculamos, además, los intervalos de confianza de cada una de las tareas.

Un intervalo de confianza es un intervalo que contiene un parámetro con cierto grado de seguridad con respecto a la muestra, es decir, la probabilidad de que el parámetro se encuentre en el rango es el grado de confianza del intervalo. En este caso hemos seleccionado un grado de confianza del 95%. Para calcular los intervalos, hemos implementado el siguiente código en Python, que define los límites del intervalo y los establece con respecto a la media:

```
def confidence_intervals(data, confidence=0.95):

    tbounds = t.interval(confidence, len(data) - 1)

    ci = tbounds[1] * np.std(data,ddof=1) / np.sqrt(len(data))

    return ci
```

Ilustración 60. Función de los intervalos de confianza.

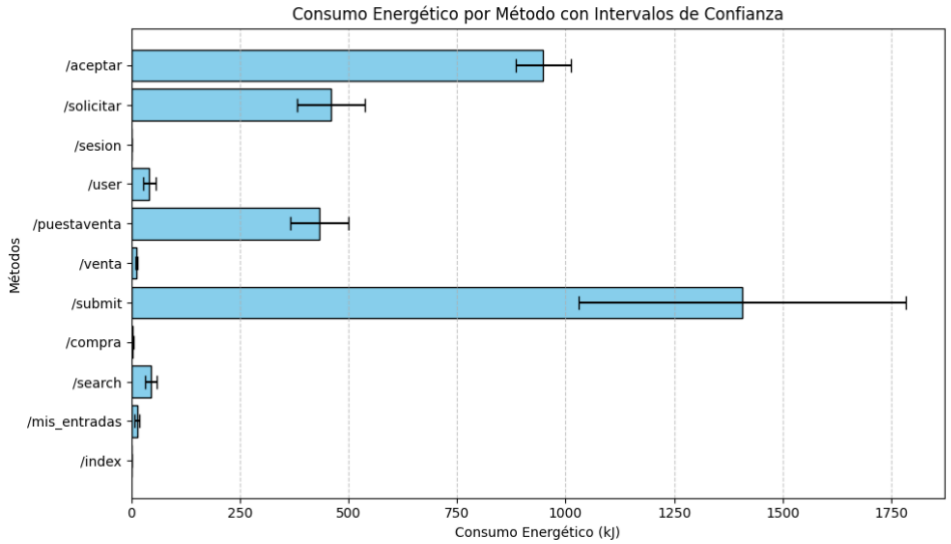

Ilustración 61. Gráfico del consumo energético de los métodos de la Plataforma Web.

Si analizamos los datos podemos ver que los métodos que únicamente operan con la parte Front-End de la plataforma, redirigiendo pestañas o mostrando datos, son los que menos energía consumen. Los métodos "/index", "/compra" y "/sesion" que únicamente devuelven una plantilla apenas consumen. Los métodos "/search", "/user", "/venta" y "/mis_entradas", acceden a documentos JSON para recopilar datos y devolverlos a la interfaz, por lo que su consumo es ligeramente mayor.

Sin embargo, el gráfico aumenta en los métodos que interactúan con la blockchain, estos son "/solicitar", "/puestaventa", "/aceptar" y "/submit". Estos dos últimos tienen el mayor consumo de todos los métodos ya que ambos registran nuevas transacciones en la blockchain y utilizan la función de la prueba de trabajo, que como hemos visto antes, es uno de los procesos que más recursos consume.

Ahora vamos a ver más a fondo cuáles son los procesos de la blockchain responsables de tal consumo energético. Para ello, implementamos la misma función que hemos introducido en el Back-End de la interfaz, pero en este caso lo aplicamos al servidor blockchain.

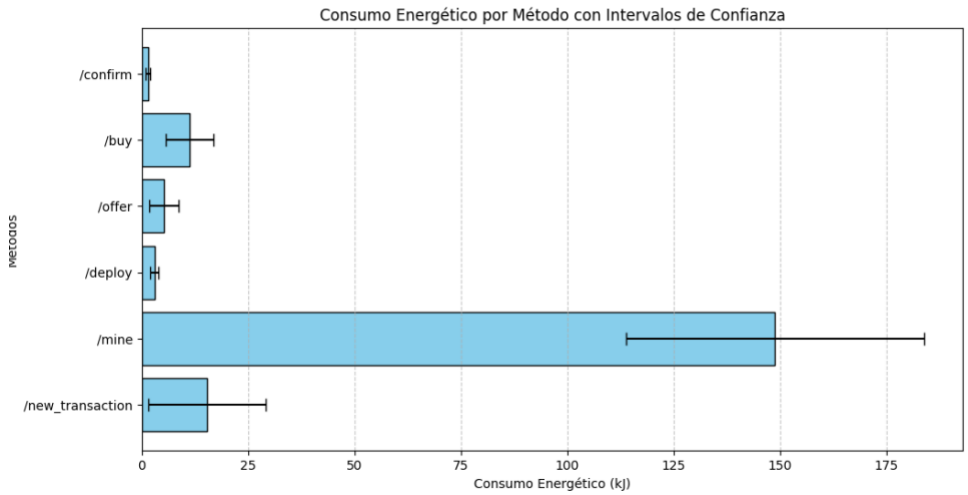

Ilustración 62. Gráfico del consumo energético de los métodos del servidor blockchain.

Como presuponíamos tras explicar el funcionamiento de las cadenas de bloques, el método del servidor blockchain que más energía consume es el de minado, ya que emplea el protocolo de consenso "Proof of Work", cuyas operaciones son las que más recursos computacionales requieren.

Respecto a los intervalos de confianza, la mayoría se acercan bastante al valor medio con excepción del método "/submit" directamente relacionado con la ruta "/mine" de la blockchain. Los intervalos de confianza son bastante más amplios en estos casos. Esto puede darse ya que, aunque la prueba de trabajo sea un proceso costoso, no siempre tiene por qué ser así. Encontrar un hash válido que cumpla con la dificultad puede obtenerse en unos pocos intentos o por el contrario llevar a cabo una gran cantidad de operaciones matemáticas para lograr que el valor cumpla con los requisitos, generando un consumo energético mayor. La variedad en el número de intentos en la prueba de trabajo de cada transacción genera una amplia variedad en el resultado de la energía consumida.

Tras este análisis, y viendo el incremento en la energía consumida por los procesos que emplean la cadena de bloques en comparación al resto, podemos ser conscientes del reto que supone implementar métodos que permitan reducir el consumo energético a la hora de emplear cadenas de bloques y los avances que se han ido produciendo respecto a protocolos de consenso alternativos como "PoS", cuyo consumo energético es menor.

Capítulo 8. Conclusión

8.1 Evaluación de los objetivos

Para concluir este trabajo, la creación de una plataforma web basada en blockchain es un avance significativo en la lucha contra el fraude y la falsificación de la compraventa online. Se ha demostrado como la blockchain puede proporcionar un sistema transparente, seguro e inmutable garantizando la integridad de cada entrada vendida y revendida.

La implementación de una blockchain propia ha permitido estudiar más de cerca el comportamiento, ejercer un mayor sobre las operaciones y la toma de decisiones acerca de los protocolos a utilizar, adaptando la cadena de bloques a las necesidades de la plataforma implementada. Realizar esto en una cadena de bloques de un tercero no habría sido posible al mismo nivel.

Además, se ha diseñado una plataforma web con una interfaz intuitiva y funcional, imprescindible para facilitar la interacción del usuario con la blockchain y garantizando una experiencia de usuario satisfactoria. Esta plataforma cumple con los objetivos de seguridad y transparencia gracias a la implementación de los contratos inteligentes y contribuye a ofrecer una solución más justa tanto a los compradores como a los vendedores.

Por último, gracias al estudio del consumo energético se ha podido evaluar el gasto de recursos real de la blockchain implantada y proponer soluciones para mejorar el impacto de las cadenas de bloques.

En definitiva, el proyecto presenta un modelo que podría servir de base para futuras aplicaciones basadas en blockchain y tecnologías descentralizadas tanto en este sector como muchos otros gracias a todas las utilidades que hemos visto.

8.2 Futuras implementaciones

En el Trabajo Fin de Grado se ha implementado la interfaz web, así como su interacción con la blockchain para llevar a cabo los registros pertinentes, demostrando así su utilidad en la venta y reventa segura de entradas. Sin embargo, para completar el sistema desarrollado en su totalidad, sería necesario añadir una serie de módulos cuyas funciones son imprescindibles para poder implementar la plataforma completamente y garantizar su uso efectivo.

En primer lugar, es fundamental introducir un sistema de pago online seguro y verificado, permitiendo completar la transacción de manera confiable a través de la web al momento de realizar la compra.

Para evitar que la sesión esté iniciada en diferentes dispositivos de manera simultánea, se debe aplicar una restricción que detecte si el usuario tiene la sesión abierta actualmente, y permitir la opción de cerrarla o evitar que se inicie en una segunda ubicación.

Además, sería necesario implementar un módulo de selección de asientos o sectores para cada recinto, permitiendo la posibilidad de elección de entradas a sus respectivos precios, según donde se quiera disfrutar del evento, y de la compra de varias entradas simultáneamente permitiendo asignarlas al usuario final.

También sería conveniente un sistema robusto de autenticación a la hora de iniciar sesión, como por ejemplo tecnologías avanzadas como las autenticaciones biométricas, o un sistema de autenticación de doble factor en el que se utilicen códigos de verificación o múltiples dispositivos.

Por último, para validar las entradas una vez se pretenda acceder al evento, es preciso desarrollar una aplicación móvil que permita validación de las entradas mediante sistemas de autenticación digital. Esta aplicación garantizaría que solo las entradas registradas en la blockchain de la aplicación con su respectivo comprador sean aceptadas.

Respecto a mejoras en la cadena de bloques, convendría ampliar la cantidad de nodos participantes en la blockchain para garantizar una mayor robustez, que asegure la validez de la cadena aplicando los protocolos de consenso. Para ello sería necesario implementar técnicas de sincronización distribuidas que sincronicen y coordinen los nodos participantes de tal manera que el sistema no se vea afectado por el proceso simultáneo de cada uno.

8.3 Conclusión personal

Para desarrollar el trabajo he querido aplicar los conocimientos y la formación adquirida a lo largo de mis estudios en el Grado Tecnología Digital y Multimedia, fundamentalmente en materias que me han resultado interesantes como “Programación”, “Seguridad” y “Tecnologías Web”. He aplicado lo aprendido sintetizándolo en un proyecto que abarque todas estas ciencias materializándose en un producto de uso cotidiano y tecnológicamente accesible. De esta manera, unas materias complejas se transforman en un elemento útil para cualquier usuario o empresa que quisiera aplicarla al alcance de un botón.

En este proceso, además de la base formativa con la que partía, he necesitado aprender nuevas tecnologías y conceptos con los que no había tratado con anterioridad. Esta situación ha supuesto un reto a la hora de acceder a estos nuevos conocimientos, asimilarlos, conocerlos y aplicarlos en el desarrollo del trabajo para ofrecer una solución al problema de la estafa en la gestión y venta de entradas.

Del planteamiento de la idea original hasta tener finalizado todo el sistema implementado, he tenido que ir resolviendo dificultades y complicaciones que me han ralentizado el desarrollo hasta dar con la solución deseada. Esto me ha servido para mejorar mi competencia en la resolución de problemas y poner a prueba mi capacidad y tenacidad para el logro del objetivo.

Uno de los retos que me ha supuesto el Trabajo Fin de Grado ha sido poner en práctica la redacción y documentación de un trabajo de investigación aplicado a una materia tecnológica de estas características.

Todo esto no habría sido posible sin contar con la dirección y el asesoramiento de mis tutores José Manuel Giménez Guzmán y María Soledad García Valls, quienes han aportado sus conocimientos y su tiempo para presarme la ayuda necesaria, incluso en su periodo vacacional. Por ello mi más profundo agradecimiento.

En consecuencia, tengo que considerar que el resultado obtenido, teniendo en cuenta que se ajusta a la idea original que me planteé, me hace sentir satisfecha y orgullosa de haberlo conseguido. Además, creo que es un trabajo con posibilidades de desarrollos futuros, muy real, accesible, y a la orden del día, tanto para las nuevas generaciones como para las empresas del sector.

Finalmente, este trabajo ha confirmado mi interés en un campo muy actual y la inquietud de querer seguir aprendiendo y formándome en una materia necesaria y demandada en la era digital en la que vivimos. Probablemente, dentro de unos años, pueda valorar con más perspectiva lo importante que ha sido para mí dedicar tantas horas y tanto esfuerzo a este Trabajo Fin de Grado.

Capítulo 9. Bibliografía

[1] TicketSwap. TicketSwap: Compra y venta de entradas de segunda mano. Disponible en: https://www.ticketswap.es (accedido el 14 de agosto 2024)

[2] IBM. Qué es la tecnología blockchain. 2024. Disponible en: https://www.ibm.com/es-es/topics/blockchain

[3] Cámara de Comercio de Valencia. Blockchain: Qué es y qué ventajas tiene [Internet]. TIC Negocios. Disponible en: https://ticnegocios.camaravalencia.com/servicios/tendencias/blockchain-que-es-y-que-ventajas-tiene/ (accedido el 26 de agosto 2024).

[4] AVG. ¿Qué es la tecnología blockchain? AVG Signal; 2022. Disponible en: https://www.avg.com/es/signal/what-is-blockchain

[5] Matesanz V. ¿Qué es Blockchain y criptomonedas? Guía fácil. Finect; 2023. Disponible en: https://www.finect.com/usuario/vanesamatesanz/articulos/que-blockchain-criptomonedas-guia-facil

[6] Baldoni R., Hélary J.-M., Raynal M., Tangui L. Consensus in Byzantine asynchronous systems. J. Discrete Algorithms, 1 (2) (2003), pp. 185-210,

[7] Bit2Me Academy. ¿Qué es Proof of Work (PoW)? 2023. Disponible en: https://academy.bit2me.com/que-es-proof-of-work-pow/

[8] Bit2Me Academy. ¿Qué es Proof of Stake (PoS)? 2023. Disponible en: https://academy.bit2me.com/que-es-proof-of-stake-pos/

[9] BBVA. De la verificación de contraseña a la firma electrónica: el secreto está en el hash. 2023. Disponible en: https://www.bbva.com/es/innovacion/de-la-verificacion-de-contrasena-a-la-firma-electronica-el-secreto-esta-en-el-hash/

[10] Cysae. El minado en blockchain. 2018. Disponible en: https://www.cysae.com/el-minado-en-blockchain/

[11] Conquer Blocks. ¿Redes blockchain: se pueden hackear? Disponible en: https://www.conquerblocks.com/post/redes-blockchain-se-pueden-hackear (accedido el 29 de agosto 2024)

[12] García-Valls M, Chirivella-Ciruelos AM. CoTwin: Collaborative improvement of digital twins enabled by blockchain. Future Gener Comput Syst. Valencia, España 2024;157:408-421.

[13] Ethereum.org. What is Ethereum? 2024. Disponible en: https://ethereum.org/en/what-is-ethereum/

[14] Unjore A. Desarrollo de una aplicación blockchain en Python. Arnaud Unjore; 2020. Disponible en: https://arnaudunjo.com/es/2020/12/17/desarrollo-de-una-aplicacion-blockchain-en-python/

[15] Web3.py. Web3.py Documentation. 2023. Disponible en: https://web3py.readthedocs.io/en/stable/

[16] Chirivella-Ciruelos AM, García-Valls M. Provenance Verification of Smart Contracts: Analysing the Cost of Ensuring Authenticity over the Logic Hosted in Blockchain Networks [Internet]. Valencia, España; 31 de Diciembre 2023. Disponible en: https://doi.org/10.3390/info15010024

[17] Solidity Team. Solidity documentation: Version 0.8.26. 2023. Disponible en: https://docs.soliditylang.org/en/v0.8.26/?color=light

[18] World Wide Web Consortium (W3C). HTML5: A vocabulary and associated APIs for HTML and XHTML. 2014. Disponible en: https://www.w3.org/TR/html5/

[19] World Wide Web Consortium (W3C). CSS (Cascading Style Sheets). 2015. Disponible en: https://www.w3.org/wiki/Es/CSS

[20] MDN Web Docs. Introduction to JavaScript. Mozilla Developer Network; 2024. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#what_is_javascript

[21] Web3.js. Web3.js Documentation. 2023. Disponible en: https://docs.web3js.org

[22] Node.js. About Node.js. Node.js Foundation. Disponible en: https://nodejs.org/en/about (accedido el 15 de agosto 2024)

[23] Python Software Foundation. Python 3 documentation. 2024. Disponible en: https://docs.python.org/3/

[24] Django Software Foundation. Django: The Web framework for perfectionists with deadlines. Disponible en: https://www.djangoproject.com (accedido el 29 de agosto 2024)

[25] Pallets Projects. Flask: A lightweight WSGI web application framework. Disponible en: https://flask.palletsprojects.com/en/3.0.x/ (accedido el 29 de agosto 2024)

[26] JSON.org. Introducción a JSON. Disponible en: https://www.json.org/json-es.html

[27] Gimenez-Guzman JM, Leyva-Mayorga I, Popovski P. Goal-Oriented Source Coding and Filtering for Vehicular Communications. IEEE Internet Things J. 2024;15(1):24.

[28] Intel Corporation. Intel Core i5-5250U Processor (3M Cache, up to 2.70 GHz). Disponible en: https://ark.intel.com/content/www/us/en/ark/products/84984/intel-core-i5-5250u-processor-3m-cache-up-to-2-70-ghz.html (accedido el 28 de agosto 2024).