UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

Uso de la tecnología blockchain para la gestión de reservas hoteleras

Use of blockchain technology for hotel reservation management

Realizado por

José García Reche

Tutorizado por

Rodrigo Román Castro

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO 2022

Agradecimientos

En primer lugar, me gustaría agradecer a mis padres todo lo que han hecho por mí, en todo momento habéis estado ahí confiando y apoyando como los que más, en los buenos momentos y en los no tan buenos. A mi abuela materna, mis abuelos paternos y a mi hermano. Gracias por estar siempre ahí apoyándome en cada momento.

También, deseo expresar mi agradecimiento a mi tutor de trabajo de fin de grado, Rodrigo Román Castro, por aceptar tutorizarme, por su dedicación y gran ayuda, ya que me ha guiado y atendido en cualquier momento. Después de todo, aunque se pensara que no íbamos a llegar a tiempo o íbamos a estar justos, hemos conseguido terminarlo sin ningún problema, con trabajo duro.

A mi familia, a mis suegros por acogerme como si fuese un hijo, os estaré siempre agradecido, a compañeros de piso y clase, a mis amigos y cada una de las personas que siempre ha estado ahí apoyándome en todos años y confiando en todo momento, gracias.

Por último y especialmente a Diana, gracias por mantener la cordura en mí. Aguantándome siempre incluso ni cuando yo mismo me aguanto. Me has entendido, me has ayudado y me has dado ese empujón que en ocasiones me ha hecho falta. Gracias por acompañarme en esta aventura y en todas las que nos quedan. Sin ti, mi vida en Málaga no hubiese sido tan fácil.

Gracias.

Resumen

Este proyecto consta de dos partes bien diferenciadas. La primera de ellas es el estudio analítico de las distintas plataformas que usan la tecnología blockchain. Esto es así porque existen multitud de estas plataformas blockchain que poseen diversas características y capacidades a nivel de usabilidad, escalabilidad y rendimiento. Al ser una tecnología en fase de expansión, si no se realizase dicho estudio previo la viabilidad de la solución podría estar en peligro. Para ello, primero se han desarrollado las bases de esta tecnología y los conceptos relacionados con ella. Seguidamente, se han analizado de forma teórica las soluciones propuestas que conforman las distintas blockchain candidatas para realizar el proyecto. Para posteriormente, desarrollar un caso práctico como prueba de concepto para validar el análisis anterior, demostrando así cuál es la blockchain que más se adecúa a nuestro proyecto.

La segunda parte del proyecto consiste en el desarrollo de la prueba de concepto, donde una aplicación web será capaz de gestionar las distintas reservas hoteleras mediante el uso de la tecnología blockchain. Para ello, es necesario desplegar un smart contract en la Testnet de Avalanche; y posteriormente, que la aplicación web sea capaz de interactuar con el estado del smart contract.

Palabras clave: Blockchain – Contratos Inteligentes – Plataformas web – Avalanche - Criptomonedas

Abstract

This project consists of two well-differentiated parts. The first of them is the analytical study of the different platforms that use blockchain technology. This is because there are many of these blockchain platforms that have different characteristics and capabilities in terms of usability, scalability and performance. Being a technology in the expansion phase, if this preliminary study is not carried out, the viability of the solution could be in danger. To do this, first the bases of this technology and the concepts related to it have been developed. Then, the proposed solutions that make up the different candidate blockchains to carry out the project have been analyzed theoretically. Afterward, we will develop a practical case as a proof of concept to validate the previous analysis, for demonstrating which is the blockchain that best adapts to our project.

The second part of the project consists of the development of the proof of concept, where a web application will be able to manage the different hotel reservations using blockchain technology. To do this, it is necessary to deploy a smart contract on the Avalanche Testnet; and subsequently, that the web application can interact with the state of the smart contract.

Keywords: Blockchain – Smart Contracts – Web platforms – Avalanche - Cryptocurrency

Índice

1

Introducción

En este apartado se explicará el contenido de la memoria, así como antecedentes, motivación, objetivos y conceptos clave.

1.1 Motivación

La industria hotelera es un sector de servicios emergentes en varias economías a nivel mundial. El sector hotelero está estrechamente relacionado con el sector turístico, uno de los principales motores económicos de España. La industria hotelera ha ido evolucionando desde los "Boomers" hasta la "Generación Z", pasando desde el albergue y las habitaciones de hotel hasta Airbnb. Se dice que la tecnología blockchain tiene el potencial de interrumpir la intermediación del proceso de reserva hotelera al eliminar las comisiones. La industria hotelera 3.0 está evolucionando desde la negociación de un precio hasta varios detalles, a través de la realidad aumentada como puede ser una visita virtual de las habitaciones a través del móvil o PC. Por tanto, estas diversas empresas emergentes se están orientando a ofrecer una experiencia personalizada para sus consumidores.

El uso de la tecnología blockchain nos proporcionaría un sistema de reservas fácil de usar y transparente, con registro auditables y sin errores en tiempo real, también mejoraría la calidad del servicio en la distribución de habitaciones, aunque es necesario tener una conectividad a internet de 24 horas al día, 7 días a la semana. Sin embargo, uno de los principales problemas de esta tecnología es el uso excesivo de electricidad

debido al gran coste computacional que podrían tener ciertas operaciones. Por ello, han surgido nuevas tecnologías blockchain que optimizan este problema y no requieren de un gran coste energético.

1.2 Objetivo

El presente proyecto tiene dos principales objetivos. Primero, realizar un estudio previo para encontrar la plataforma blockchain basada en Solidity (el lenguaje de programación más utilizado para los contratos inteligentes) que se ajuste mejor a los requisitos del proyecto. Segundo, desarrollar un caso práctico como prueba de concepto que valide el análisis anterior, demostrando como la blockchain más óptima (según ese estudio anterior) puede formalizar una reserva entre el usuario y la cadena hotelera en cuestión. De tal manera, que tanto el pago como la confirmación de la reserva sea de forma automática y en el caso de que se solicitara un reembolso, se proceda tal y como haya sido configurado los contratos inteligentes consensuados previamente entre ambas partes.

1.3 Metodología

En este apartado se explicará cuál ha sido la metodología utilizada durante el desarrollo del proyecto.

Para el cumplimiento del primer objetivo, se procederá a realizar un estudio analítico que analice de forma exhaustiva los requisitos funcionales y no funcionales de las distintas plataformas blockchain (p.ej. Ethereum, Avalanche, Polygon, Binance Smart Chain, Capa dos de Ethereum, Hyperledger Besu…, etcétera) que poseen diversas características y capacidades a nivel de usabilidad, escalabilidad y rendimiento (p.ej. costes/transacción, tiempo/transacción, velocidad/transacción), y es necesario identificar cuál(es) de ellas se adecua(n) más a las necesidades de nuestra aplicación y pueda(n) tener una mayor proyección en el futuro, condición necesaria para saber el

nivel de escalabilidad de nuestra aplicación. Al ser una tecnología en fase de expansión, si no se realizase dicho estudio previo la viabilidad de la solución podría estar en peligro.

1.4 Conceptos clave

En esta sección se explican los conceptos claves que se utilizarán en adelante en el proyecto.

Blockchain: Base de datos distribuida que se comparte entre los nodos de una red informática. La información almacenada es consensuada por los nodos y almacenada en bloques **[1].**

Smart Contract: Es un programa autoejecutable en el que los términos del acuerdo entre el comprador y vendedor se escriben directamente en líneas de código. El código controla la ejecución y las transacciones son rastreables e irreversibles **[2].**

Transacción: Es un acuerdo completo entre un comprador y un vendedor para intercambiar bienes, servicios o activos financieros a cambio de dinero físico o virtual **[3].**

Aplicación Distribuida: Aplicaciones de software que se almacenan y ejecutan principalmente en plataformas de computación en la nube y que se ejecutan en múltiples sistemas simultáneamente. Estos sistemas operan en la misma red y se comunican entre sí en un esfuerzo por completar una tarea determinada **[4].**

Protocolo de Consenso: Es un método específico para verificar si una transacción es verdadera o no. Proporciona un método de revisión y confirmación de qué datos deben agregarse al registro de una blockchain **[5].**

1.5 Acrónimos

Durante el presente proyecto nos encontraremos con numerosos acrónimos, todos están recogidos en esta sección.

DApp – Distributed Application

P2P – Peer to Peer

PoW – Proof of Work

ETH – Ethereum

PoS – Proof of Stake

PoA – Proof of Authority

IDE – Integrated Development Environment

TPS– Transactions Per Second

DeFi – Decentralized Finance

ZK Rollups – Zero-Knowledger Rollups

BTC – Bitcoin

AI – Artificial Intelligence

IoT – Internet of Things

POS – Point of Sale

BSC – Binance Smart Chain

DAO – Decentralized Autonomous Organization

DoS – Denial of Service

X-Chain – Exchange Chain

C-Chain – Contract Chain

P-Chain – Platform Chain

EIP – Ethereum Improvement Proposals

BC – Binance Chain

ICO – Initial Coin Offering

PoSA – Proof of Stake Authority

IBFT – Istanbul Byzantine Fault Tolerance

API – Application Programming Interface

2

Estudio preliminar

En este apartado se mostrará una investigación sobre la Blockchain y los Smart Contracts, para comprender como funcionan, su hardware, como funcionan en la red, conocer sus aspectos técnicos y los principales éxitos en varios casos de uso.

2.1 Blockchain

En 2008, un artículo de Satoshi Nakamoto introdujo la tecnología blockchain y la criptomoneda más famosa hasta la fecha, Bitcoin. La tecnología se basa en tres conceptos: redes P2P, criptografía de clave pública y métodos de verificación de transacciones. Estos tres conceptos son claves para crear un registro de información, como por ejemplo las transacciones entre divisas, que serán almacenadas y compartidas en múltiples nodos (también llamados bloques). La tecnología blockchain se inventó a partir del concepto del "Árbol de Merkel", un sistema que permite enviar datos verificados de un sistema informático a otro, asegurándose de que los datos sean correctos debiendo ser aceptados por todos o por la mayoría de los participantes de la red para verificar su exactitud. Este proceso, se puede hacer a través de un algoritmo de consenso automatizado. Mientras que no se puede acceder a una base de datos cuando el servidor falla o pueda tener datos corruptos, la tecnología blockchain asegura que incluso si hay un problema con uno de sus nodos, los datos aún son accesibles a través de todos los demás nodos. Si bien esto no significa una seguridad de datos completa, es una gran mejora respecto a los servidores de datos tradicionales.

Figura 1. Símbolo de Bitcoin.

La blockchain es un sistema distribuido, que fue en primer lugar introducido por Bitcoin y las monedas digitales, promete un futuro innovador en colaboración con el mercado empresarial. Se podría definir como un libro de contabilidad que puede registrar transacciones de manera global, eficiente y permanente en una cadena de bloques con una marca de tiempo especificada.

Blockchain consta de un conjunto de nodos conectados a través de P2P (Peer to Peer). Los nodos pueden interactuar directamente sin ningún tercer nodo de confianza, lo que proporciona una infraestructura con transacciones más rápidas y económicas. En la blockchain, cada nodo puede crear una serie de transacciones, que pasan una función hash varias veces y luego se agrupan y se registran como bloques. Básicamente, la blockchain puede ser vista como una cadena de bloques que incluyen el hash del registro de transacciones, y solo se puede agregar un bloque a la cadena a la vez. Cada bloque posee una marca de tiempo y un enlace al bloque anterior, el cual se identifica por su valor hash. Cada nodo de la cadena de bloques está copiado en todos los nodos de la red. Por tanto, la información almacenada en la blockchain no se puede modificar ni eliminar, y la blockchain garantiza esta propiedad mediante la criptografía.

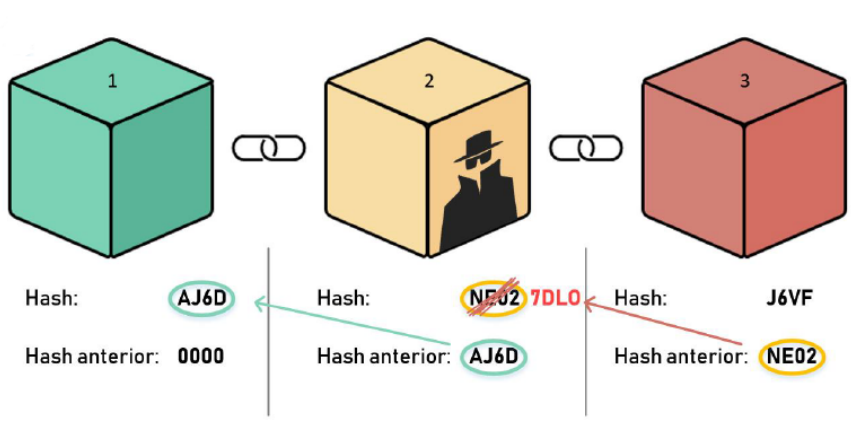


Fig. 2. Cadena de bloques manipulada

En el caso de la figura anterior, si modificáramos el bloque 2 (amarillo), provocará que en el campo “hash anterior” del bloque 3 no coincidan. Por tanto, modificar un bloque, provocará que todos los siguientes queden inválidos. El hashing permite detectar, pero no prevenir la manipulación de los bloques. Para prevenir este problema, se creó una técnica de consenso llamada “Proof-of-Work” (Prueba de trabajo) o comúnmente llamado minado. Consiste en que los usuarios resuelvan un problema matemático y deberán resolverlo lo más rápido posible. Este mecanismo de consenso es costoso de realizar y gasta muchos recursos. Al confiar en la honestidad de los mineros que encuentran ese valor que resuelva la prueba de trabajo, el bloque se añade a la blockchain. La complejidad del mecanismo aumenta proporcionalmente al número de usuarios que minan los bloques. Como es un proceso esencial para el funcionamiento de la blockchain, cada vez que uno de los nodos de la red encuentre una solución valida, este será recompensado en Bitcoins.

Este proceso crea un consenso entre todos los usuarios de la red. Por tanto, para poder manipular una blockchain, deberíamos ser capaces de modificar gran parte de los bloques de la cadena, y ser capaz de calcular todas las pruebas de trabajo y, por último, tomar el control del más del 50% de la red, algo que sería casi imposible.

La blockchain ofrece seguridad, confiabilidad, integridad y durabilidad. Además, reduce costos al eliminar a terceros y enviar transacciones más rápidas. También, proporciona una variedad de casos de uso, más allá de las simples transacciones financieras como contratos inteligentes, aplicaciones descentralizadas, identidad digital y mucho más.

Es cierto que las transacciones de blockchain son más lentas que las bases de datos centralizadas porque blockchain debe manejar más tareas en comparación con bases de datos centralizadas. Estas tareas adicionales incluyen: la verificación de la firma, el consenso de los mecanismos y la redundancia. Sin embargo, debemos recordar que blockchain tiene una ventaja determinista cuando se trata de proporcionar una forma confiable, robusta y segura de almacenar datos, sin que interfieran intermediarios y más tolerancia a fallos que las bases de datos centralizadas.

Sin embargo, existen numerosas voces críticas con esta tecnología que califican esta técnica como "inmadura" para hacer un cambio en la reserva de viajes a corto plazo y que la tecnología no está diseñada para la desintermediación y, por tanto, no tiene un valor añadido a las reservas de sitios web. Junto a esta opinión, existe también una corriente de expertos los cuáles concluyen que muchas empresas de blockchain que ahora surgen en la industria de los viajes trabajan principalmente para vender sus tokens y enriquecer a sus fundadores en lugar de crear un caso de negocio.

Además, que la información sea inmutable puede ser a su vez un problema. Si guardamos información personal, esta no podría eliminarse (violando así la ley de protección de datos). Otro punto negativo es la recuperación de la clave, ya que en el caso de que olvidemos nuestra clave, no sería posible recuperarla, por ello debemos tener especial cuidado con olvidar o perder las claves privadas. A todos estos problemas, se añaden otras desventajas importantes como la escalabilidad, lentitud en el

procesamiento de ciertas transacciones y el exponencial aumento del tamaño de memoria a medida que se van aumentando los usuarios.

2.1.1 Tipos de blockchain

Todos los tipos de blockchain se pueden caracterizar como *no permisionadas*, *permisionadas* o ambas. Las cadenas de bloques *no permisionadas* permiten a cualquier usuario unirse de forma pseudoanónima a la red de la cadena de bloques (es decir, convertirse en "nodos" de la red) y no restringen los derechos de los nodos en la red de la cadena de bloques.

Por el contrario, las cadenas de bloques *permisionadas* restringen el acceso a la red a ciertos nodos y también pueden restringir los derechos de esos nodos en esa red. Las identidades de los usuarios de una cadena de bloques *permisionada* son conocidas por los otros usuarios de esa cadena de bloques.

Las cadenas de bloques *no permisionadas* tienden a ser más seguras que las cadenas de bloques *permisionadas*, porque hay muchos nodos para validar transacciones y sería difícil para los atacantes coludirse en la red. Sin embargo, las cadenas de bloques *no permisionadas* también tienden a tener tiempos de procesamiento de transacciones prolongados debido a la gran cantidad de nodos y al gran tamaño de las transacciones.

Por otro lado, las cadenas de bloques *permisionadas* tienden a ser más eficientes. Debido a que el acceso a la red está restringido, hay menos nodos en la cadena de bloques, lo que se traduce en menos tiempo de procesamiento por transacción.

Como tantas cosas, también existen una serie de desventajas, y el tiempo de procesamiento reducido en las cadenas de bloques *permisionadas* no es una excepción:

la centralización de las cadenas de bloques *permisionadas* a diversas autoridades más centralizadas (ya sea un gobierno, una empresa, un grupo comercial o alguna otra entidad o grupo que otorgan el permiso a los nodos y crean las restricciones de la cadena de bloques) lo convierte en un sistema menos seguro y más propenso a las vulnerabilidades tradicionales de los sistemas informáticos. Cuantos menos nodos haya en una cadena de bloques, más fácil será para los atacantes coludirse, por lo que los administradores privados de la cadena de bloques deben asegurarse de que los nodos que agregan y verifican bloques son altamente confiables **[6].**

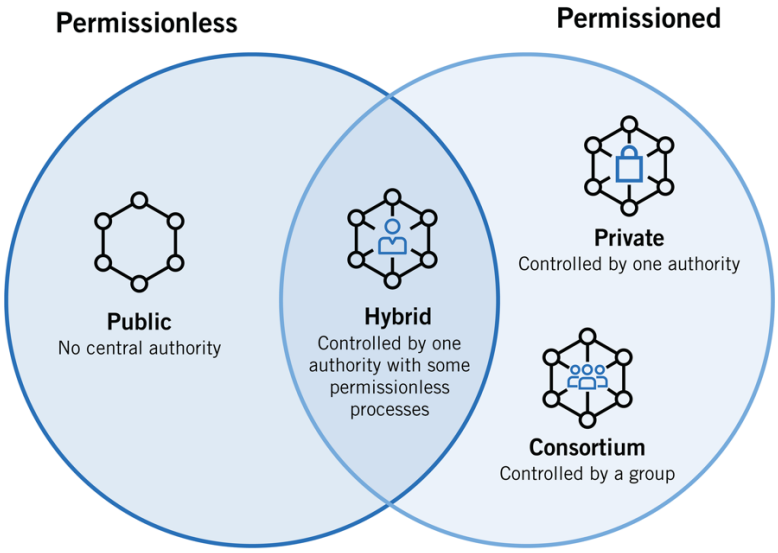


Figura 3. Tipos de Blockchain

Red pública frente a la privada

Principalmente se diferencian hasta cuatro tipos:

- **Blockchain pública.** Las cadenas de bloques públicas no tienen permiso por naturaleza, permiten que cualquiera se una y están completamente descentralizadas. Las cadenas de bloques públicas permiten que todos los nodos de la cadena de bloques tengan los mismos derechos para acceder a la cadena de bloques, crear nuevos bloques de datos y validar bloques de datos. Hasta la

fecha, las cadenas de bloques públicas se utilizan principalmente para intercambiar y extraer criptomonedas.

- **Blockchain privada.** Las blockchains privadas son cadenas de bloques autorizadas controladas por una sola organización. En una cadena de bloques privada, la autoridad central determina quién puede ser un nodo. La autoridad central tampoco otorga necesariamente a cada nodo los mismos derechos para realizar funciones. Las cadenas de bloques privadas solo están parcialmente descentralizadas porque el acceso público a estas cadenas de bloques está restringido. Algunos ejemplos de cadenas de bloques privadas son la red de intercambio de moneda virtual de empresa a empresa, Ripple e Hyperledger, un proyecto general de aplicaciones de cadena de bloques de código abierto.

  Tanto las cadenas de bloques privadas como las públicas tienen inconvenientes: las cadenas de bloques públicas tienden a tener tiempos de validación más largos para los nuevos datos que las cadenas de bloques privadas, y las cadenas de bloques privadas son más vulnerables al fraude y a los intrusos. Para abordar estos inconvenientes, se desarrollaron cadenas de bloques híbridas y de consorcio.

- **Blockchain de consorcio.** Las cadenas de bloques de consorcio son cadenas de bloques autorizadas dirigidas por un grupo de organizaciones, en lugar de una entidad, como en el caso de la cadena de bloques privada. Las cadenas de bloques de consorcio, por lo tanto, disfrutan de una mayor descentralización que las cadenas de bloques privadas, lo que resulta tener una mayor seguridad. Sin embargo, la creación de consorcios puede ser un proceso complicado, ya que requiere la cooperación entre varias organizaciones, lo que presenta desafíos logísticos, así como un riesgo potencial antimonopolio. Además, es posible que

algunos miembros de las cadenas de suministro no cuenten con la tecnología o las infraestructuras necesarias para implementar las herramientas de blockchain, y aquellos que sí la tengan pueden decidir que los costos iniciales son un precio demasiado elevado para digitalizar sus datos y conectarse con otros miembros de la cadena de suministro.

- **Blockchain híbrida.** Las cadenas de bloques híbridas son cadenas de bloques que están controladas por una sola organización, pero con un nivel de supervisión realizado por la cadena de bloques pública, que es necesaria para realizar ciertas validaciones de transacciones. Un ejemplo de una cadena de bloques híbrida es IBM Food Trust, que se desarrolló para mejorar la eficiencia en toda la cadena de suministro de alimentos.

2.1.2 Redes de capa 1 y 2

Antes de entrar más en detalle, explicaremos la diferencia entre ambas capas. Las redes de capa-1 se refiere principalmente a una blockchain. Sin embargo, las redes de capa-2 son integraciones o protocolos que dan soporte a las soluciones de capa-1.

2.1.2.1 Redes de capa 1

La escalabilidad de blockchain es la expansión de una red en el espacio digital en términos de velocidades de procesamiento de transacciones y potencia de procesamiento, para adaptarse a la adición de nuevas aplicaciones y el aumento de las operaciones de los usuarios. Al escalar, las redes blockchain podrán competir con éxito con las redes centralizadas por los volúmenes de transacciones, la acumulación de aplicaciones y la participación del usuario al ofrecer mayores capacidades de procesamiento. Desde un punto de vista técnico, “escalado” se refiere a un aumento en

la tasa de rendimiento, medido en términos del número de transacciones por segundo **[6].**

Una de las soluciones de vanguardia para abordar el problema de la escalabilidad es la introducción de soluciones de Capa 1. Una cadena de bloques de Capa 1 es un conjunto de soluciones que mejoran el protocolo base en sí para hacer que el sistema en general sea mucho más escalable. Los dos enfoques propuestos para implementar soluciones de Capa 1 incluyen el protocolo de consenso y la fragmentación (**Sharding**).

Protocolos de Consenso

Prueba de trabajo, o PoW, es el mecanismo de consenso tradicional para Bitcoin y ETH. Su objetivo es lograr tanto el consenso como la seguridad utilizando mineros para decodificar algoritmos criptográficos complejos. Sin embargo, PoW se enfrenta a dos problemas principales: llegar a un consenso entre todos los participantes es un proceso lento, y adicionalmente se consumen muchos recursos para alcanzar ese consenso – lo cual incurre en los gastos energéticos altos de los cuales los PoW son sobradamente conocidos.

Proof-of-Stake, o PoS, es un mecanismo que presenta un consenso distribuido sobre la red blockchain. Los usuarios pueden autenticar transacciones en bloque en función de su participación. PoS gana sobre PoW no sólo en términos de velocidad de transacción, sino también en una reducción del gasto energético – puesto que no es necesario resolver algoritmos criptográficos complejos para alcanzar el consenso. Por otra parte, los PoS pierden en términos de seguridad, puesto que el funcionamiento de la red puede alterado por un grupo de entidades que tengan una gran cantidad de tokens en la red.

A día de hoy, la cadena de bloques Ethereum desea realizar la transición de PoW a PoS a través de Ethereum 2.0. Ethereum 2.0 es el término colectivo para un conjunto de actualizaciones que están actualmente en curso para hacer que la cadena de bloques Ethereum sea más escalable y sostenible.

Por otro lado, tenemos el protocolo de consenso Snowman, utilizado por la blockchain de Avalanche. Este protocolo se ha optimizado para contratos inteligentes y para un alto rendimiento, optimizando la interoperabilidad con contratos inteligentes en la red Ethereum. Snowman se basa en un mecanismo de consenso de prueba de participación (PoS) diseñado por Ava Labs que requiere que los usuarios hagan staking de AVAX para convertirse en validadores de transacciones. Para participar en el consenso, los validadores deben apostar al menos 2000 tokens AVAX, o tal vez, delegar su AVAX a un validador de su elección.

Por último, entre las blockchains privadas, el algoritmo de consenso más importante es Proof of Authority (PoA). Se caracteriza por ofrecer la identidad real de una persona para permitir la validación dentro de la cadena. Por tanto, dichos validadores ponen su identidad y reputación a cambio de ofrecer una mayor transparencia, y por tanto cualquier acto malintencionado recae directamente sobre esa persona.

Fragmentación

Sharding es otro método que ha sido portado desde el sector de bases de datos distribuidas y adaptado para soluciones de Capa 1. La fragmentación es un enfoque experimental en el espacio de la cadena de bloques, ya que implica la división de una red en una serie de bloques de base de datos separados conocidos como “fragmentos”, de ahí el término “Sharding”, que esencialmente hace que la cadena de bloques sea más manejable. Este enfoque también facilita los requisitos actuales para que todos los

nodos procesen o manejen transacciones para mantener la red, ya que todos los "fragmentos" se procesan en una secuencia paralela, lo que permite liberar mayores capacidades de procesamiento para otros procesos.

2.1.2.2 Redes de capa 2

La capa 2 se refiere a un marco o protocolo secundario que se construye sobre un sistema blockchain existente. El objetivo principal de estos protocolos es resolver las dificultades de escalado y velocidad de las transacciones a las que se enfrentan las principales redes de criptomonedas **[8].**

Por ejemplo, Bitcoin y Ethereum todavía no pueden procesar miles de transacciones por segundo (TPS), y esto ciertamente es perjudicial para su crecimiento a largo plazo. Es necesario un mayor rendimiento antes de que estas redes se puedan adoptar y utilizar de forma eficaz a una escala más amplia.

En este contexto, el término "capa 2" se refiere a las múltiples soluciones que se proponen al problema de escalabilidad de la cadena de bloques. Dos ejemplos importantes de soluciones de capa 2 son Bitcoin Lightning Network y Ethereum Plasma. A pesar de tener sus propios mecanismos de trabajo y particularidades, ambas soluciones se esfuerzan por proporcionar un mayor rendimiento a los sistemas blockchain.

En un sentido más amplio, los protocolos de capa 2 crean un marco secundario, donde las transacciones y procesos de blockchain pueden tener lugar independientemente de la capa 1 (cadena principal). Por esta razón, estas técnicas también pueden denominarse soluciones de escala "fuera de la cadena".

Una de las principales ventajas de usar soluciones fuera de la cadena es que la cadena principal no necesita pasar por ningún cambio estructural porque la segunda capa se agrega como una capa adicional. Como tal, las soluciones de capa 2 tienen el potencial de lograr un alto rendimiento sin sacrificar la seguridad de la red.

En otras palabras, una gran parte del trabajo que realizaría la cadena principal se puede mover a la segunda capa. Entonces, mientras que la cadena principal (capa 1) brinda seguridad, la segunda capa ofrece un alto rendimiento, pudiendo realizar cientos, o incluso miles, de transacciones por segundo **[9].**

Las diferentes soluciones Capa 2 son:

- **Canales de estado.** Permiten a los usuarios realizar transacciones varias veces en una cadena diferente (capa 2). Por el contrario, la cadena principal (capa 1) procesa solo dos transacciones, una cuando el canal está abierto y otra cuando el canal está cerrado. Al hacer esto, la cadena principal no procesa todas las transacciones, pero proporciona el mismo nivel de seguridad en la finalidad de la transacción. Una vez que las transacciones se completan y el canal ya no es necesario, los participantes envían sus copias del historial de transacciones para verificar sus copias de datos y asegurarse de que no haya discrepancias. Tras ello, se publica, la transacción final se carga en la cadena y el canal se cierra **[9].**

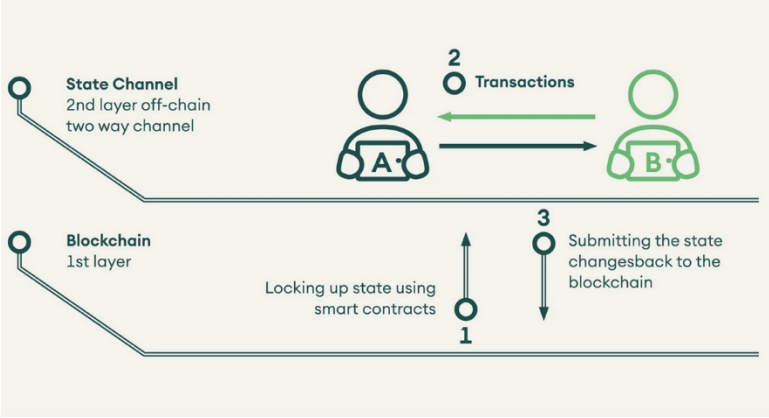


Figura 4. Canal estatal y los pasos para realizar una transacción.

- **Plasma.** El plasma consta de varias copias de la cadena principal que se ejecutan junto a él. Miles de transacciones se procesan en estas cadenas secundarias, se agrupan y se envían a la cadena principal como una sola transacción. Por definición, una child chain es una cadena sin confianza y sin custodia donde los usuarios controlan sus fondos. Por lo tanto, si hay errores o vulnerabilidades, pueden consultar la cadena de plasma más reciente y restaurar sus tokens **[10].**

  La ventaja de las capas de plasma es su alto rendimiento para procesar más de 1000 transacciones por segundo a una fracción del costo. Aquí, no es necesario tener un número fijo de entidades o individuos conocidos con los que realizar transacciones. Al igual que los canales de estado, las soluciones de plasma no son totalmente compatibles con los contratos inteligentes y solo son adecuadas para transacciones e intercambios. Los proyectos que utilizan capas de plasma incluyen Polygon (MATIC) y OMG Network (OMG).

- **Sidechains.** Las sidechains y las child chains (plasma) son similares excepto por un elemento: la seguridad. Si bien las cadenas de plasma dependen de la seguridad de su cadena principal en un entorno sin confianza y están optimizadas para un alto rendimiento y seguridad, las sidechains son cadenas de bloques independientes que se ejecutan en paralelo a la cadena principal y tienen sus propios mecanismos de consenso y algoritmos de seguridad.

  La ventaja de las sidechains es que generalmente son independientes de la cadena de bloques y pueden admitir múltiples capas base con cualquier cadena de bloques que quieran ejecutar. Estas sidechains pueden tener sus propios tokens,

pueden admitir contratos inteligentes y solo se comunican con la cadena principal cuando desean actualizar su estado. Estas sidechains pueden lograr hasta 10.000 transacciones por segundo dependiendo de su diseño. Sin embargo, esto tampoco viene sin sus desventajas. Los usuarios deben transferir la custodia de los fondos a la sidechain, y el mecanismo de seguridad puede ser más débil que la cadena principal. xDai (xDai) y Polygon (MATIC) son dos ejemplos de sidechains de Ethereum.

- **Rollups.** Agrupan miles de transacciones en un solo bloque de rollup, publicando solo los datos resumidos en la cadena principal. Potencialmente, puede proporcionar un aumento de 100 veces su rendimiento, ya que todos los cálculos y el almacenamiento ocurren fuera de la cadena principal. Al agrupar las transacciones y mover el procesamiento fuera de la cadena, los paquetes acumulativos reducen significativamente las tarifas de transacción y el tiempo de procesamiento. Existen dos tipos de rollups:
    - **Optimistic Rollups.** Utilizan una sidechain para procesar un lote de transacciones en paralelo a la cadena principal, resumirlo y certificar las transacciones en la parte superior de la red principal. Trabajan con la suposición básica de que todas las transacciones enviadas a la cadena principal son válidas. Solo cuando un usuario cuestiona un resumen, todo el bloque se calcula en la capa base. Como resultado, los fondos se bloquean durante un tiempo, generalmente una semana, antes de liberarlos en la capa base.
    - **ZK Rollups (Zero-Knowledger Rollups).** Ejecutan todos los cálculos fuera de la cadena y envían una prueba de validez en Ethereum. Se diferencian de las “optimistic rollups” porque no hay suposición de confianza, ya que la prueba de validez se imprime directamente en la

cadena. Si bien los “optimistic rollups” requieren evidencia de fraude, los “ZK Rollups” tienen pruebas de validez para cada transacción.

Como no existe un supuesto de confianza, no hay demora en el traslado de fondos de la capa 1 a la capa 2 y viceversa. Actualmente, no existe una solución generalizada basada en “ZK Rollup” compatible con EVM (Ethereum Virtual Machine), y solo están disponibles soluciones específicas para transferencias o intercambios.

	State Channels	Plasma	Sidechains	Optimistic Rollups	ZK Rollups
Full smart contract support	✗	✗	✓	✓	✗
Trustless	✓	✓	✗	✓	✓
Instant withdrawal	✓	✗	✓	✗	✓

Figura 5. Soluciones de capa 2.

2.1.3 Casos de uso

2.1.3.1 Sector sanitario

La industria médica ha sufrido mucho por la incapacidad de compartir y acceder de forma segura a los datos confidenciales de los pacientes. Las redes Blockchain permiten a las organizaciones establecer transparencia y privacidad por igual, lo que ayuda a ocultar los datos confidenciales de los pacientes y al mismo tiempo permite el acceso y el intercambio cuando sea necesario. Protegidos por soluciones de seguridad de última generación, los atacantes requerirían grandes capacidades computacionales para incluso intentar apuntar a una red impulsada por blockchain, limitando severamente la frecuencia, posibilidad y efectividad de los ataques. A su vez, esto permitirá que los sistemas de información sanitaria trabajen juntos dentro y fuera de los límites de la

organización a fin de promover la prestación eficaz de atención médica para las personas y las comunidades **[11].**

2.1.3.2 Moneda digital

Es el caso de uso más sonado actualmente y podría resolver grandes problemas dependiendo de su implementación. La más conocida y con mayor capitalización de mercado es Bitcoin (BTC), tras ella, le sigue de cerca Ethereum (ETH), que además de ser parecida a Bitcoin (BTC), incluye Smart Contracts y DApps, aplicaciones descentralizadas ejecutadas en la Blockchain, en el próximo apartado veremos sus utilidades. Por último, el uso de monedas digitales nos proporcionaría una manera rápida, sin costes (a excepción del coste computacional) y sin intermediarios (no existe servicio bancario) de enviar dinero.

2.1.3.3 Industria 4.0

La industria 4.0 implica innovaciones con las próximas tecnologías digitales, y blockchain es una de ellas. Se puede incorporar la tecnología blockchain para mejorar la seguridad, la privacidad y la transparencia de los datos tanto para pequeñas como para grandes empresas. La industria 4.0 es una síntesis de los nuevos métodos de producción que permiten a los fabricantes alcanzar su objetivo de manera más rápida. Se han realizado investigaciones sobre diversas tecnologías de la Industria 4.0, como Inteligencia Artificial (AI), Internet de las cosas (IoT), Big data y Blockchain, y cómo podrían crear interrupciones significativas en los últimos años. Estas tecnologías brindan diversas posibilidades en el mundo de la fabricación y la cadena de suministro. Blockchain es una tecnología que ha ganado mucho reconocimiento y puede mejorar el entorno de fabricación y cadena de suministro. Actualmente se están analizando varios impulsores, habilitadores y capacidades asociadas de la tecnología Blockchain para la Industria 4.0 para obtener información **[12].**

Finalmente, esta tecnología funcionaría para lograr mejores resultados y funcionaría individualmente para mejorar el proceso. A modo de ejemplo, un fabricante de automóviles que lanza un vehículo que contiene piezas defectuosas, generaría retiradas y reparaciones costosas, por lo que podría usar la cadena de bloques para rastrear al proveedor de las piezas defectuosas de manera más eficiente, conteniendo el problema y reduciendo el tiempo y los costos laborales.

2.1.3.4 Cadenas de suministro

La tecnología Blockchain puede permitir un seguimiento de extremo a extremo más transparente y preciso en la cadena de suministro: las organizaciones pueden digitalizar sus activos físicos y crear un registro inmutable descentralizado de todas las transacciones, lo que hace posible rastrear los activos desde la producción hasta la entrega o el uso por parte del usuario final. Esta mayor transparencia en la cadena de suministro brinda más visibilidad tanto a las empresas como a los consumidores.

Además, Blockchain puede impulsar una mayor transparencia en la cadena de suministro para ayudar a reducir el fraude de bienes de alto valor, como diamantes y medicamentos. También podría ayudar a reducir las pérdidas de ganancias del comercio falsificado y del mercado negro, así como aumentar la confianza en los usuarios del mercado final al reducir o eliminar el impacto de los productos falsificados. Por otro lado, las empresas pueden mantener un mayor control sobre la fabricación. Se puede dedicar menos tiempo a la validación de datos y se puede dedicar más tiempo a la entrega de bienes y servicios, ya sea mejorando la calidad, reduciendo los costos o ambos **[13].**

Finalmente, la tecnología blockchain puede agilizar los procesos administrativos y reducir los costos al permitir una auditoría efectiva de los datos de la cadena de suministro. Los procesos que implican verificaciones manuales para propósitos de cumplimiento que actualmente pueden llevar semanas pueden acelerarse a través de un libro mayor distribuido de toda la información relevante.

2.1.3.5 Identidad digital

Como ya hemos mencionado anteriormente, cuando un archivo se registra en un sistema blockchain, la autenticidad de la información es garantizada por los numerosos nodos que mantienen la red. Dicho de otra forma, un "grupo de alegaciones", de múltiples usuarios, respalda la validez de todos los datos registrados. En tal escenario, los nodos de la red pueden ser controlados por agencias autorizadas o instituciones gubernamentales responsables de verificar y validar los registros digitales. Básicamente, cada nodo puede "emitir un voto" respecto a la autenticidad de los datos, para que así los archivos puedan ser usados como documentos oficiales, pero con unos niveles de seguridad más elevados.

2.2 Smart Contracts

Los contratos inteligentes fueron propuestos por primera vez en 1994 por Nick Szabo, un informático estadounidense que inventó una moneda virtual llamada "Bit Gold" en 1998. Szabo definió los contratos inteligentes como protocolos de transacciones computarizados que ejecutan los términos de un contrato. Quería extender la funcionalidad de los métodos de transacción electrónica, como POS (punto de venta), al ámbito digital **[14].**

Los contratos inteligentes han avanzado bastante con el tiempo. Comenzaron como simples declaraciones if-then que un programador podía crear e implementar.

Afortunadamente, esos mismos desarrolladores están trabajando para resolver problemas de accesibilidad. Desde sus inicios, los desarrolladores lo han hecho para que se puedan realizar contratos inteligentes sin conocimientos de codificación. Están aumentando la seguridad con diferentes lenguajes de programación, creando alternativas como contratos secretos y diseñando formas de almacenar automáticamente el historial de contratos inteligentes en un formato legible por humanos, mucho más fácil que usar la cadena de bloques.

Por tanto, un contrato inteligente (Smart Contract) se podría definir como un contrato autoejecutable en el que los términos del acuerdo entre el comprador y el vendedor se escriben directamente en líneas de código. El código y los acuerdos contenidos en el mismo existen a través de una red blockchain descentralizada y distribuida. El código controla la ejecución y las transacciones son rastreables e irreversibles. Los contratos inteligentes permiten que se lleven a cabo transacciones y acuerdos confiables entre partes anónimas y desconocidas sin la necesidad de una autoridad central, un sistema legal o un mecanismo de cumplimiento externo **[15].**

Pero entonces, ¿qué es un contrato ejecutado? Simplemente es un contrato firmado que establece una conexión contractual entre dos o más partes. Cada parte se compromete a cumplir con los deberes legales que acordaron en el acuerdo escrito una vez que el contrato esté debidamente firmado. Popularizados por la segunda cadena de bloques más popular del mundo, Ethereum (ETH), los contratos inteligentes han llevado a la variedad de aplicaciones descentralizadas (DApps) de la red y otros casos de uso.

Como ya hemos mencionado anteriormente, un beneficio clave de las redes blockchain es la automatización de tareas que tradicionalmente requieren un intermediario externo. Por ejemplo, en lugar de necesitar que un banco apruebe una transferencia de

fondos del cliente al autónomo, el proceso puede ocurrir automáticamente, gracias a un contrato inteligente. Otro ejemplo podría ser un grupo legislativo y los ciudadanos a los que representa debatiendo una ley. Si estas dos partes llegan a un acuerdo en un sistema basado en blockchain, la ley se implementaría mediante un acuerdo ejecutado. Tal vez los usuarios puedan leer sobre la nueva ley a través de una DApp legal o interactuar con ella de otra manera basada en blockchain.

Por tanto, cualquier persona puede crear e implementar un contrato inteligente en una cadena de bloques.

2.2.1 Funcionamiento

Para empezar, explicaremos un pequeño caso de uso. Supongamos que un supermercado le pide a un agricultor 100 mazorcas de maíz. El primero bloqueará los fondos en un contrato inteligente, que luego se puede aprobar cuando el segundo entre la mercancía. Cuando el agricultor cumple con su obligación, los fondos se liberarán inmediatamente (es decir, después del cumplimiento de un contrato legal). Sin embargo, el contrato se cancela y los fondos se devuelven al cliente si el agricultor no cumple con su fecha límite.

Pero no solo eso, los contratos inteligentes se pueden programar para que funcionen para grandes masas de personas, reemplazando los mandatos gubernamentales y los sistemas minoristas, entre otros. Además, los contratos inteligentes eliminarían potencialmente la necesidad de llevar ciertos desacuerdos a los tribunales, ahorrando tiempo y dinero a las partes.
Esta seguridad se debe en gran medida al código del contrato inteligente subyacente. En Ethereum, por ejemplo, los contratos están escritos en su lenguaje de programación Solidity, que es Turing-completo. Esto significa que las reglas y limitaciones de los

contratos inteligentes están integradas en el código de la red y ningún mal actor puede manipular tales reglas. Además, estas limitaciones mitigarían las estafas o las alteraciones ocultas de los contratos. Los contratos inteligentes criptográficos solo pueden realizarse si todos los participantes están de acuerdo y firman al respecto. Una vez se dé esa situación, está configurado de por vida **[16].**

En términos más técnicos, la idea de un contrato inteligente se puede dividir en unos pocos pasos. Primero, un contrato inteligente necesita un acuerdo entre dos o más partes. Una vez establecido, los dos pueden acordar las condiciones en las que el contrato inteligente se considerará completo. La decisión se escribiría en el contrato inteligente, que luego se encripta y se almacena en la red blockchain. Una vez que se completa el contrato, la transacción se registra en la cadena de bloques como lo haría cualquier otra. Luego, todos los nodos actualizarán su copia de la cadena de bloques con esta transacción, actualizando el nuevo "estado" de la red.

Ahora, es posible que nos preguntemos si Bitcoin (BTC) y el resto de las redes blockchain pueden utilizar contratos inteligentes. Hasta cierto punto, sí. Cada transacción de BTC es técnicamente una versión simplificada de un contrato inteligente, y se han desarrollado soluciones de capa dos para expandir la funcionalidad de la red. Dicho esto, el uso de contratos inteligentes por parte de Ethereum es un caso especial.

2.2.2 Ethereum Contracts

A diferencia de la mayoría de las redes blockchain que se describen como un libro mayor distribuido, Ethereum es lo que se considera una máquina de estado distribuida, que contiene lo que se conoce como la Máquina Virtual Ethereum (EVM). Este estado de la máquina, del cual todos los nodos de Ethereum acuerdan mantener una copia, almacena el código del contrato inteligente y las reglas que deben cumplir estos

contratos. Dado que cada nodo tiene las reglas integradas a través del código, todos los contratos inteligentes de Ethereum tienen las mismas limitaciones.

Para representar esto en un modelo matemático, podemos tomar Y como una función de transición de estado. La función se puede escribir como Y(S, T) = S', donde:

- S es el antiguo estado válido.
- T es el conjunto de nuevas transiciones válidas que se agregarán al siguiente bloque.
- S' es el nuevo estado válido.

Esta función toma el antiguo estado válido y un conjunto de nuevas transacciones válidas para producir un nuevo estado válido como salida **[17].**

EVM crea un entorno de espacio aislado que ejecuta el código de bytes de los contratos inteligentes. Esto significa que el código de la máquina está completamente aislado de la red, el sistema de archivos o el proceso en la máquina host. Para cada instrucción implementada en el EVM, un sistema realiza un seguimiento del costo de ejecución. Por tanto, el usuario que desee ejecutar esta instrucción debe reservar algo de éter para pagar como tarifa de gas.

Además, tenemos que entender como se implementan los contratos inteligentes. Primero se compilan y se convierten en código de bytes cuando se implementa el contrato. Este código de bytes se almacena en la cadena de bloques y se le asigna una dirección. La dirección del contrato se determina en función de la dirección de la persona que crea el contrato (remitente) y la cantidad de transacciones que el creador ha enviado. La dirección del remitente y el nonce están codificados con RLP y con hash con el algoritmo keccak-256.

Los contratos inteligentes se implementan y prueban principalmente con Ethereum Remix IDE. Remix IDE es un IDE (Integrated Development Environment) basado en navegador de código abierto para contratos inteligentes de Ethereum.

2.2.3 Otras blockchains que integran Smart Contracts

Además de la blockchain de Ethereum, existen numerosos proyectos de blockchain que también integran contratos inteligentes: [**18**]

Solana (SOL)

Es la blockchain más rápida en creación de bloques en este momento, con velocidades de 50.000 transacciones por segundo (TPS). Para poner eso en perspectiva, Ethereum funciona de 15 a 45 TPS, aunque se acelerará dramáticamente después de "Eth2". Las tarifas promedio en Solana son una fracción de céntimo.

Utiliza una técnica llamada "prueba de historial" para procesar transacciones más rápidamente. Sin ser demasiado técnico, utiliza la incorporación de marcas de tiempo en sus registros de transacciones, lo que significa que no desperdicia poder de cómputo al verificar transacciones que ya se han procesado.

Solana tiene alrededor de 400 proyectos ejecutándose en su sistema, incluida la moneda estable de rápido crecimiento USDC. USDC se ejecuta tanto en Ethereum como en Solana.

Polkadot (DOT)

Si bien hemos comentado que Solana destaca por su velocidad, Polkadot destaca por su interoperabilidad (lo bien que funciona con otras plataformas).

Polkadot usa los llamados parachains. Estos se ejecutan en paralelo a la cadena de bloques principal y le permiten procesar transacciones más rápido. Los contratos inteligentes se ejecutan en esas parachains, no en la cadena de bloques principal.

ERGO (ERG)

Esta plataforma de contratos inteligentes no cobra tarifas de gas, lo que la distingue de las otras criptomonedas en esta lista. Ergo está diseñado para procesar contratos más complejos, lo que podría atraer a la industria DeFi (finanzas descentralizadas). Sin embargo, aún no figura en muchos de los principales intercambios de cifrado y tendrá que trabajar duro para construir su perfil.

Algorand (ALGO)

Al igual que las otras nuevas plataformas de contratos inteligentes, Algorand promete bajos costos, escalabilidad y velocidad, sin comprometer la seguridad. El hombre detrás del proyecto es el profesor Silvio Micali, quien, entre otras cosas, priorizó hacer accesible el lenguaje de los contratos inteligentes.

Los desarrolladores pueden usar diferentes lenguajes de programación para escribir contratos inteligentes en Algorand. Uno de ellos, Clarity, está diseñado para facilitar que los usuarios entiendan lo que hará el contrato, incluso si no son desarrolladores experimentados.

Cardano (ADA)

Aunque solo lanzó su funcionalidad de contratos inteligentes en septiembre, vale la pena comentarla porque la noticia de la tan esperada actualización llevó a Cardano a las tres principales criptomonedas por capitalización de mercado.

Cardano adopta un enfoque lento y constante para el desarrollo. Cada paso es revisado por pares y probado cuidadosamente, lo que significa que ha llevado mucho tiempo introducir la funcionalidad que otros han estado ejecutando durante años. Sin embargo, la expectativa es que se ponga al día rápidamente.

2.2.4 Ventajas

Como hemos explicado en la introducción a la blockchain, las cadenas de bloques brindan varios beneficios, que incluyen velocidad, eficiencia, precisión, confianza, transparencia, seguridad y ahorro en costes.

En primer lugar, los acuerdos automatizados disminuyen la posibilidad de manipulación de terceros al eliminar el requisito de que los corredores u otros intermediarios que ratifiquen los contratos legales ya firmados.

Además, la falta de un intermediario en los contratos inteligentes ahorra dinero. También, todas las partes relevantes tienen total visibilidad y acceso a los términos y condiciones de estos contratos. Por lo tanto, no hay forma de echarse atrás una vez que se firma el contrato. Esto asegura que la transacción sea completamente transparente para todas las partes involucradas.

Por último, todos los documentos guardados en la cadena de bloques se duplican muchas veces, lo que permite la restauración de los originales en caso de pérdida de datos. Los contratos inteligentes están encriptados y la criptografía protege todos los documentos para que no sean manipulados.

2.2.5 Desventajas

Sin embargo, los contratos inteligentes tienen desventajas que actualmente están siendo abordadas por especialistas en cadenas de bloques distribuidas y tecnologías de registro. La principal desventaja sigue siendo el riesgo a fallos inherentes de cualquier programa.

La mayoría del código de contrato inteligente es de código abierto, pero si está mal diseñado, puede permitir que los piratas informáticos exploten las fallas que contiene en detrimento de otros usuarios. El ejemplo más conocido de piratería es The DAO, que resultó en una pérdida de más de 150 millones de dólares (en éteres).

Este tipo de problemas plantea la cuestión de la inmutabilidad de las cadenas de bloques ("Code is Law") ante la necesidad de reintroducir la intervención humana en el marco de un contrato inteligente para restaurar la moral y el orden público. Existe una doctrina que argumenta que sería concebible agregar una forma adicional de gobernanza a las cadenas de bloques orquestadas por un conjunto inteligente de contratos inteligentes para definir posibles casos de arbitraje en caso de emergencia. Por otro lado, los puristas de Bitcoin y sus evoluciones están a favor de preservar en la medida de lo posible estos ecosistemas de la intervención humana y política. Este es un problema ideológico, que aún divide a la comunidad.

3

Análisis de las distintas Blockchains

En este apartado, se analizarán las distintas blockchains propuestas en la sección anterior. En base a los requisitos necesarios para un buen desarrollo y uso del sistema planteado en la introducción, se seleccionará(n) cual(es) de ellas es la que más se ajusta(n) a esos requisitos para desarrollar la prueba de concepto.

3.1 Requisitos necesarios para desarrollar el proyecto

Los siguientes requisitos han sido seleccionados en base a las necesidades del proyecto y siguiendo coherencia. Dadas las características de esta tecnología descritas en la sección 2 obtenemos los siguientes requisitos:

R1 - Escalabilidad

Necesitamos que la blockchain elegida sea escalable a medio y largo plazo. Aún no sabemos que carga tendrá nuestra red en un futuro, pero si necesitamos asegurarnos de que una blockchain pueda escalar para conseguir cubrir la demanda de nuestros clientes. En ese caso, necesitaremos incrementar la capacidad para gestionar un número de transacciones mayor. Por tanto, esta capacidad de crecer y acomodar la demanda será estudiada positivamente en los siguientes apartados.

R2 - Seguridad

Nuestro objetivo será asegurar que la información esté siempre en confidencialidad con el usuario (p.ej. Datos personales) y asegurar los fondos que disponga el cliente en su cartera virtual. Por tanto, se tendrá en cuenta los mecanismos de seguridad que utilizan las distintas blockchain estudiadas para mantener la integridad de los datos. En definitiva, este requisito tendrá en cuenta de forma individual la solución que ofrece cada blockchain analizada.

R3 - Descentralización

Uno de nuestros principales objetivos será la transparencia, por tanto, necesitamos que la blockchain escogida no esté en manos de unas cuantas personas. Si no que, sea algo público y verificable por cualquier persona. Por tanto, es imprescindible que detrás de una blockchain no esté una empresa u organismo gubernamental. En definitiva, necesitamos que no exista una figura con una autoridad mayor que el resto.

R4 - Capacidad de procesar grandes cantidades de transacciones

A priori, no sabemos la envergadura que tendrá nuestro proyecto, pero una blockchain debe estar preparada para procesar grandes cantidades de transacciones si en un momento puntual se necesitara. Hoy en día, en algunas blockchains se producen grandes cuellos de botella, entre las causas el algoritmo de consenso PoW (lento, dependiendo de qué transacción se mine). Por tanto, vamos a suponer que necesitamos procesar mínimo unas 500 TPS.

R5 - Costes por transacción

Este requisito es fundamental. Necesitamos estimar las tasas que tendremos que pagar por cada transacción que se haga en la blockchain, o bien puede ser un simple traspaso de criptomonedas o hasta la ejecución de un smart contract. Por tanto, necesitamos

tener un equilibrio entre los costes y el rendimiento. Desde el punto de vista económico, si conseguimos abaratar los costes por transacción, podremos ofrecer al cliente unos precios más competitivos sin necesidad de tener una base de datos centralizada y asumir los costes de esta.

3.2 Blockchains a estudiar

Como ya hemos comentado anteriormente, las cadenas de bloques que analizaremos serán Ethereum (ETH), Avalanche (AVAX), Polygon (MATIC), Binance Smart Chain (BSC) e Hyperledger Besu.

3.2.1 Ethereum (ETH)

En primer lugar, Ethereum es una red completamente descentralizada por su definición, por tanto, cumpliría el requisito 3. Actualmente, es capaz de procesar unas 10 o 15 transacciones por segundo.

En cuanto a la escalabilidad, actualmente se están investigando, probando e implementando múltiples soluciones que adoptan diferentes enfoques para lograr una mayor escalabilidad. En primer lugar, se necesita aumentar la capacidad de la red en términos de velocidad y rendimiento para la adopción masiva y significativa de Ethereum. Además, se comentó que el protocolo de consenso Proof of Stake (PoS) consigue una mayor escalabilidad y eficiencia, que precisamente es el protocolo usado por Ethereum.

En la capa 1 de Ethereum (Mainnet), la fragmentación (Sharding) es actualmente el enfoque principal para conseguir escalabilidad, tal y como explicamos en el apartado "2.1.2.1 Redes de capa 1". Sharding consistía en la división de la información en la base de datos para repartir la carga. En definitiva, esta solución lograría una mayor

descongestión de la red, lo que haría posible procesar más transacciones por segundo, sin embargo, al ofrecernos entre unas 10 o 15 TPS no lograría cumplir el requisito 4 (procesar gran cantidad de transacciones).

En la capa 2 de Ethereum, que también fue explicada en la sección "2.1.2.2 Redes de capa 2" se utilizaría Rollups, Canales de estado, Plasma y Sidechains (una de ellas precisamente es Polygon (MATIC). Algunas de estas soluciones podrían entrar en la próxima actualización de Ethereum (2.0) prevista para este año 2022.

Concluyendo, podríamos afirmar que actualmente la cadena de bloques no es escalable, pero se han propuesto soluciones a largo plazo, por lo que en el caso de que no se llevaran a cabo nuestro proyecto podría estar en peligro (requisito 1).

En cuanto a seguridad de los datos personales, entre las ideas fundamentales de criptomonedas se encuentra la seguridad. Ethereum es una blockchain no permisionada, y su software es *open-source*, consiguiendo así que tanto científicos y criptógrafos puedan analizar todos los aspectos de la red y su seguridad. Sin embargo, otro aspecto importante a destacar es que la seguridad de las aplicaciones desplegadas en la blockchain de Ethereum depende de los propios desarrolladores. Esto es debido a que el código podría contener *"bugs"*, lo que podría provocar incluso la pérdida de nuestros fondos monetarios **[19].**

De hecho, en 2016 se produjo el ataque más dañino que ha recibido Ethereum, donde se robaron unos 150 millones de dólares. El ataque fue lanzado por DAO (Decentralized Autonomous Organization), una organización que logró encontrar vulnerabilidades en el código base de Ethereum. Este ataque provocó un *"hard fork"* para restaurar los fondos robados, algo que supuso una gran polémica para la comunidad de Ethereum.

Por tanto, Ethereum no dispone de protocolos fuertes de seguridad (requisito 2), aunque ofrece consejos para los desarrolladores en su página web principal.

A fecha de redacción del presente trabajo de fin de grado, las tarifas de Ethereum son considerablemente más altas que las de la mayoría de las redes de contratos inteligentes. Hace siete días, las tarifas de ETH eran de 51,24 $ por transferencia y hoy (10 de diciembre de 2021), las métricas indican que cuesta 30,85 $ a 33,04 $ en gas por transferencia. Los datos de estas tarifas solo representan el movimiento de Ethereum (ETH), ya que cuesta más interactuar con un contrato inteligente para mover un ERC20 o intercambiar tokens **[20].**

Las tarifas de transacción promedio en la red Ethereum son algunas de las tarifas más altas pagadas a los mineros. Las tarifas de transacción medianas registradas es de alrededor de 6.82 $ por transferencia. Por tanto, podemos concluir que Ethereum no cumple el requisito 5 (bajos costes por transacción).

Máquina Virtual de Ethereum (EVM)

Es el entorno de ejecución para los Smart Contracts de Ethereum. Consigue ejecutar el código exactamente como se pretende en una pila de registro de 256 bits. Se compila en el bytecode EVM, lenguaje por defecto de la máquina. Esta máquina tiene la finalidad de ejecutar código no confiable por lo que está orientada a la seguridad y bajo una serie de restricciones:

- Cada operación ejecutada en ella debe pagar previamente una tasa, con ello, se trata de evitar un ataque de denegación de servicio (DNS), puesto que necesitaríamos Ether infinitos para realizarlo.
- No se pueden acceder a los estados de otros programas, únicamente pueden interactuar los datos de un array de longitud aleatoria.

- Un programa EVM solo puede acceder y modificar su propio estado interno, y activar la ejecución de otros programas dentro del entorno.

Ether

Ethereum tiene su propia criptomoneda llamada Ether (ETH). Es la criptomoneda oficial utilizada por los clientes de la blockchain de Ethereum. Se podría decir que es una criptomoneda similar a Bitcoin para poder realizar pagos a otras personas, pero también es usada en el desarrollo de los Smart Contracts. Además, es el reclamo que asegura que los programadores desarrollen aplicaciones con una calidad óptima y a su vez, reciban los mineros una recompensa por incluir bloques en la cadena.

Gas

En la EVM, podría ejecutarse el siguiente programa:

```
function foo(){
     while (true){
     . . .
     }
}
```

Figura 6. Bucle infinito.

Donde un while(true) es un bucle infinito, por tanto, esto podría provocar un ataque de denegación de servicio (DoS). Por ello, Ethereum desarrollo un mecanismo de defensa para protegerse de este ataque, esta solución fue llamada "gas". El concepto puede asemejarse al coste que tiene una transacción dentro de la red. Cada operación u opcode tiene asociado un coste de gas, por ejemplo: ADD (3 gas), hash SHA (30 gas), realizar una transacción de Ether (21.000 gas) … etcétera. A mayor dificultad computacional, mayor será el gas requerido.

La EVM necesita del parámetro STARTGAS (transaction gaslimit), que indica la cantidad de gas enviada en una transacción, aunque es un parámetro modificable por

el usuario. A parte, se ha de indicar una cantidad máxima de gas a pagar. Cuando se trata de pagar por el gas consumido, el pago se realiza en Ether, y se define en el campo "gasprice" (precio del gas) definido como Gwei/gas (1 Gwei es 10^9 Ether). El gas no solo está asociado al coste por transacción u operación, también se emplea como mecanismo para pagar a los mineros. Estos prefieren transacciones con un alto precio en gas, ya que obtendrán una mayor recompensa y tardarán menos tiempo en confirmarse la transacción. El gas, también está presente en los bloques de la blockchain, ya que existe un campo llamado block gaslimit, que representa el máximo gas combinado por todas las transacciones que permite un bloque aceptar.

3.2.2 Avalanche (AVAX)

Avalanche (AVAX) es una plataforma blockchain compatible con los contratos inteligentes cuyos principales objetivos son ofrecer a los usuarios la mayor velocidad de transacciones por segundo, bajos costes y ser una blockchain "*eco-friendly*".

Fue lanzada en 2020 por el equipo de Ava Labs, que introdujeron un modelo innovador en los ecosistemas blockchain. La principal innovación de Avalanche es que se compone de tres cadenas de bloques en lugar de una, como es habitual. La razón existente de esta elección de diseño se basa: cada cadena de bloques se especializa en una tarea dentro del ecosistema más amplio de Avalanche en lugar de tener una cadena que las haga todas. La distribución de esas tareas entre diferentes cadenas ayuda a mantener la agilidad de la plataforma, lo que le permite lograr la trinidad dorada de las características de blockchain: descentralización, seguridad y escalabilidad **[21].**

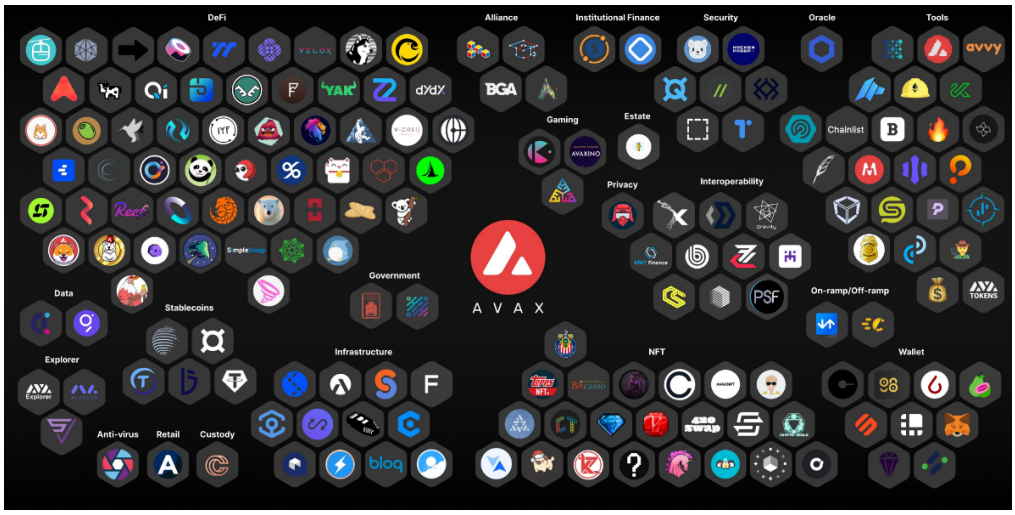


Figura 7. Ecosistema de Avalanche (AVAX)

En cuanto a la compatibilidad, Avalanche respalda la totalidad de las herramientas de desarrollo de Ethereum y permite a millones de validadores independientes participar como productores de bloques completos. En la red de prueba de Avalanche, más de 1.000 nodos completos fueron producidos, estableciendo así un nuevo récord global.
La blockchain de Avalanche es capaz de procesar órdenes de mayor magnitud que otras redes blockchain (más de 4.500 transacciones/segundo, cumpliendo así el requisito 4) y niveles de seguridad considerablemente más allá del 51% de otras redes (cumple el requisito 2) **[22].**

En cuanto al protocolo de consenso, existe una gran distinción entre Avalanche y otras redes, de hecho, muchos criptográficos llegaron a la conclusión errónea de que las cadenas de bloques deben ser lentas para ser escalables. Sin embargo, el protocolo de Avalanche utiliza un nuevo método para lograr consenso y proporcionar garantías de seguridad mientras logra una finalidad rápida y un alto rendimiento sin comprometer la descentralización. Además, Avalanche permite crear redes blockchain personalizadas con un conjunto de reglas y agregarlas a redes públicas o privadas existentes que se ajusten a los distintos casos de uso **[23].**

Si nos centramos en la interoperabilidad, que es una característica clave para las empresas, las cadenas de bloques interoperables de Avalanche son ilimitadas en número. Como ya hemos comentado al principio, debemos tener en cuenta que la blockchain de Avalanche está dividida en tres cadenas de bloques: X-Chain, C-Chain y P-Chain.

X-Chain (Exchange Chain) se utiliza para generar e intercambiar monedas AVAX (moneda principal de Avalanche) y otros activos digitales. Estos activos, como los estándares de token de Ethereum, contienen reglas cambiantes que controlan su comportamiento. AVAX se usa para pagar tarifas de transacción y la cadena de bloques emplea el sistema de consenso Avalanche.

Por otro lado, C-Chain (Contract Chain) usada para construir contratos inteligentes para las DApps. De hecho, la máquina virtual Ethereum (EVM) está implementada en esta cadena, permitiendo a los desarrolladores de Avalanche hacer un *"fork"* sobre DApps compatibles con EVM. Para ello, se emplea el protocolo de consenso *Snowman*, que es una versión modificada del mecanismo de consenso de la cadena de bloques de Avalanche. Por tanto, en comparación con las demás blockchains, Avalanche establece el consenso con un muestreo de nodo aleatorio. Este exclusivo sistema de consenso, junto con una variedad de técnicas criptográficas, garantiza que todos los actores de la red estén en sintonía. Además, muchas de las redes descentralizadas de hoy en día se basan en protocolos de consenso antiguos que pueden enfrentar problemas de escalabilidad. En muchas redes de cadenas de bloques, el consenso de bloque se logra votando desde los nodos, pero dichos sistemas de consenso pueden tener problemas para escalar a medida que aumenta la cantidad de nodos. Avalanche al evitar estos problemas, podremos afirmar que es escalable en comparación con las demás blockchains estudiadas (requisito 1). De hecho, se ha establecido como una de las más

escalable entre las blockchains actuales, a pesar de los muchos desafíos comúnmente asociados con las cadenas de bloques de capa 1 **[24].**

Por último, P-Chain (Platform Chain) se encarga de organizar validadores de la red, realizando un seguimiento de las subredes activas y permitiendo el establecimiento de nuevas subredes. Las subredes son grupos de validadores que brindan consenso para cadenas de bloques personalizadas. Solo una subred puede validar una cadena de bloques, pero cada subred puede validar varias cadenas de bloques. El protocolo de consenso *Snowman* también es utilizado por P-Chain. A fecha del presente trabajo existen 1215 nodos validadores en la red de Avalanche, por lo que no podemos demostrar que sea tan descentralizada como se pretendía serlo en su idea original. Por tanto, no cumpliría totalmente el requisito 3 **[25].**

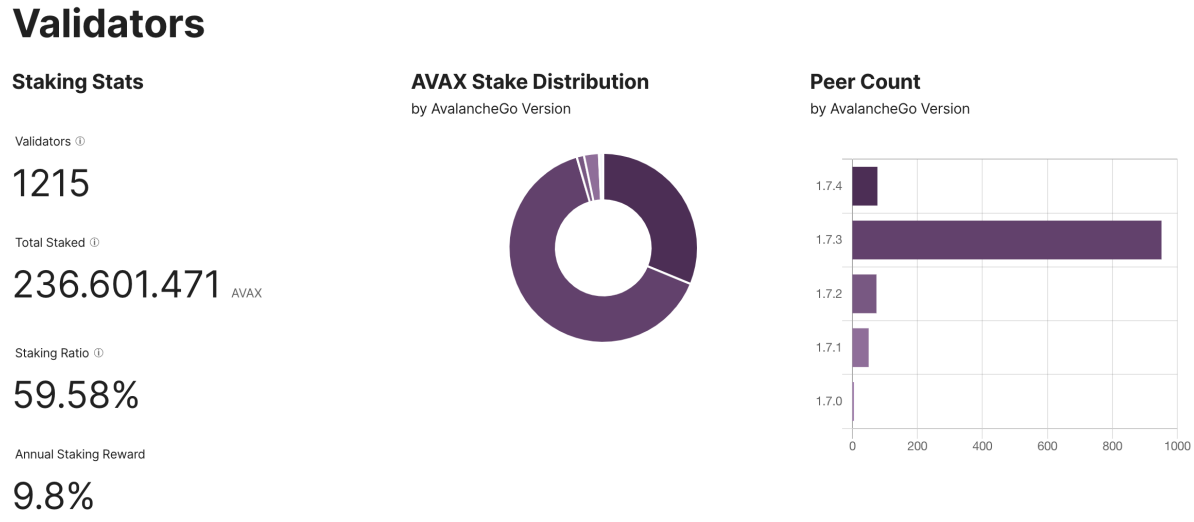


Figura 8. Validadores totales en Avalanche.

Respecto a las comisiones que tiene Avalanche para hacer una transacción, depende en qué cadena se realice la misma. En la C-Chain, es un algoritmo el que determina la tarifa base. Esta aumenta cuando la utilización de la red está sobrecargada y disminuye cuando ocurre la situación contraria. Las tarifas de transacción para transacciones no atómicas se basan en las transacciones de tarifas dinámicas de estilo EIP-1559 de

Ethereum, que consisten en un límite de tarifa de gas y un límite de punta de gas. Además, existe la tarifa de prioridad, que especifica la cantidad máxima por encima de la tarifa base que la transacción está dispuesta a pagar por unidad de gas. A diferencia de Ethereum, donde la tarifa de prioridad se paga al minero que produce el bloque, en Avalanche se liberan tanto la tarifa base como la tarifa de prioridad. Para las transacciones heredadas, que solo especifican un precio único del gas, el precio del gas sirve como tope de tarifa de gas. Por tanto, podemos concluir que en Avalanche, cada transacción cuesta de media menos que en sus competidores, como por ejemplo, Ethereum. De hecho, se estima que cada transacción en Avalanche cuesta aproximadamente 0,11 euros. Concluimos que las transacciones tienen unos costos bastantes reducidos, por lo que Avalanche cumple el requisito 5 **[26].**

3.2.3 Polygon (MATIC)

Polygon es un protocolo y un framework de capa 2 cuyo principal objetivo es construir y conectar redes blockchain compatibles con Ethereum, además fue diseñado para combatir el problema de escalabilidad que tenía Ethereum. Es una plataforma multinivel que consigue una gran escalabilidad gracias a sus sidechains (cadenas laterales). Por tanto, concluimos que cumple el requisito 1.

La cadena principal de Polygon es una blockchain que utiliza Proof of Stake (PoS) como método de consenso y utiliza el framework Plasma. Además, Polygon utiliza su propia criptomoneda, llamada MATIC, usada para pagar las tarifas de la red Polygon, hacer staking (forma en que se verifican sus transacciones y se obtienen recompensas) y para la gobernanza (los titulares de MATIC pueden votar sobre los cambios en Polygon). Como dato curioso, el nombre MATIC proviene de una etapa anterior en el desarrollo de Polygon. Después de su lanzamiento como Matic Network en octubre de 2017, los desarrolladores cambiaron su nombre a Polygon a principios de 2021 **[27].**

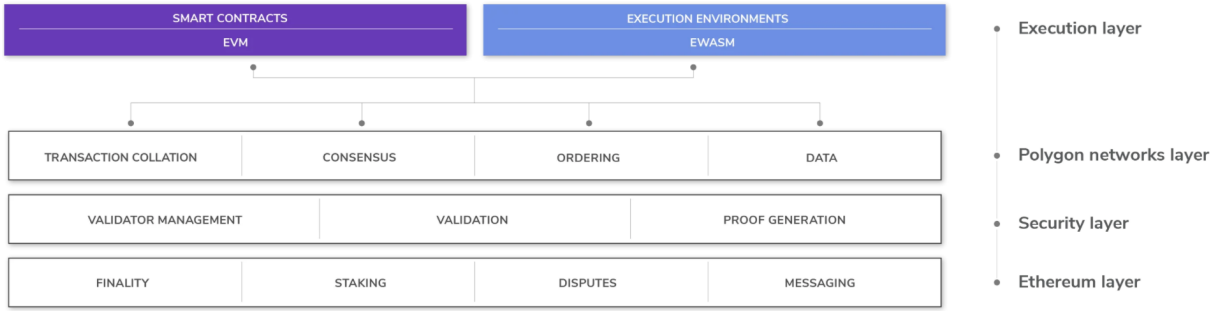


Figura 9. Arquitectura de Polygon.

En cuanto a la seguridad de Polygon, tiene una capa especializada para ello, tal y como podemos apreciar en la figura 9. Se trata de una capa no obligatoria que administra un conjunto de validadores que pueden verificar periódicamente la validez de cualquier cadena Polygon a cambio de una tarifa. La capa de seguridad es completamente abstracta y puede tener múltiples instancias, implementadas por diferentes entidades y con diferentes características. También se puede implementar directamente en Ethereum, en cuyo caso los mineros de Ethereum realizarían la validación. Por tanto, decimos que cumple con el requisito 2, es decir, es bastante más segura que las demás blockchains comparadas.

Ahora vamos a analizar si Polygon es poco o muy descentralizada:

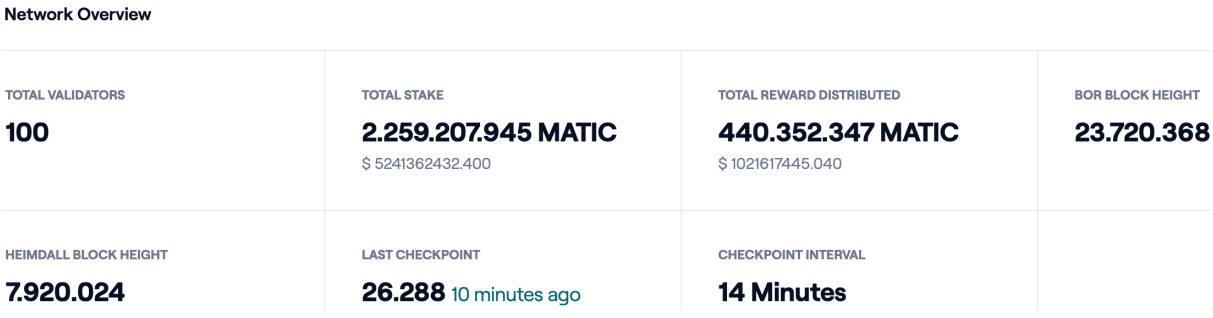


Figura 10. Resumen de la red Polygon.

Polygon se compone exactamente de 100 validadores, incluso menos que Avalanche (1215), por tanto, podríamos afirmar que Polygon es semidescentralizada. Además,

debemos tener en cuenta que no cualquiera puede convertirse en validador. Esto supondría poner en riesgo nuestro proyecto, ya que tenemos que confiar en esos 100 validadores para la gestión de nuestras reservas hoteleras. Concluyendo, Polygon no cumpliría el requisito 3 (descentralización) **[28].**

El tiempo de procesamiento de Polygon es de 2,3 segundos por transacción, recordemos que el propio Ethereum tarda unos 13 segundos por transacción. En teoría, Polygon puede realizar hasta 65.000 transacciones por segundo, pero la realidad es diferente. De media, Polygon realiza 7.000 transacciones por segundo, lo que nos proporcionaría una gran capacidad de realizar nuestras reservas hoteleras, cumpliendo así el requisito 4.

Cada vez el uso de Ethereum y en su consecuencia, Polygon, está provocando un aumento en las tasas por transacción (gas). Actualmente, la tarifa por transacción en Polygon oscila entre los 0,10 y 0,50 dólares. Algo que resulta ser más caro que Avalanche y más barato que Ethereum. Recordemos que Polygon al igual que Ethereum se rigen por el "Gas Price". Por tanto, ese coste si sería posible asumirlo dentro de nuestro proyecto, por lo que Polygon cumpliría el requisito 5 **[29].**

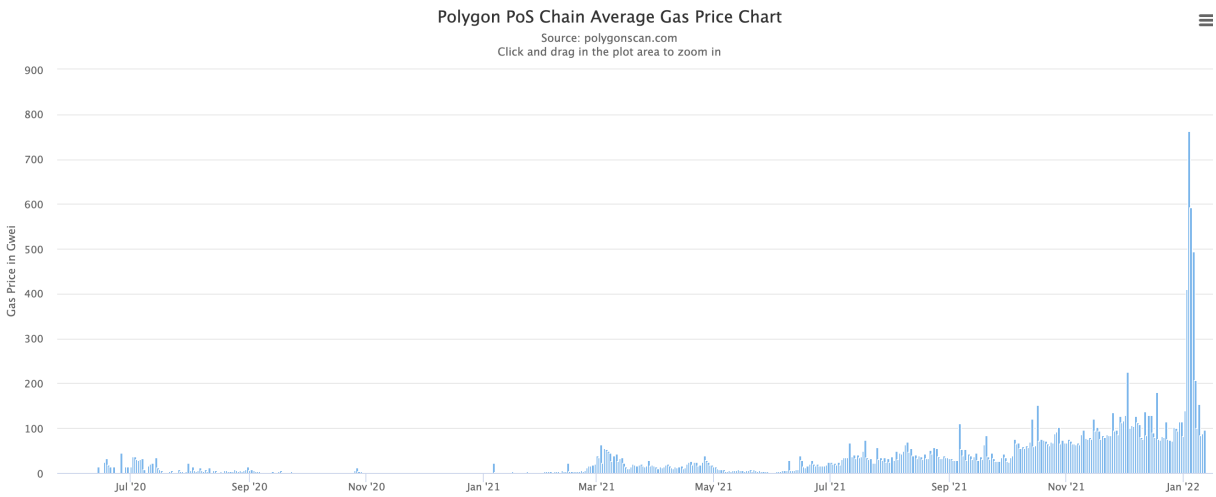


Figura 11. Precio medio del gas en unidades Gwei.

3.2.4 Binance Smart Chain (BSC)

Binance Smart Chain (BSC) es una cadena de bloques que se ejecuta en paralelo a la principal cadena de bloques Binance Chain (BC) de Binance (principal plataforma de intercambio de criptomonedas). BSC se lanzó en septiembre de 2020, aproximadamente un año y medio después del lanzamiento de su hermano mayor BC, con el propósito explícito de introducir la programabilidad de contratos inteligentes que no era compatible con BC. Por tanto, principalmente es una plataforma de contratos inteligentes que intenta imitar la funcionalidad de Ethereum y otras plataformas compatibles con contratos inteligentes. Su principal moneda es Binance Coin (BNB) creada en julio de 2017 como un token de utilidad en el intercambio de Binance. Inicialmente se utilizó como un incentivo para que la comunidad criptográfica participara en la ICO (oferta inicial de monedas) de Binance. Además, se realizaban descuentos en las operaciones si los titulares usaban la moneda como método de pago.

Por un lado, optaron por un mecanismo de consenso de Prueba de autoridad estacada (PoSA) en lugar de la Prueba de trabajo (PoW) actual de Ethereum. PoSA permite tiempos de bloque cortos y costos de transacción más bajos. En primer lugar, los validadores de BSC son participantes que "apuestan" una cierta cantidad de BNB. Cuando proponen un bloque válido, reciben tarifas de transacción como recompensa por las transacciones contenidas en ese bloque. Esto significa que no se acuñan nuevos BNB como recompensas para los validadores, sino que la recompensa proviene de tokens BNB ya existentes.

La principal razón por la que Binance adoptó el staking en lugar de la minería fue que priorizaron la velocidad de la red a expensas de la descentralización y la seguridad. La cadena de bloques de BSC está impulsada por 21 validadores comunitarios que construyen y verifican bloques en la cadena, y estos validadores son compensados en

monedas BNB por sus servicios. Mientras tanto, deben ser reelegidos diariamente por el gobierno de participación para permanecer en el conjunto de validadores. Los validadores que apuestan las mayores cantidades de BNB conforman la lista de 21 validadores, y esa lista se actualiza cada 24 horas. Hacer staking de BNB en BSC da como resultado recompensas basadas en las tarifas pagadas por transacciones en la cadena de bloques de BSC. Por tanto, podríamos afirmar que BSC es centralizada, ya que está apoyada por una empresa, en este caso Binance y es validada por tan solo 21 validadores. En consecuencia, el requisito 3 (descentralización) no se cumpliría si utilizáramos esta cadena de bloques **[30].**

Aunque BSC cuenta con una escalabilidad notable con tiempos de bloque de alrededor de tres segundos por bloque, al mismo tiempo requiere tarifas menores que los altos costos de gas incurridos en la cadena de bloques de Ethereum. De hecho, BSC fue diseñada para una adopción masiva y escalabilidad. Por tanto, el requisito 1 (escalabilidad) se cumpliría para nuestro sistema.

Ahora vamos a analiza si BSC es segura para nuestro proyecto o no. En primer lugar, partimos de la base de que el algoritmo de consenso PoSA construido a través de 21 validadores elegidos evita que los validadores individuales obtengan demasiado control sobre la red y se vuelvan deshonestos. Por tanto, la red BSC y el algoritmo con el que opera son realmente muy seguros. Hasta la presente fecha, el historial de BSC está limpio de incidentes y hackeos, algo que muestra que aún no hay vulnerabilidades conocidas o vectores de ataque que puedan ser objeto de abuso en la red. Para controlar la seguridad de la cadena de bloques, hay varias empresas de seguridad de BSC como “Peckshield” y “Certik” que auditan y verifican diferentes tokens y DApps de BSC. Las auditorías de seguridad buscan vulnerabilidades potenciales en el código, el modelo comercial y otros aspectos. También verifican a menudo a los miembros del equipo

central, revisan su experiencia previa o auditan las finanzas del proyecto. Por todas estas características consideramos que BSC es bastante segura, por lo que cumple el requisito 2 (seguridad).

Según podemos observar en BscScan (https://bscscan.com/), el tiempo promedio por bloque de Binance Smart Chain es de 3 segundos, y suponiendo que tuviéramos un uso completo de la red, BSC puede procesar alrededor de 160 TPS, pero si cogemos un día cualquiera tenemos que la media baja hasta las 67 TPS aproximadamente. Esto quiere decir, que actualmente, no supone ningún riesgo para el proyecto, pero a largo plazo puede ser un poco escaso. Concluyendo, BSC no cumpliría el requisito 4 (capacidad de procesar), ya que las anteriores blockchains estudiadas, superan esa barrera.

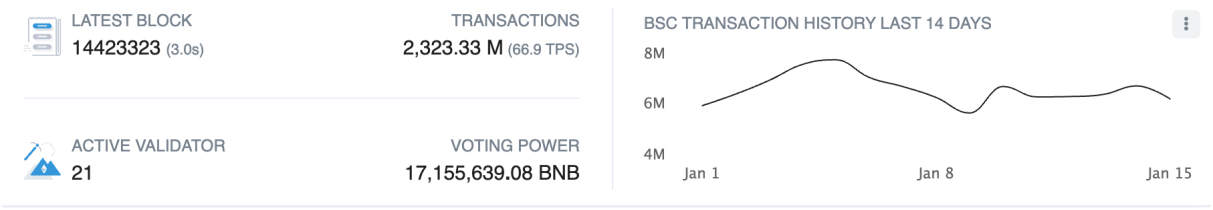


Figura 12. Estadísticas de BSC proporcionadas por BscScan. (16/01/2021)

Por último, necesitamos saber la tasa que pagaremos por cada transacción en BSC. Como se puede observar en la figura 12, la tarifa de gas ha ido descendiendo su precio hasta llegar a su mínimo, 5 Gwei. Si hacemos los cambios de unidades correspondientes tenemos que la tarifa de gas promedio de Binance Smart Chain por transacción es de 0,31 euros. Este precio correspondería a una tarifa base (no se incluyen las transacciones complejas). A pesar de que cumple con el requisito 5 (costes transacciones) ya que es una cifra asumible, está por encima de otras blockchains comparadas anteriormente **[31].**

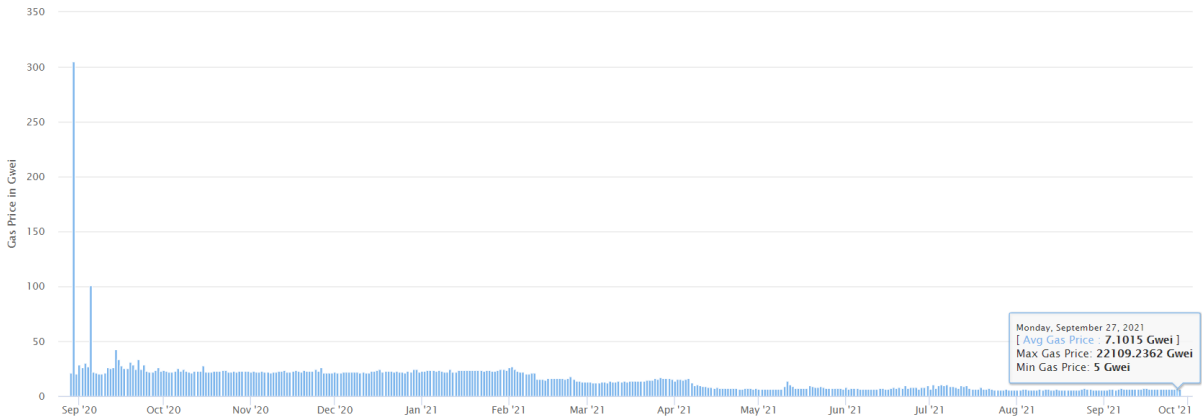


Figura 13. Tarifa promedio de gas de Binance Smart Chain. Fuente: BscScan.com

3.2.5 Hyperledger Besu

Hyperledger Besu se podría definir como un cliente de Ethereum de código abierto desarrollado bajo la licencia Apache 2.0 y escrito en Java. Se puede ejecutar en la red pública de Ethereum o en redes privadas autorizadas. Cuenta con redes de prueba como Rinkeby, Ropsten y Görli, donde puede ser ejecutada.

Besu está diseñado para ser lo más modular posible, de hecho, incluye varios algoritmos de consenso, incluidos PoW, PoA, e IBFT (Istanbul Byzantine Fault Tolerance). Entre las principales características de Hyperledger Besu se encuentran: la máquina de Ethereum, diferentes algoritmos de consenso, almacenamiento, redes P2P, API orientadas al usuario, supervisión, privacidad y gestión de permisos.

Besu incluye una interfaz de línea de comandos y una API JSON-RPC para ejecutar, mantener, depurar y monitorear nodos en una red Ethereum. La API es compatible con las funcionalidades típicas de Ethereum, como: minería de Ether, desarrollo de contratos inteligentes y desarrollo de aplicaciones descentralizadas. El cliente de Besu es compatible con el desarrollo, la implementación y los casos de uso operativos de contratos inteligentes y DApp comunes, utilizando herramientas como Truffle, Remix y web3j. Condición necesaria para el desarrollo del proyecto. De hecho, Besu

principalmente se orienta a empresas que requieran un procesamiento de transacciones seguro y un alto rendimiento, con un buen nivel de privacidad y permisionado **[32].**

En Hyperledger Besu, se distinguen dos tipos de nodos: *"full nodes"* y *"archives nodes".* Los "*full nodes"* (nodos completos) son usados para enviar y firmar transacciones, verificar el saldo actual y observar el estado actualizado de la red. Por el contrario, los *"archives nodes"* usan las mismas características que los nodos completos, además de almacenar el estado intermedio de cada cuenta desde el bloque génesis (primer bloque de cada blockchain). Es importante destacar que cada nodo contiene un par de clave público/privada y una dirección de nodo.

Como hemos comentado anteriormente, Besu implementa varios protocolos de consenso:

- **Ethash** (Proof of Work). Renombrado a Ethash, pero básicamente es el protocolo Proof of Work explicado anteriormente.
- **Clique** (Proof of Authority). Las transacciones y los bloques son creadas y validadas por turnos y son validadas por los "*signers*" (firmantes). Para añadir nuevos firmantes, es necesario que más del 50% de los firmantes existentes estén de acuerdo. Además, en este protocolo se pueden producir "*forks*" (bifurcaciones), esto quiere decir que no todos los nodos válidos son incluidos en la red principal.
- **IBFT** (Proof of Authority). Al ser un PoA, las cuentas validadas se conocen como "validators" y se alternan para validar transacciones y bloques. Se necesita un 66% de aprobación por parte de los "*validators*" para que un bloque se añada. Además, tienen que existir al menos 4 "*validators*" para que el protocolo funcione y no se admiten "*forks*", esto quiere decir que todos los nodos válidos son incluidos en la red principal.

Una de las características destacables de Besu es la gestión de transacciones privadas entre los participantes, sin que el resto de la red puedan ver esas transacciones. Para conseguir esa privacidad en las transacciones, Besu utiliza a nodos Orion para ello, cada nodo Besu debe tener un nodo Orion asociado. Primero, la transacción se genera en los nodos Besu, los cuales derivan la misma a los nodos Orion, que se encargarán de cifrar y de distribuir la transacción privada, punto a punto, entre el resto de los nodos Orion involucrados **[33].**

En la figura 14, podemos ver un ejemplo de la gestión de las transacciones privadas con nodos Orion. Según, se observa en la imagen, el procesamiento de una transacción privada implica en el fondo dos transacciones. Una es la transacción privada distribuida por los nodos Orion a los participantes involucrados, y la segunda, la transacción que se incluye en la cadena de bloques públicas.

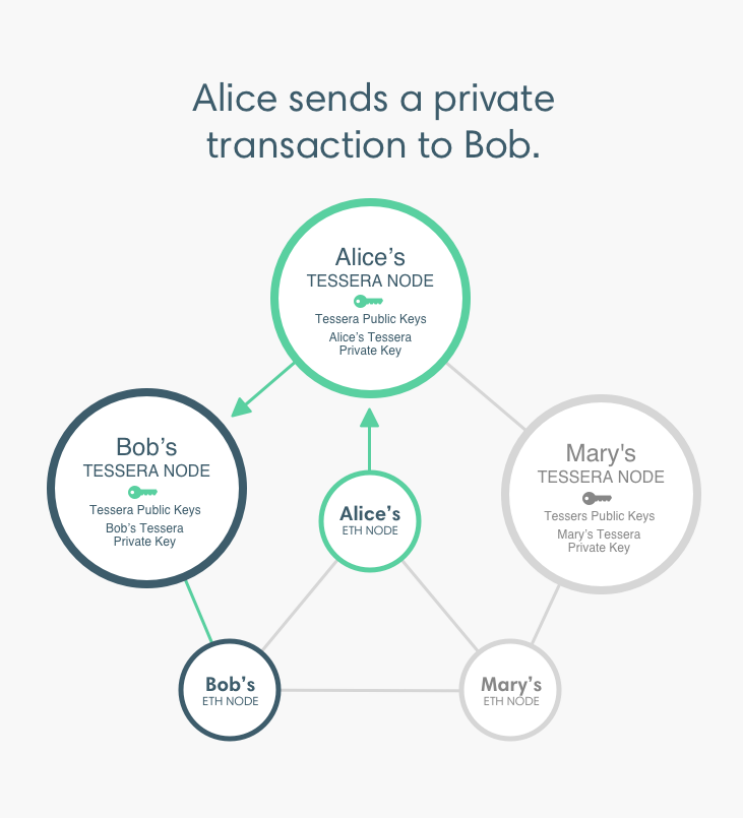


Figura 14. Gestión de transacciones privadas con nodos Orion.

Hyperledger Besu además de gestionar las transacciones privadas, también maneja los llamados “grupos de privacidad”. Los grupos de privacidad son un conjunto de nodos que son identificados por un único identificador Orion. Esto permite que la blockchain Besu mantenga el estado público mundial para la cadena de bloques y un estado privado para cada grupo de privacidad.

Otra característica interesante que presenta Hyperledger Besu es el permisionado a nivel de nodo, y también a nivel de cuenta. Por un lado, el permisionado a nivel de nodo solo permite el acceso a los participantes conocidos. Además, se pueden establecer permisos a “nivel local”, o a “nivel de cadena”, para ello, cada nodo de la red dispone de un archivo de configuración de permisos. Otra opción es programar un smart contract que coordine e indique nuevas reglas de permisionado para que los nodos se actualicen a medida que se propaguen por la red. Esto de cara a nuestro proyecto sería bastante interesante, ya que podríamos saber que nodos tenemos en la red interactuando en la blockchain y así controlar a nodos intrusos. Además, esta característica obligaría a los usuarios a registrarse de manera obligatoria. Pero esto tiene su aspecto negativo, y es que perderíamos esa transparencia que comentamos en el requisito 3 (descentralización) **[34].**

Sin embargo, al tener el control de quién puede acceder a nuestra red y quién no, obtendríamos una mayor seguridad (requisito 2) que el resto de blockchains estudiadas donde en la mayoría, podríamos acceder sin tener permisos especiales.

Respecto al requisito 1 (escalabilidad), Hyperledger Besu por definición, fue diseñada para aportar escalabilidad y seguridad a Ethereum, por lo que podríamos considerar que es mucho más escalable que el propio Ethereum.

Respecto a la capacidad de procesar transacciones (requisito 4), Besu 1.4 es capaz de procesar hasta 300 TPS y 400 TPS en Besu 1.5. Estos datos ofrecidos están sujetos a una red de cinco nodos donde cuatro eran validadores y un único nodo no era validador. Se usó IBFT como algoritmo de consenso con un tiempo de bloque de dos segundos. En definitiva, podemos estimar que no necesitamos más TPS de los que nos podría ofrecer Besu, por lo que cumple el requisito 4 **[35].**

Los costes por transacción dependen de que cómo sea la complejidad de esta, aún así al igual que Ethereum se rigen por el "Gas Price". Vamos a calcular algunos ejemplos de costes usando el entorno de ejecución Remix (https://remix.ethereum.org/):

- Una simple transacción: 21.000 Gwei = 0,06 €
- Implementación de un contrato inteligente de almacenamiento simple: 0,32 €

También contamos con la referencia de una DApp llamada, Pet Shop, proporcionada por Truffle (https://trufflesuite.com/), la cual ha sido lanzada en Besu usando un docker y smart Contracts. En resumen, el coste total de lanzar la aplicación ha sido de 0,0092294 Ether, o lo que es equivalente, 26,40 €. Este sería el coste inicial de lanzar nuestro sistema en Besu. A partir de ello, tendríamos que asumir las tarifas de gas por cada transacción o smart contract que se ejecute. Por tanto, determinamos que es un coste elevado (no cumple el requisito 5) al igual que Ethereum, ya que hemos estudiado blockchains más económicas que Besu **[36].**

3.3 Elección de la tecnología

Para empezar, observaremos una tabla-resumen que nos permitirá visualizar la información de manera más generalizada.

Blockchain/Requisito	**R1**	**R2**	**R3**	**R4**	**R5***	**RT**

Ethereum	1/5	2/5	5/5	1/5	2/5	11/25
Avalanche	4/5	3/5	3/5	4/5	5/5	19/25
Polygon	4/5	3/5	2/5	5/5	4/5	18/25
Binance Smart Chain	3/5	4/5	1/5	2/5	3/5	13/25
Hyperledger Besu	2/5	4/5	2/5	3/5	2/5	13/25

Tabla 1. Cumplimiento de los requisitos por cada Blockchain. (Escala 1 a 5).

R5*: A mayor puntuación menos costes tendría la red.

RT: Resultado total.

De la tabla 1, se han tenido en cuenta el estudio realizado en los apartados anterior del presente trabajo. La blockchain que menos se ajusta a nuestros requisitos es Ethereum, y si, sorprendentemente la segunda criptomoneda con mayor capitalización de mercado según CoinMarketCap (https://coinmarketcap.com/) es la última en cuanto a la exigencia de requisitos para nuestro sistema. A la espera de la actualización "Ethereum 2.0" vemos que no logra buenos resultados en escalabilidad. También necesita implementar mayores mecanismos de seguridad y capacidad de procesar grandes transacciones, si a eso le sumamos los grandes costes de gas que está teniendo en el último año, hace que sea una solución poco viable.

Un peldaño por encima nos quedaría Binance Smart Chain e Hyperledger Besu. Ambas son buenos proyectos, sin embargo, carecen de un requisito principal para nuestro proyecto que es la descentralización, ya que intentamos ser lo más transparentes posible en cuanto a transacciones entre clientes y hoteles. Además, tienen unos mayores costes transaccionales.

El resultado final ha sido muy igualado entre Avalanche y Polygon. Podríamos llevar a cabo nuestro proyecto en ambos proyectos sin ponerlo en riesgo. Sin embargo, la

mayor descentralización que nos aporta Avalanche y unos costes transaccionales menores, ha hecho que nos decantemos por **Avalanche**.

Pese a que desarrollaremos nuestro proyecto con la Blockchain de Avalanche, esto no quiere decir, que encaja perfectamente con nuestro proyecto, ya que podría ser aún más descentralizada e implementar más protocolos de seguridad. En definitiva, aún no existe una blockchain perfecta.

Figura 15. Logo de Avalanche, blockchain elegida para desarrollar el proyecto.

4

Prueba de concepto

En este apartado se desarrollará una explicación técnica del sistema que hace uso de la tecnología blockchain de Avalanche para funcionar. En esta prueba de concepto se incluye:

1. Desarrollo de un contrato inteligente en el que están desplegadas las reglas de negocio, el método de pago para confirmar las reservas y el almacenamiento de estas. Podríamos afirmar que el contrato inteligente será nuestro backend del sistema.
2. Desarrollo de una aplicación web que sea capaz de ofrecer al cliente todos los hoteles que colaboran con la web, para poder realizar una reserva y un posterior pago con la moneda principal de la blockchain de Avalanche (AVAX). La presente página web estará disponible para ordenadores y smartphones, sirviendo como interfaz de usuario.

4.1 Introducción

El principal objetivo de esta prueba de concepto consiste en desarrollar un sistema de reservas hoteleras con el uso de la tecnología blockchain Avalanche. Para poder realizar esta prueba de concepto, el contrato inteligente y las cuentas de Metamask presentes en esta prueba son simuladas. Eso quiere decir que, la red con la que interactuaremos será la red de pruebas de Avalanche (Avalanche Testnet Fuji). Por tanto, las cuentas utilizadas contendrán AVAX "falsos" (sin ningún valor económico) para poder realizar las pruebas correctamente.

Hay que tener en cuenta que en la blockchain se almacenan los datos de cada reserva y los hoteles asociados a esas reservas. Además, llevaremos la contabilidad de cada reserva hotelera, de la cual nos quedaremos un 5% de las ganancias. Desde la aplicación web, los clientes podrán interactuar con la blockchain, reservando sus estancias, así como cancelando las mismas. Para proceder a realizar una reserva, el usuario debe tener habilitada e instalada su cartera de Metamask con un saldo superior al requerido para hacer la reserva. La pestaña de reserva del hotel en cuestión se encargará de formalizar los datos y añadirlos al smart contract, tras ello la cartera de Metamask nos avisará de que debemos de pagar los costes de gas, así como el propio coste de la reserva.

Una vez se haya completado la transacción, deberá ser confirmada por los validadores de la blockchain. Posteriormente, ya tendríamos hecha y validada nuestra reserva. La propia aplicación web nos mostrará un hash de reserva, que deberemos guardar para el día de entrada en el hotel en cuestión. Este hash es el resultado de ejecutar una función criptográfica que contiene nuestro DNI o pasaporte junto con las noches de estancia, el día de entrada y el identificador del hotel.

Una vez llegue el día de entrada, el cliente mostrará su DNI y su número hash proporcionado. El personal del hotel mediante una vista también diseñada en el presente trabajo comprobará que los datos del cliente devuelven el hash que dice el cliente tener. En caso de que ambos hashes coincidan, la reserva sería correcta y el cliente podría hospedarse, sin embargo, si no coincide, quiere decir que la persona en cuestión no es la que reservó.

A modo de recopilación mostraremos un esquema de comunicación entre los distintos componentes del sistema.

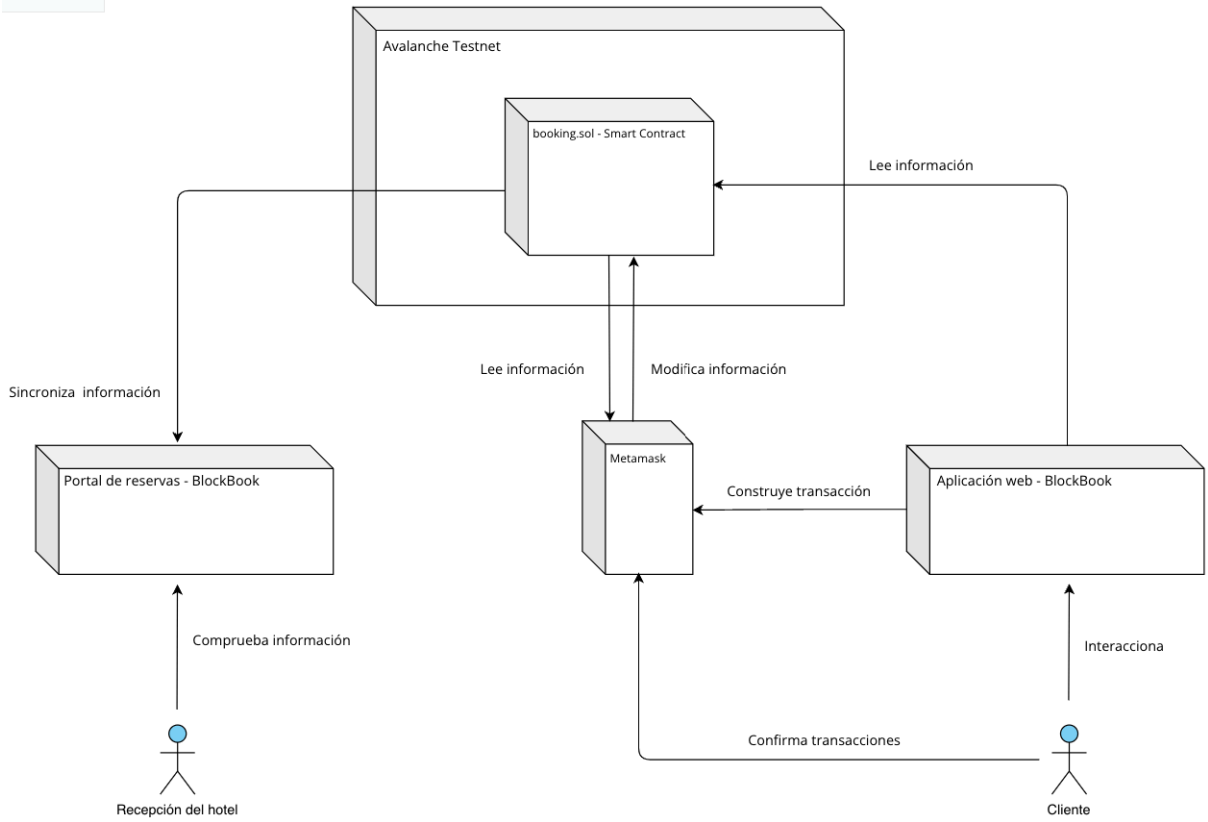


Figura 16. Esquema de comunicaciones entre los distintos componentes del sistema.

Después de especificar como funcionan las distintas partes del sistema, es necesario razonar con detalle el uso de la tecnología Blockchain frente a otras tecnologías backend y almacenamiento de datos tradicionales (estructura cliente, servidor y base de datos).

1. Desde el punto de vista económico, no es necesario mantener servidores ni alquilar servicios cloud para el backend o almacenamiento de datos. La tecnología blockchain es capaz de mantener de formar descentralizada nuestro contrato inteligente, lo que nos brinda de mayor fiabilidad porque tendrían que averiarse cada uno de los nodos, en contra de las bases de datos tradicionales, donde cualquier avería inhabilitaría el acceso a la aplicación web.

En cuanto al pago de cuotas, las únicas cuotas a pagar son el despliegue inicial del contrato inteligente y las comisiones cobradas por el uso de la red blockchain, teniendo en cuenta, que solo producirá un coste añadido aquellas funciones que modifican el estado del contrato inteligente, y en el caso de cancelación de la reserva, el pago de la cuota corre a cargo del cliente.

2. Otro aspecto destacado es la capacidad de incorporar de forma automática los pagos instantáneos. La tecnología blockchain permite al instante enviar criptomonedas de forma global con comisiones bastantes más bajas comparadas con las comisiones de los bancos tradicionales. Por tanto, el cliente puede realizar un pago a través de nuestra interfaz web sincronizada por su cuenta de Metamask.

3. En el ámbito de la seguridad y la integridad de la información, la tecnología Blockchain destaca por almacenar información inmutable. En los sistemas de almacenamiento tradicionales, la información es modificable. Por tanto, la única manera para que un atacante pueda modificar la información almacenada en el contrato inteligente es corromper la red blockchain completa, lo cuál requiere una capacidad de cómputo inasumible económicamente. Sin embargo, en sistemas con las bases de datos tradicionales, cualquier atacante puede alterar el estado del sistema si consigue infiltrarse.

4.2 Tecnologías y herramientas utilizadas

4.2.1 JavaScript

JavaScript es un lenguaje de programación comúnmente utilizado en el desarrollo web. Puede utilizar secuencias de comandos del lado del cliente, lo que significa que el

navegador web del cliente procesa el código fuente en lugar del servidor web. Esto significa que las funciones de JavaScript pueden ejecutarse después de que se haya cargado una página web sin comunicarse con el servidor.

El código JavaScript se puede insertar en cualquier lugar dentro del HTML de una página web. Sin embargo, solo la salida del código del lado del servidor se muestra en el HTML, mientras que el código JavaScript permanece totalmente visible en el código fuente de la página web. También se puede hacer referencia a él en un archivo .JS separado, que también se puede ver en un navegador **[37].**

Figura 17. Logo de JavaScript.

4.2.2 HTML

El lenguaje de marcado de hipertexto (HTML) es el conjunto de símbolos o códigos de marcado insertados en un archivo destinado a mostrarse en Internet. El marcado les dice a los navegadores web cómo mostrar las palabras e imágenes de una página web.

Cada código de marcado individual (que estaría entre los caracteres "<" y ">") se denomina elemento, aunque muchas personas también se refieren a él como una etiqueta. Algunos elementos vienen en pares que indican cuándo comenzará y cuándo terminará algún efecto de visualización.

En esencia, HTML es una serie de códigos cortos escritos en un archivo de texto. Estas son las etiquetas que impulsan las capacidades de HTML. El texto se guarda como un archivo HTML y se visualiza a través de un navegador web. El navegador lee el archivo

y traduce el texto a una forma visible, siguiendo las instrucciones de los códigos que el autor usó para escribir lo que se convierte en la representación visible. Escribir HTML requiere que las etiquetas se utilicen correctamente para crear la visión del autor **[38].**

Figura 18. Logo de HTML5.

4.2.3 Bootstrap

Es un framework CSS de código abierto que favorece el desarrollo web de una forma más sencilla y rápida. Incluye plantillas de diseño basadas en HTML y CSS con las que es posible modificar fuentes, formularios, botones, tablas, navegaciones, menús desplegables, etc. También existe la posibilidad de utilizar extensiones adicionales de JavaScript. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end **[39].**

Figura 19. Logo de Bootstrap.

4.2.4 Remix IDE

Remix IDE es una aplicación web y de escritorio de código abierto. Fomenta un ciclo de desarrollo rápido y tiene un amplio conjunto de complementos con GUI intuitivas. Remix se utiliza para todo el proceso de desarrollo del contrato, así como para actuar como un campo de juego para aprender y enseñar Ethereum.

Además, Remix IDE es una poderosa herramienta de código abierto que lo ayuda a escribir contratos de Solidity directamente desde el navegador. Está escrito en JavaScript y es compatible con el uso en el navegador, pero se ejecuta localmente y en una versión de escritorio. Remix IDE tiene módulos para probar, depurar e implementar contratos inteligentes y mucho más. Disponible en remix.ethereum.org.

Figura 20. Logo de Remix IDE

4.2.5 Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity).

Figura 21. Logo de Visual Studio Code

4.2.6 Web3.js – API Javascript de Ethereum

Web3.js es una colección de bibliotecas que permiten interactuar con un nodo Ethereum local o remoto mediante HTTP, IPC o WebSocket.

Figura 22. Web3 rodeado de sus sistemas compatibles.

4.2.7 MetaMask

MetaMask es una extensión del navegador web de Ethereum que actúa como una billetera de Ethereum y una interfaz para DApps basadas en Ethereum.

En otras palabras, MetaMask permite a los usuarios almacenar datos relacionados con Ethereum, como direcciones públicas y claves privadas, como cualquier otra billetera de Ethereum (es decir, podemos guardar nuestros tokens allí), y permite a los usuarios interactuar con sitios web que ejecutan aplicaciones basadas en Ethereum y contratos inteligentes. (Es decir, convierte nuestro navegador web en un navegador Ethereum) **[40].**

La conclusión es que MetaMask permite a los usuarios hacer cualquier cosa relacionada con Ethereum (como interactuar con contratos inteligentes, enviar/almacenar/recibir Ether o interactuar con aplicaciones web basadas en Ethereum) simplemente ejecutando una extensión de navegador simple.

Figura 23. Logo de MetaMask.

4.3 Smart Contract

Una vez aclarados los motivos del uso de la tecnología blockchain, en este apartado se explicará el porqué del uso de contratos inteligentes en el presente proyecto.

Un Smart Contract puede conformar el backend del sistema, ya que puede almacenar la información (obviando las transacciones con criptomonedas), y aprovechar esa información para programar diversas funcionalidades (en nuestro caso, programar una reserva hotelera).

En nuestro proyecto, se ha desarrollado un único Smart Contract (denominado Booking) que abarca todas las funcionalidades del sistema (gestión de reservas, backend del hotel, backend de la aplicación web, transacciones… etcétera). La descripción detalla del contrato inteligente se incluye en el apéndice A.

Booking.sol (Contrato inteligente)

Este contrato inteligente, implementa la funcionalidad necesaria para la creación, confirmación, cancelación y comprobación de reservas hoteleras. Además, incorpora la

estructura de datos necesaria para el correcto funcionamiento del sistema. Por un lado, mediante una estructura de datos se representan a las reservas, y otra estructura representaría el hotel en cuestión. También incorpora métodos "getters" para recuperar información y "setters" para modificar algunos parámetros cuando el usuario quiera cancelar su reserva.

Este contrato es capaz de comprobar si el usuario tenía o no reserva, y en el caso que el usuario lo solicite, devolverle el 95% del coste de la reserva. Tanto para realizar la devolución de los fondos como para comprobar si el cliente que entra un determinado día al hotel es el propietario de la reserva, el presente contrato inteligente implementa mecanismos de seguridad para comprobar la veracidad de su identidad. Por tanto, para que un usuario malicioso no cancele reservas en nuestro sistema, tendrá que introducir un hash conformado por 66 caracteres correspondiente al hash de la reserva, y como medida adicional, una palabra de recuerdo compuesta por ocho dígitos aleatorios, lo que limita al atacante la opción de usar hardware adicional para intentar cancelar una reserva.

Los únicos datos que recoge el contrato inteligente del usuario son el documento nacional de identidad (no lo guardaremos en claro, guardaremos el hash de este), hotel en el que se alojará, fecha de entrada y noches que permanecerá en él. Por tanto, el usuario ha de presentar su documento nacional de identidad y su hash de reserva el día de entrada al hotel.

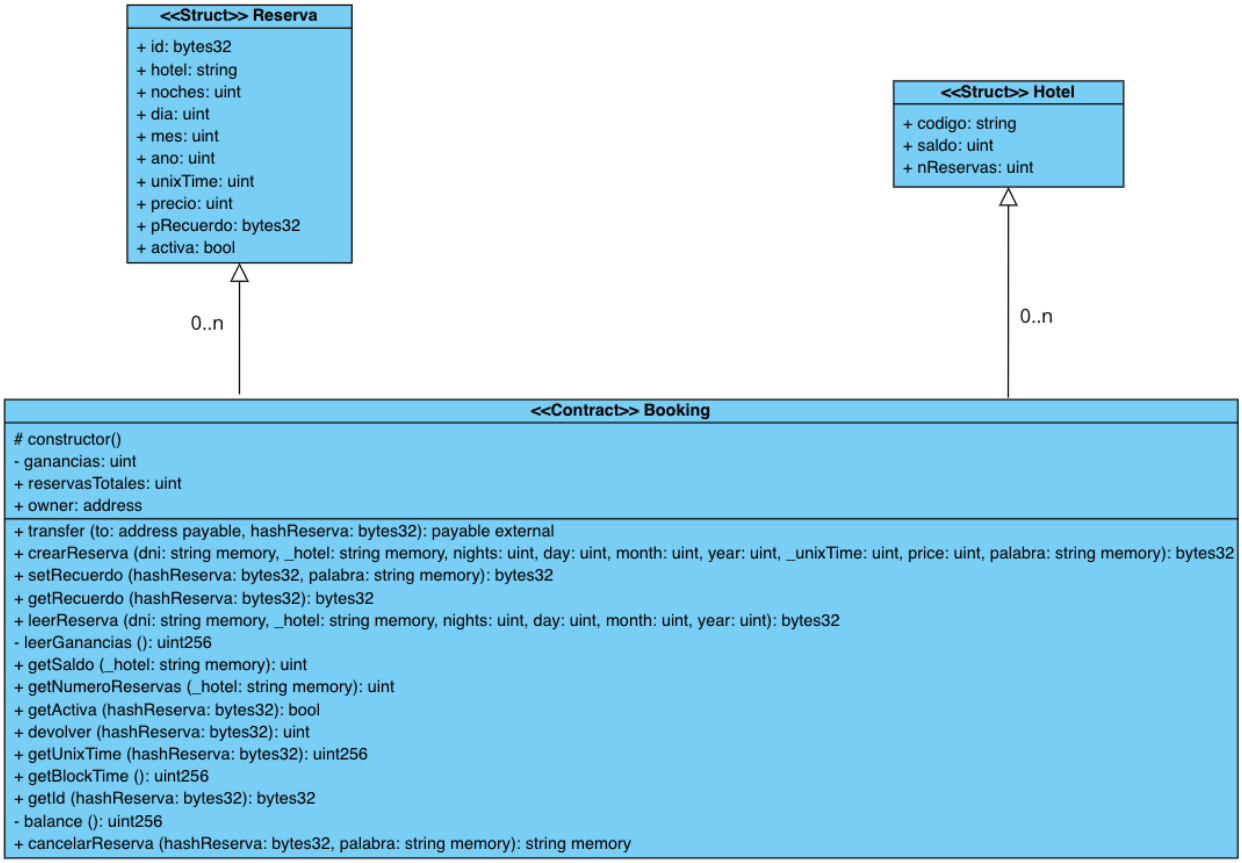


Figura 24. Modelado del contrato inteligente.

4.4 Aplicación Web

La aplicación web ha sido desplegada en la testnet de Avalanche para simular el funcionamiento del sistema como si fuese en la cadena principal, pero dotando a los desarrolladores de mayores recursos (p. ej. Las monedas AVAX “falsas” del entorno de prueba). Esta testnet ofrece las herramientas necesarias para realzar la presente prueba de concepto. Para ello, era necesario crear una interfaz de usuario que permita al mismo interactuar con la funcionalidad del contrato inteligente.

La aplicación web, tal y como hemos mencionado en las tecnologías utilizadas, ha sido implementada utilizando Javascript, donde en ciertos componentes de la web se han desarrollado numerosos scripts para interactuar con el contrato inteligente.

Respecto al front-end, hemos usado Bootstrap y hemos aplicado una de sus plantillas a nuestro sistema.

Nuestra aplicación web se divide en 4 pantallas principales: Inicio, Sobre nosotros, Ofertas y Cancela tu reserva, tal y como muestra la figura 25. A parte de estos componentes tenemos una web única para la recepción del hotel "checking.html" o formalmente conocido como "Gestor de reservas de BlockBook", usada para la comprobación y validación de una reserva. Realmente, las pantallas que necesitan comunicación con el contrato inteligente son: "Cancela tu reserva", "Gestor de reservas de BlockBook" y las páginas de reservas donde el cliente introduce sus datos de reserva. Esta comunicación es posible gracias a web3.js, una colección de bibliotecas que permiten interactuar con un nodo Ethereum local o remoto mediante HTTP entre otros. Web3.js también es capaz de asegurarse que la cartera virtual del cliente está conectada y a través de Javascript, comunicarse y traspasar información al contrato inteligente desde el propio HTML.

Hay que tener en cuenta que la forma de comunicarse con el contrato inteligente varía del tipo de función que ejecute. Por ejemplo, las llamadas de lectura no tienen ningún coste asociado y se hace directamente desde web3.js; y escritura, donde se envía la información al contrato inteligente a través de la transacción que construye MetaMask, y por tanto, al modificar valores en la blockchain, habría que pagar una comisión de gas. Principalmente, este tipo de transacciones ocurren cuando el usuario interacciona con un botón y posteriormente, acepta la alerta producida por el cobro de la transacción. La respuesta a estas peticiones del usuario, quedan a cargo del front-end, que mostrará un mensaje de si el envío ha tenido éxito o no (p.ej. Al realizar una reserva si el pago ha sido correcto).

La singularidad de esta aplicación web es la inexistencia de una ventana para los usuarios, ya que no es necesario tener una cuenta en el sistema para poder realizar una reserva (descartando los métodos de acceso tradicionales). El contrato inteligente considera usuarios a todos aquellos que interactúen con él. Además, el balance del contrato inteligente (es decir, el 5% de las ganancias), solo podrá ser retirado por el propietario del contrato inteligente.

Se dará una explicación más detalla de cada ventana de la aplicación web en el apéndice B.

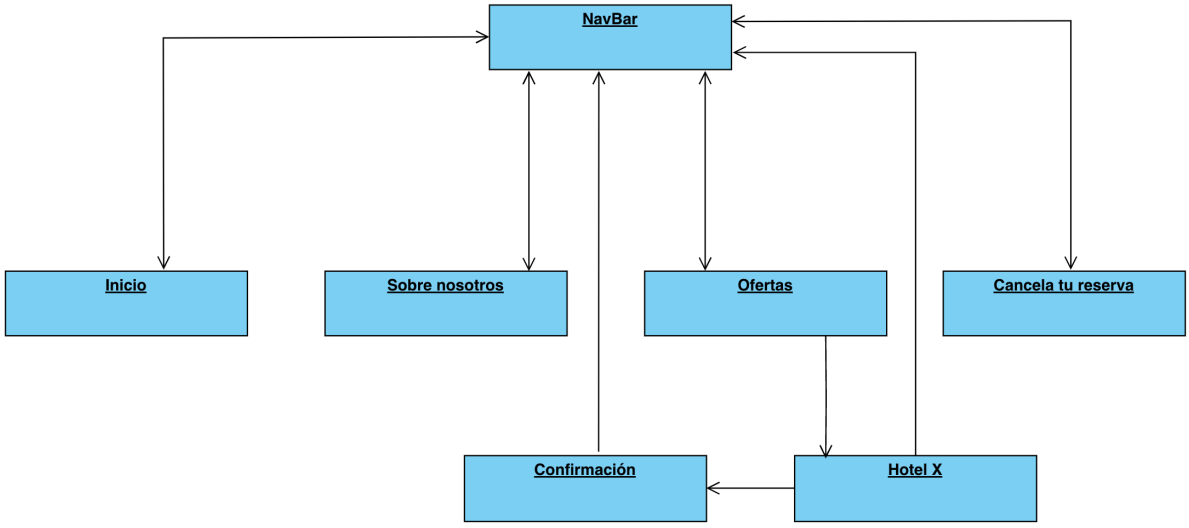


Figura 25. Esquema de navegación de BlockBook.

4.5 Problemas durante el desarrollo del sistema

En este apartado se explicarán los principales problemas que se tuvieron durante el desarrollo del sistema, así como la solución de estos.

En general, el proyecto ha tenido unos plazos bastantes cortos y exigentes, por lo que uno de los principales problemas que ha existido ha sido el tiempo. Por ello, se decidió desarrollar el sistema en HTML/CSS, ya que estaba familiarizado con estos lenguajes,

aunque la idea inicial era empezar un proyecto React, recortando así el tiempo de aprendizaje.

Además, también se invirtió bastante tiempo en el estudio previo, ya que era necesario realizarlo de forma exhaustiva y elegir bien la tecnología blockchain a desarrollar. Hay que tener en cuenta que esta tecnología al ser bastante nueva no existe tanta información accesible como la existente en las tecnologías tradicionales. Nuestro sistema, ha sido lanzado a la testnet de Avalanche, y como comentamos en la sección 3.2.2 Avalanche, el proyecto fue lanzado en 2020, por lo que la documentación es más escasa que en el caso de Ethereum. Pese a ello, decidimos seguir la conclusión del estudio para que el sistema sea más rápido, seguro y escalable.

Durante la realización de la prueba de concepto, también han surgido problemas en el desarrollo del contrato inteligente. Todo ello, debido al lenguaje de programación Solidity. Al ser un lenguaje de bajo nivel, el uso de operaciones matemáticas puede llevar a problemas por el acarreo de bits (p.ej. Al especificar la cantidad de wei en la transacción). Además, han surgido problemas entre la cartera de MetaMask del cliente y el contrato inteligente, ya que existen varias funciones en Solidity para llamar a MetaMask para realizar una transacción de un determinado número de wei.

Por último, el problema que ha quitado más tiempo durante el desarrollo de la aplicación web ha sido la devolución parcial del importe pagado por el cliente al cancelar la reserva. MetaMask revertía todo tipo de devoluciones a causa de un overflow. El problema residía en que se utilizaba la misma cuenta de MetaMask para implementar el contrato y para realizar una reserva de prueba. Tras varios días, buscando información, se decidió crear otra cuenta secundaria, además de utilizar otras

funciones de Solidity para traspasar fondos (p.ej. Uso de “transfer ()” en vez de “send ()”).

4.6 Observaciones finales

Se podría afirmar que se ha conseguido implementar la tecnología blockchain en una aplicación web para gestionar las reservas hoteleras. Respecto a los objetivos establecidos nos encontramos que:

El objetivo de desarrollar un contrato inteligente que logre gestionar las reservas hoteleras funcionando como backend. Además, en la prueba de concepto del mismo, admite pagar con la criptomoneda AVAX para la reserva de habitaciones.

Se ha completado con éxito el estudio analítico previo a la prueba de concepto, estudiando las ventajas y desventajas de cada blockchain, escogiendo la que mejor se ajustaba al proyecto y más ventajas poseía. Además, se ha tenido en cuenta cuál de ellas puede tener una mayor proyección en el futuro, ya que por tomar una mala elección podría haber peligrado la viabilidad de la solución.

No se ha podido abarcar en la gestión de habitaciones de cada hotel, así como en los diferentes tipos existentes de estas, por lo que se ofrece como una posible ampliación del presente proyecto. Para ello, se propone a estudio el uso de un oráculo. Un oráculo tiene como principal función, ser un servicio mediante el cual una blockchain o un smart contract se nutre de información externa a la blockchain sobre la que se ejecuta (se entiende por información externa, la disponibilidad de cada habitación del hotel) **[41].** Por tanto, nuestro sistema tal y como está programado, siempre encontrará habitaciones libres, algo muy alejado de la realidad.

5

Conclusiones

En esta sección se detallarán las conclusiones obtenidas de la realización del proyecto.

Durante la fase de análisis se ha ido mostrando como una de las blockchains con mayor capitalización de mercado y de la que todo el mundo habla de ella, Ethereum, no es capaz de procesar una gran cantidad de transacciones por segundo superiores a sus principales competidores. Además, Ethereum tiene numerosos estudios y pruebas en su contra. Seguramente, muchos lectores del presente trabajo hayan pesado al inicio de este, que Ethereum sería la blockchain más factible para el proyecto, precisamente dejándonos llevar por la fama que ha adquirido. Sin embargo, como pudimos observar en el estudio analítico, de las cinco blockchains comparadas, Ethereum logró quedar en última posición. Es cierto que a Ethereum le queda un gran camino, sobre todo con la próxima actualización hacia Ethereum 2.0, la cual estoy convencido que se hablará de ella en próximos trabajos de fin de grado.

Sobre Avalanche, me gustaría recalcar los grandes resultados que ha conseguido en nuestro estudio analítico siendo aún un proyecto tan joven. Por tanto, le auguro mucho potencial y pienso que la gran comunidad que se está creando en torno a ella, hará que sea una de las principales tecnologías blockchain del futuro.

Tras realizar el estudio previo teórico, se puede sacar en conclusión que el principal problema de la mayoría de ellas es la escalabilidad. Aún que hemos considerado algunas blockchains como “escalables”, si es cierto que no presentan soluciones sobresalientes.

La tecnología blockchain saltó a la fama con Bitcoin, en el año 2008. Pese a que ha evolucionado bastantes estos catorce años, aún sigue siendo una tecnología inmadura a la que le quedan varios aspectos por resolver, la escalabilidad (como hemos comentado antes), la capacidad de procesar grandes volúmenes de datos; y el ecosistema de aplicaciones en la envuelven. Una de esas partes del ecosistema son las aplicaciones que producen una interacción entre la blockchain y el usuario final, tal y como hemos podido demostrar en este trabajo fin de grado. Una colección de herramientas que entre otras cosas buscan que los desarrolladores se sientan cómodos con ellas para intentar hacer una comunidad más grande (p.ej. Los entornos de desarrollo online de Smart Contract).

Por último, me gustaría introducir otra parte del ecosistema blockchain, por el cual siento bastante curiosidad y me gustaría realizar un estudio sobre él y que puede ser objeto de estudio en las próximas décadas. Esta parte del ecosistema se trata del "Metaverso". Hoy en día, es un concepto que marca la siguiente generación de internet, y que pone como objetivo sentir una experiencia inversiva y multisensorial con el uso aplicado de diversos dispositivos. El "Metaverso" engloba muchas características de la blockchain y diversas funcionalidades, como los NFTs o proyectos como CityDao. Por tanto, a la comunidad le queda bastante por descubrir, entre ellos Mark Zuckerberg (fundador de Facebook), quien puso el foco de atención en el llamado "Metaverso".

Bibliografía

[1] Hayes, Adam. (20 de enero de 2022). Blockchain Definition: What You Need To Know. Investopedia.
https://www.investopedia.com/terms/b/blockchain.asp
[2] Frankenfield, Jake. (26 de mayo de 2021). Smart Contracts Definition. Investopedia.
https://www.investopedia.com/terms/s/smart-contracts.asp
[3] Chen, James. (1 de enero de 2022). Transaction Definition. Investopedia.
https://www.investopedia.com/terms/t/transaction.asp
[4] Frankenfield, Jake. (31 de marzo de 2021). Distributed Applications (DApps) Definition. Investopedia.
https://www.investopedia.com/terms/d/distributed-applications-apps.asp
[5] Kramer, Melanie. (30 de abril de 2019). What Are Consensus Protocols? Decrypt.
https://decrypt.co/resources/consensus-protocols-what-are-they-guide-how-to-explainer
[6]. Kathleen E. Wegrzyn, Eugenia Wang. (19 de Agosto de 2021). Types of Blockchain: Public, Private, or Something in Between. Foley.
https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between
[7] Binance Blog. (13 de diciembre de 2021). Layer 1 Blockchain Tokens: Everything You Need to Know. Binance.
https://www.binance.com/en/blog/fiat/layer-1-blockchain-tokens-everything-you-need-to-know-421499824684903155
[8] Román, Rodrigo. (s.f.). Futuro de Ethereum. Presentación PowerPoint del Curso de Extensión Universitaria en Tecnologías Blockchain.
[9] Paszke, Antoni. (s.f.). Layer 2. Binance.
https://academy.binance.com/en/glossary/layer-2
[10] Longchamp, Yves. (2 de diciembre de 2021). The different layer-2 solutions for Ethereum. Crypto Valley Journal.
https://cvj.ch/en/education/basics/the-different-layer-2-solutions-for-ethereum/
[11] Morey, Jose. (25 de octubre de 2021). The Future Of Blockchain in Healthcare. Forbes.
https://www.forbes.com/sites/forbestechcouncil/2021/10/25/the-future-of-blockchain-in-healthcare/?sh=1431967b541f

[12] Mohd, Javaid. Abid, Haleem. Ravi, Pratap Singh. Shahbaz, Khan. Rajiv, Suman. (diciembre de 2021). Blockchain technology applications for Industry 4.0: A literature-based review. Elsevier. https://www.sciencedirect.com/science/article/pii/S2096720921000221#:~:text=Blockchains%20could%20also%20facilitate%20voting,of%20the%20way%20parts%20work.
[13] Laaper, Stephen. Fitzgerald, Joseph. (s.f.). Using blockchain to drive supply chain transparency. Deloitte. https://www2.deloitte.com/us/en/pages/operations/articles/blockchain-supply-chain-innovation.html
[14] Frankenfield, Jake. (26 de mayo de 2021). Smart Contracts Definition. Investopedia. https://www.investopedia.com/terms/s/smart-contracts.asp
[15] Coinbase.com. (s.f.). What is a smart contract? Coinbase. https://www.coinbase.com/es/learn/crypto-basics/what-is-a-smart-contract
[16] CoinTelegraph. (s.f.). What are smart contracts in blockchain and how do they work? https://cointelegraph.com/ethereum-for-beginners/what-are-smart-contracts-a-beginners-guide-to-automated-agreements
[17] Sahil, Sen. (27 de diciembre de 2021). An overview of how smart contracts work on Ethereum. QuickNode. https://www.quicknode.com/guides/solidity/an-overview-of-how-smart-contracts-work-on-ethereum
[18] Newbery, Emma. (21 de septiembre de 2021). 6 Top Cryptocurrencies With Smart Contracts. The Ascent. https://www.fool.com/the-ascent/cryptocurrency/articles/6-top-cryptocurrencies-with-smart-contracts/
[19] Coinbase. (s.f.). What is Ethereum? https://www.coinbase.com/es/learn/crypto-basics/what-is-ethereum#is-ethereum-secure
[20] Redmna, Jamie. (19 de enero de 2022). Ethereum Fees Drop 35% Since Last Week, Average ETH Gas Fee Still Above $30 per Transfer. https://news.bitcoin.com/ethereum-fees-drop-35-since-last-week-average-eth-gas-fee-still-above-30-per-transfer/#:~:text=The%20median%2Dsized%20fee%20to,ether%20or%20%2414.78%20per%20transfer.
[21] The Shrimpy Team, (31 de diciembre de 2021). What is Avalanche (AVAX)? The Next DeFi Blockchain Explained. Shrimpy Academy. https://academy.shrimpy.io/post/what-is-avalanche-avax-the-next-defi-blockchain-explained
[22] Oyinloye, Bosun. (s.f.). What is Avalanche? Coin Rivet. https://coinrivet.com/guides/altcoins/what-is-avalanche/

[23] Rejolut. (s.f.). Avalanche Development.
https://rejolut.com/blockchain/avalanche-development/
[24] Avalanche Docs. (s.f.). Launch Your Ethereum dApp on Avalanche.
https://docs.avax.network/build/tutorials/platform/launch-your-ethereum-dapp/
[25] Avax Explorer Network. (s.f.). Validators.
https://explorer.avax.network/validators
[26] Avalanche Docs. (s.f.). Avax Transaction Fee.
https://docs.avax.network/learn/platform-overview/transaction-fees/
[27] Coinbase. (s.f.). What is Polygon (MATIC)?
https://www.coinbase.com/es/learn/crypto-basics/what-is-polygon
[28] Polygon Technology. (s.f.). Polygon.
https://polygon.technology/technology/
[29] Polygon Staking. (s.f.). Wallet Polygon Technology.
https://wallet.polygon.technology/staking/
[30] Rhodilee, Jean Dolor. (22 de octubre de 2021). Binance Smart Chain (BSC) Explained | A Beginner's Guide.
https://finbold.com/guide/binance-smart-chain/
[31] CoinTelegraph. (s.f.). BSC network: A beginner's guide to the Binance Smart Chain blockchain.
https://cointelegraph.com/blockchain-for-beginners/bsc-network-beginners-guide-to-the-binance-smart-chain-blockchain
[32] Hyperledger Besu Docs. (21 de diciembre de 2021). Besu Enterprise Ethereum client.
https://besu.hyperledger.org/en/stable/
[33] Hyperledger Besu Docs. (21 de diciembre de 2021). Consensus Protocols.
https://besu.hyperledger.org/en/stable/Concepts/Consensus-Protocols/Overview-Consensus/
[34] Universidad de Alcalá. (5 de junio de 2020). Hyperledger Besu. MasterBlockchain.
https://masterblockchain.net/hyperledger-besu-master-blockchain-online/
[35] Hyperledger Besu Team. (6 de agosto de 2020). Hyperledger Besu 1.5 Performance Enhancements.
https://www.hyperledger.org/blog/2020/08/06/hyperledger-besu-1-5-performance-enhancements
[36] Hyperledger Besu Docs. (20 de julio de 2021). Quorum Developer Quickstart tutorial.
https://besu.hyperledger.org/en/stable/Tutorials/Examples/Private-Network-Example/
[37] TechTerms.com. (8 de agosto de 2014). Javascript.
https://techterms.com/definition/javascript

[38] Hayes, Adam. (13 de noviembre de 2020). HyperText Markup-Language-HTML. Investopedia.
https://www.investopedia.com/terms/h/html.asp
[39] Arimetrics. (s.f.). What is Bootstrap.
https://www.arimetrics.com/en/digital-glossary/bootstrap
[40] CryptoCurrency Facts. (s.f.). MetaMask Explained.
https://cryptocurrencyfacts.com/metamask-explained/
[41] Bit2Me Academy. (s.f.). ¿Qué son los oráculos blockchain?
https://academy.bit2me.com/que-es-oraculos-blockchain/

Apéndice A

Smart Contract

En este apéndice se explicará con más detalle el contrato inteligente desplegado en la Testnet de Avalanche.

El contrato inteligente se utiliza para la gestión de reservas hoteleras (crear, cancelar y leer reservas). En él se declaran las reservas almacenadas en un mapping (mapa de claves y valores). Cada reserva contiene un id (hash del DNI), el nombre del hotel donde ha realizado la reserva, el número de noches, la fecha de entrada, el UnixTime (medida de tiempo) de la reserva, el precio por noche, el hash de la palabra de recuerdo y una variable booleana que indicará si la reserva está activa o no. También se ha declarado un mapping de los hoteles adheridos con su respectiva información (código del hotel, saldo acumulado y número de reservas totales).

Además de estas variables, se ha declarado la variable privada ganancias, que irá acumulando el 5% del coste de cada reserva (siguiendo nuestro modelo de negocio). Con ello, también se ha añadido la variable "reservasTotales" para llevar la cuenta de cuantas reservas se han hecho en total para posteriormente, compararlas con las ganancias.

El constructor declara propietario del smart contract a la dirección que implemente el mismo en la blockchain, en este caso, nuestra dirección.

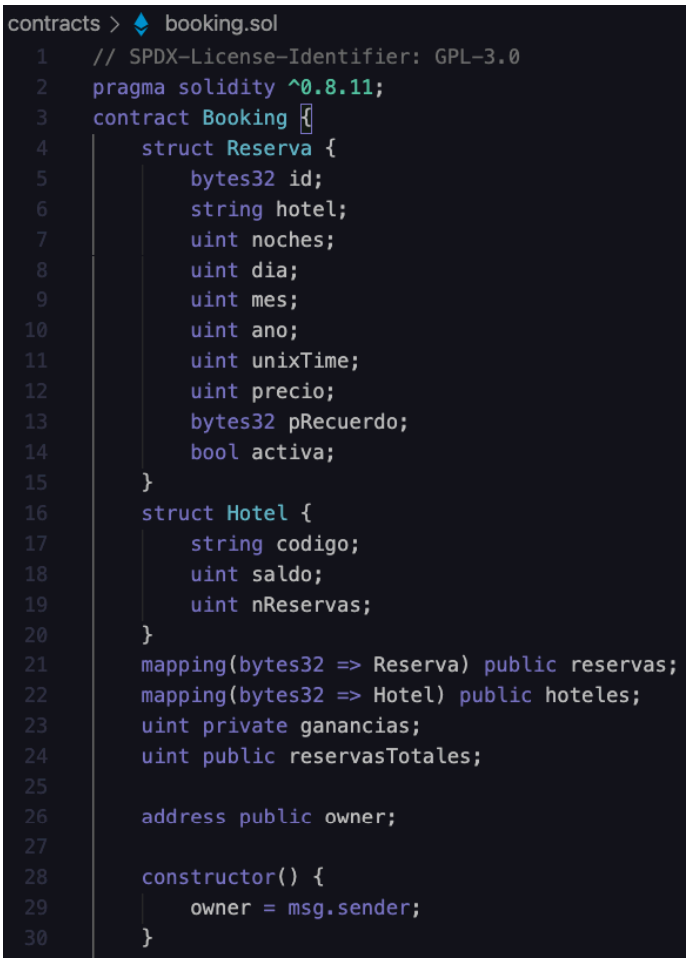


Figura 26. Variables declaradas del contrato inteligente Booking.

Las funciones implementadas en el contrato son:

- La función transfer se encarga de hacer la devolución de la reserva en el caso de que el cliente lo solicitara. Recibe como parámetros de entrada, la dirección de la cartera MetaMask del cliente y el hash de la reserva que recibió cuando reservó. Además, actualiza los parámetros de contabilidad de reservas así como actualizar los saldos correspondientes.

```
function transfer(address payable to, bytes32 hashReserva) payable external {
    bytes32 hashHotel = keccak256(abi.encodePacked(reservas[hashReserva].hotel));
    hoteles[hashHotel].saldo = hoteles[hashHotel].saldo - devolver(hashReserva);
    hoteles[hashHotel].nReservas--;
    reservasTotales--;
    reservas[hashReserva].activa = false;
    to.transfer(devolver(hashReserva));
}
```

Figura 27. Contrato inteligente Booking, función transfer.

- La función crearReserva toma como parámetros de entrada todas las variables descritas anteriormente pertenecientes a una reserva. La función es pública y del tipo payable, lo que significa que puede recibir Ether (en este caso el coste de la reserva en cuestión). Por tanto, crea una reserva y esta es guardada en el mapping.

```
function crearReserva(string memory dni, string memory _hotel, uint nights, uint day, uint month, uint year, uint _unixTime, uint price, string memory palabra)
    public
    payable
    returns (bytes32)
{
    if(leerReserva(dni,_hotel,nights,day,month,year)!=0x0000000000000000000000000000000000000000000000000000000000000000) {
        revert("Ya tienes reserva");
    }
    bytes32 hashReserva = keccak256(abi.encodePacked(dni, _hotel, nights, day, month, year));
    reservas[hashReserva] = Reserva(keccak256(abi.encodePacked(dni)),_hotel,nights,day,month,year,_unixTime,price,keccak256(abi.encodePacked(palabra)),true);
    reservasTotales=reservasTotales+1;
    bytes32 hashHotel = keccak256(abi.encodePacked(_hotel));
    uint s = getSaldo(_hotel);
    s=s+((msg.value*19)/20);
    uint nR = getNumeroReservas(_hotel);
    hoteles[hashHotel] = Hotel(_hotel,s,nR+1);
    ganancias=(msg.value/20)+ganancias;
    return hashReserva;
}
```

Figura 28. Contrato inteligente Booking, función crearReserva.

- La función setRecuerdo toma como entrada el hash de la reserva y una palabra aleatoria y cambia la palabra de recuerdo almacenada en la reserva, usando el algoritmo de cifrado keccak256.
- La función getRecuerdo toma como entrada el hash de la reserva y devuelve la palabra de recuerdo asociada a esa reserva.
- La función leerReserva es utilizada por el personal del hotel en el día de entrada del cliente, mediante el DNI, nombre del hotel en cuestión, número de noches y el propio día de entrada, devuelve el hash de la reserva asociada o 0 en caso contrario.

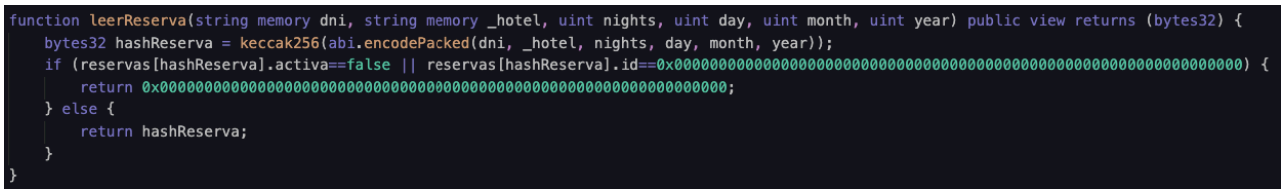
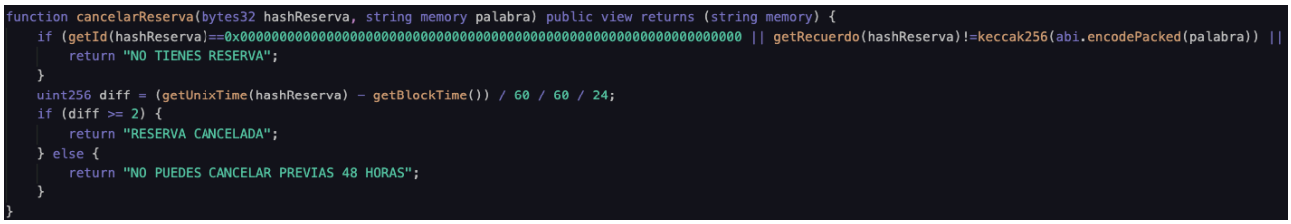


Figura 29. Contrato inteligente Booking, función leerReserva.

- La función privada leerGanancias devuelve las ganancias que hemos obtenido en total con las reservas vencidas.
- La función getSaldo devuelve el saldo del hotel en cuestión pasado como parámetro.
- La función getNumeroReservas devuelve las reservas totales realizadas en el hotel pasado como parámetro.
- La función getActiva determina si una reserva está activa o no, pasando el hash de la reserva como parámetro.
- La función devolver calcula la cantidad a devolver al cliente, es decir, el 95% del total de la reserva. Recibe como parámetro el propio hash de la reserva.
- La función getUnixTime devuelve el tiempo Unix en el que se realizó la reserva. Recibe como parámetro el propio hash de la reserva.
- La función getBlockTime devuelve el timestamp del bloque en la blockchain.
- La función getId devuelve el hash del DNI que contiene una determinada reserva. Recibe como parámetro el propio hash de la reserva.
- La función balance devuelve la cantidad de Ether bloqueada y depositada en el contrato inteligente.
- La función cancelarReserva nos informará si un usuario puede cancelar una reserva o no. Recibe como parámetros de entrada: el hash de la reserva y la palabra de recuerdo.



```
function cancelarReserva(bytes32 hashReserva, string memory palabra) public view returns (string memory) {
    if (getId(hashReserva)==0x0000000000000000000000000000000000000000000000000000000000000000 || getRecuerdo(hashReserva)!=keccak256(abi.encodePacked(palabra)) ||
        return "NO TIENES RESERVA";
    }
    uint256 diff = (getUnixTime(hashReserva) - getBlockTime()) / 60 / 60 / 24;
    if (diff >= 2) {
        return "RESERVA CANCELADA";
    } else {
        return "NO PUEDES CANCELAR PREVIAS 48 HORAS";
    }
}
```

Figura 30. Contrato inteligente Booking, función cancelarReserva.

Apéndice B

Aplicación Web

En este apéndice principalmente se mostrará la parte de la aplicación web que realiza la comunicación con el smart contract y las distintas pantallas que contiene, sin centrarse en el código de las últimas.

Las vistas que interactúan con el smart contract son: cancelación.html, checking.html, confirmado.html, only-you.html y vincci.html (estos últimas son análogas, por lo que solo mostraremos una de ellas). Todas estas vistas tienen en común la carga del contrato inteligente para que podamos interactuar con él, para ello son necesarias las siguientes funciones:

```
async function loadWeb3() {
    if (window.ethereum) {
        window.web3 = new Web3(window.ethereum);
        window.ethereum.enable();
    }
}

async function loadContract() {
    return await new window.web3.eth.Contract([
```

Figura 31. Funciones de carga del smart contract. LoadWeb3() y LoadContract().

La función loadWeb3 se encarga de cargar la biblioteca web3.js y loadContract carga el contrato inteligente, para ello, recibe como parámetros el ABI y la dirección del contrato inteligente (debido a la enorme extensión del ABI, no han sido mostrados en la figura 31.).

Además de estas funciones, necesitamos:

```
    async function getCurrentAccount() {
        const accounts = await window.web3.eth.getAccounts();
        return accounts[0];
    }

    async function load() {
        await loadWeb3();
        window.contract = await loadContract();
    }

    load();
</script>
```

Figura 32. Funciones de carga del smart contract. getCurrentAccount() y load().

La función load es necesaria para la carga del contrato inteligente y getCurrentAccount se encarga de recuperar la dirección de nuestra cartera virtual de MetaMask para que el sistema sepa a quién tiene que cobrar la reserva.

B.1 Cancelacion.html

Esta vista se encarga de proceder a la cancelación de una reserva si cumple las condiciones. El fragmento de código más relevante es el siguiente:

```
async function alerta(){
    var opcion = confirm("¿Estás seguro que quieres cancelar tu reserva? ");
    if (opcion == true) {
        cancelaReserva();
    }
}

async function cancelaReserva() {
    const account = await getCurrentAccount();
    const hash = document.getElementById("hash").value;
    const palabra = document.getElementById("recuerdo").value;
    var string = await window.contract.methods.cancelarReserva(hash,palabra).call();
    if (string=="RESERVA CANCELADA") {
        await window.contract.methods.transfer(account,hash).send({
            from: account,
            to: account,
            amount: 0
        });
    }
    alert(string);
}
```

Figura 33. Funciones alerta() y cancelarReserva() en cancelación.html

Antes de cancelar la reserva, alertamos al usuario para saber si está seguro de la cancelación (usamos la función alerta()). La función cancelarReserva captura del formulario web el hash de la reserva y la palabra de recuerdo y llama al smart contract para comprobar si existe o no. Si nos devuelve "RESERVA CANCELADA" llamaremos a la función transfer del contrato inteligente para iniciar la devolución.

Figura 34. Vista de cancelacion.html

B.2 Checking.html

La función de esta vista es comprobar que el usuario que entra al hotel es el verdadero propietario de la reserva. Por tanto, deberá adjuntar su DNI para que el sistema comprueba realmente que esa reserva existe. Si el hash devuelto coincide con el que el cliente tiene, es la persona adecuada. Además, se le puede preguntar por la palabra de recuerdo como medida de seguridad adicional.

El fragmento de código más relevante es el siguiente:

```
async function comprueba() {
    const dni = document.getElementById("dni").value;
    const hotel = document.getElementById("hotel").value;
    const dia = document.getElementById("start").value;
    const noches = document.getElementById("noches").value;
    var day = dia[8]+dia[9];
    var month = dia[5]+dia[6];
    var year = dia[0]+dia[1]+dia[2]+dia[3];
    var hashid = await window.contract.methods.leerReserva(dni, hotel, noches, day, month, year).call();
    if (hashid==0x0000000000000000000000000000000000000000) {
        alert("NO EXISTE NINGUNA RESERVA");
    } else {
        alert("RESERVA CONFIRMADA CON EL HASH: "+hashid);
    }
}
```

Figura 35. Función comprueba() en checking.html

La función comprueba llama a “leerReserva” en el smart contract y devuelve el hash de la reserva si los datos que se han introducido tienen asociada una reserva.

Figura 36. Vista de checking.html

B.3 Confirmado.html

La función de esta vista consiste en mostrar al cliente que la reserva ha sido confirmada, mostrando el hash de la reserva y la palabra de recuerdo. Para ello, se necesitan las siguientes funciones:

```
<script type="text/javascript">
    const  generateRandomString = () => {
        let recuerdo = document.getElementById("recuerdo");
        recuerdo.innerHTML=localStorage.getItem("recuerdo");
        console.log("Palabra de recuerdo: "+localStorage.getItem("recuerdo"));
    }
    const hash = () => {
        let hash = document.getElementById("hash");
        hash.innerHTML=localStorage.getItem("hash");
        console.log("HASH: "+localStorage.getItem("hash"));
    }

</script>
```

Figura 37. Funciones generateRandomString() y hash() en confirmado.html

La función generateRandomString recupera de la página anterior (only-you.html o vincci.html) la palabra aleatoria que se generó mediante el almacenamiento local. La función hash hace exactamente lo mismo, pero con el hash de la reserva.

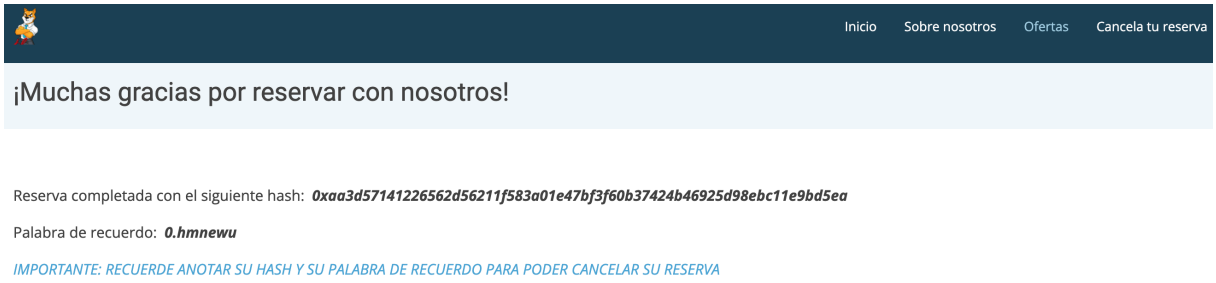


Figura 38. Vista de confirmado.html

B.4 Only-you.html y vincci.html

La función de esta vista consiste en recopilar los datos solicitados al cliente y comunicarse con el smart contract para que formalice la reserva y efectúe la transacción del cliente al contrato inteligente. Además, debemos asegurarnos de que el cliente no ha realizado una reserva previamente en esa misma fecha para ese mismo hotel. Para ello, se necesitan las siguientes funciones:

- La función generateRandomString genera una palabra de recuerdo aleatoria formada por 8 caracteres.
- La función alerta informa al usuario del precio total de la reserva, en el caso de que acepte, se guardará el hash y la palabra de recuerdo en el almacenamiento local y llamará a la función redirigir.
- La función redirigir muestra la vista anterior, confirmado.html.
- La función setReserva, recopila del formulario todos los datos adjuntados por el cliente y llamada a la función crearReserva del smart contract para guardar los datos de la reserva y crear la transacción con el precio total de la reserva. Esos fondos estarán bloqueados en la dirección del smart contract. Además, podemos observar los tres componentes de la transacción: from (dirección de nuestra cartera de MetaMask), to (dirección del smart contract) y value (cantidad total de la reserva en wei).

```
    const  generateRandomString = (num) => {
        const characters ='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
        let result1= Math.random().toString(36).substring(0,num);
        return result1;
    }

async function alerta(){
  var opcion = confirm("El precio total de la reserva serán "+document.getElementById("noches").value*3+" AVAX");
  if (opcion == true) {
    const palabra = await generateRandomString(8);
    const hash = await setReserva(palabra);
    localStorage.setItem("recuerdo", palabra);
    localStorage.setItem("hash", hash);
    redirigir();
    }
}

async function redirigir(){
  window.location.assign("http://127.0.0.1:8080/confirmado.html");
}

async function setReserva(palabra) {
    const account = await getCurrentAccount();
    const dia = document.getElementById("start").value;
    var unixTime = parseInt((new Date(document.getElementById("start").value).getTime() / 1000).toFixed(0))+86400;
    const dni = document.getElementById("dni").value;
    const noches = document.getElementById("noches").value;
    var day = dia[8]+dia[9];
    var month = dia[5]+dia[6];
    var year = dia[0]+dia[1]+dia[2]+dia[3];
    await window.contract.methods.crearReserva(dni,"ONLYYOU",noches,day,month,year,unixTime,3,palabra).send({
      from: account,
      to: 0x2902B5128957457Ff5585a4A698a595Bd3F45d84,
      value: '100000000000000000'*noches*3
    });
    return await window.contract.methods.leerReserva(dni,"ONLYYOU",noches,day,month,year).call();
}
```

Figura 39. Funciones generateRandomString(), alerta(), redirigir(), setReserva(palabra) en only-you.html

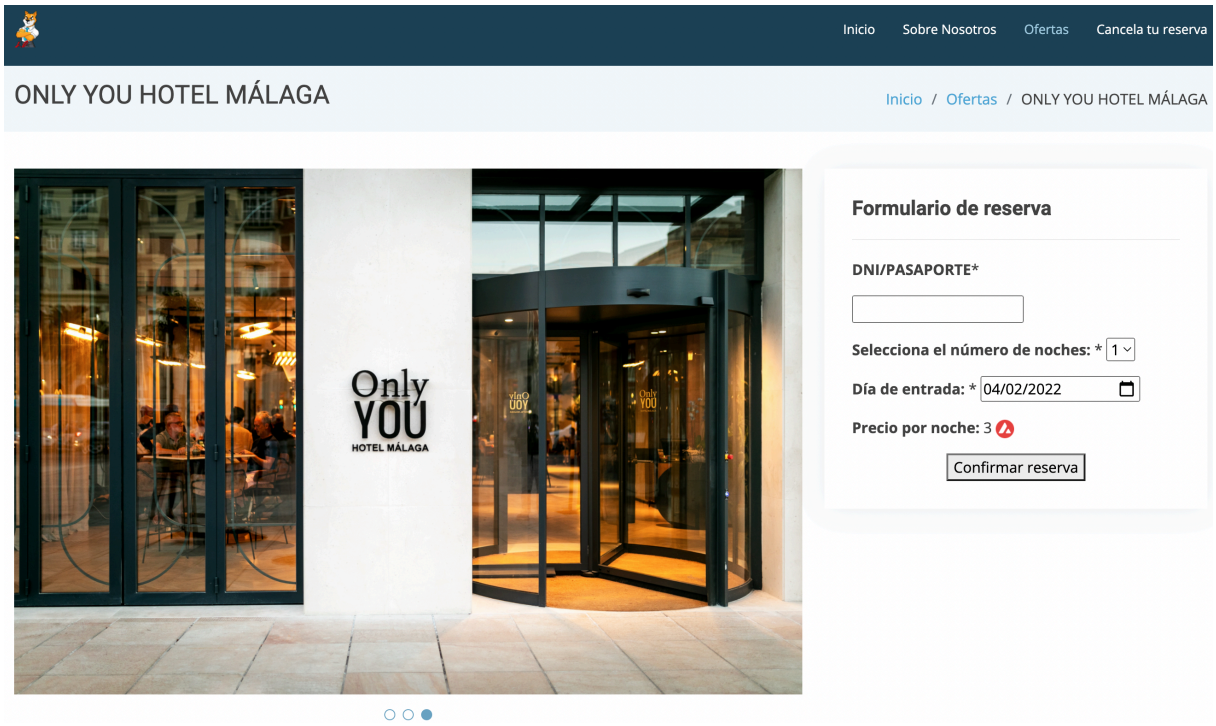


Figura 40. Vista de only-you.html

Apéndice C

Instalación

Para la instalación de la aplicación web se requiere http-server para montar un servidor local estático. Para ello, seguiremos los siguientes pasos:

1. Abrir el terminal en la carpeta raíz proyecto.

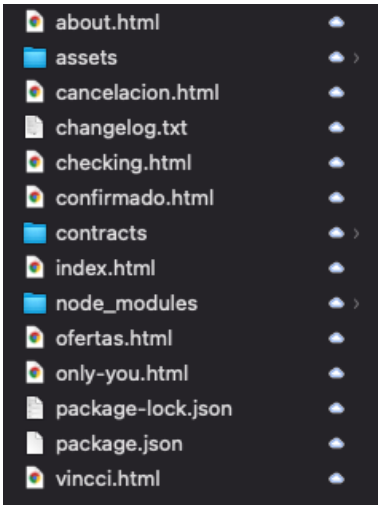


Figura 41. Carpeta raíz del proyecto.

2. Instalaremos http-server, para ello ejecutamos en nuestro terminal: “npm install --global http-server”.
3. Una vez instalado ejecutamos el comando “http-server” para iniciar la web en un servidor estático local.
4. Accedemos a la web escribiendo en el navegador la dirección del localhost: 127.0.0.1:8080 o la URL que nos indique http-server al terminar de ejecutarse.
5. Configuramos nuestra cuenta de MetaMask con la extensión para Chrome u otros navegadores. Si no se dispone de cuenta Metamask, debemos crearnos una.

6. En MetaMask, hacemos clic en el menú desplegable de Red y seleccionamos el RPC personalizado.

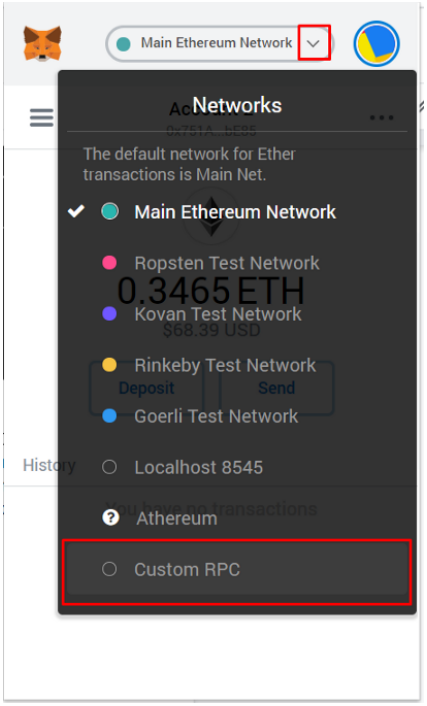


Figura 42. Selección de RPC personalizada en MetaMask.

7. Configuramos la red de Avalanche.

FUJI Testnet Settings:

- **Network Name**: Avalanche FUJI C-Chain
- **New RPC URL**: https://api.avax-test.network/ext/bc/C/rpc
- **ChainID**: `43113`
- **Symbol**: `AVAX`
- **Explorer**: https://testnet.snowtrace.io/

Figura 43. Configuración de la Testnet de Avalanche.

8. Asegúrate de que dispones de AVAX suficientes para poder realizar tu primera reserva. Puedes reclamar AVAX ficticios en: https://faucet.avax-test.network/

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA