



UNIVERSIDAD
NEBRIJA

DESARROLLO DE UN SISTEMA DE LOTERÍAS BLOCKCHAIN
UTILIZANDO NFTS.

JAIME TELLO SÁNCHEZ

UNIVERSIDAD NEBRIJA
GRADO EN INGENIERÍA INFORMÁTICA
MEMORIA TRABAJO FIN DE GRADO

Jaime Tello Sánchez

7 DE FEBRERO DE 2022



UNIVERSIDAD
NEBRIJA

DESARROLLO DE UN SISTEMA DE LOTERÍAS BLOCKCHAIN
UTILIZANDO NFTS.

JAIME TELLO SÁNCHEZ

UNIVERSIDAD NEBRIJA
GRADO EN INGENIERÍA INFORMÁTICA
MEMORIA TRABAJO FIN DE GRADO

Tutores:

María Teresa González Mac Dowell

Autor:

Jaime Tello Sánchez

12 DE SEPTIEMBRE DE 2022

PÁGINA DE DERECHOS

D. /Dña. Jaime Tello Sánchez autoriza a que el presente trabajo se guarde y custodie en los repositorios de la Universidad Nebrija y además NO autoriza a su disposición en abierto.

Resumen del proyecto.....	6
Memoria.....	8
Necesidades y objetivos	8
Necesidad detectada y justificación de su resolución	8
Requisitos técnicos y funcionales.....	9
Objetivos del proyecto	9
Bases de desarrollo del proyecto.....	10
Historia de la lotería	10
Bases matemáticas de la lotería.....	11
Teoría de la probabilidad.....	11
Esperanza matemática de la lotería	12
Tipos de loterías en España	13
Lotería Nacional	13
Primitiva y Bonoloto	13
Gordo de la primitiva	13
Euromillones	14
Quiniela y el Quinigol.....	15
Cupón de la Once	15
Ventaja de una lotería blockchain frente a las loterías tradicionales.....	16
Herramientas y tecnologías a utilizar	17
Blockchain	17
Definición	17
Cómo funciona blockchain	18
Elementos clave de blockchain.....	19
Tipos de blockchain.....	20
Casos de uso de blockchain	23
Solidity.....	24
Chainlink.....	24
Protocolo de Chainlink.....	25
Privacidad en Chainlink	26
Beneficios de Chainlink.....	26
Oráculos	27
Definición de oráculos	27
Tipos de Oráculos.....	27
Números aleatorios	28

Oráculo de aleatoriedad VRF de Chainlink	29
Computación cuántica	32
Librerías de Solidity.....	33
NFT.....	34
Tecnología blockchain Ethereum orientada a los NFTs.....	35
Criptomonedas.....	36
OpenZeppelin.....	38
Hooks.....	39
Librerías que se utilizan	40
ERC-20	40
ERC-721	41
ERC-1155	41
Líneas futuras	43
Entorno front-end de desarrollo del proyecto.....	43
Metaverso	43
Entono web	47
Arquitectura del proyecto	48
Fases de desarrollo del proyecto	52
Diseño e implementación de la lotería	53
Compilación, ejecución y conclusiones.....	66
Compilación y ejecución del código	66
Conclusiones sacadas del proyecto	72
Economía.....	72
Anexo.....	74
Glosario	74
Bibliografía.....	77

Resumen del proyecto

Desde 2009 que se creó la criptomoneda, estas tecnologías han ido creciendo y tomando popularidad en los últimos años. En 2017 es cuando Bitcoin se revalorizó de golpe y desde entonces han ido apareciendo distintas criptomonedas y entornos webs relacionados con ellas.

Tras estar trabajando en el departamento de *blockchain* de una empresa del sector banca, desde esta, se me propuso realizar el proyecto de fin de grado en colaboración con ellos. Y me propusieron el realizar una lotería online en alguna *testnet* o web free del metaverso, utilizando un criptoactivo propio. Debido a que si se llevase a cabo el proyecto con un criptoactivo propio, no tendría cabida, debido a que no tendría popularidad y las diferentes plataformas no la acatarían como moneda, se utilizará el Ether como moneda. Tras estudiar el tema y resultarme interesante acepté a realizar la investigación en el ámbito del *blockchain* y llevar a cabo el desarrollo del proyecto en colaboración con ellos, ya que, ellos me proporcionaban formación y medios para formarme en la materia y yo hacía el *backend*, de la aplicación, dejando en su mano, la parte del *front* y plataforma en la que se instaurará .

El proyecto se basa en la creación de una lotería en el metaverso utilizando un criptoactivo ya instaurado, aunque en el futuro se podría llegar a crear con otro *token* diferente creado por uno mismo, dependiendo de la red en la que se desee desarrollar el *front*. Este sistema de lotería se puede desarrollar en un entorno virtual, llamado el metaverso, debido al gran auge que está experimentando últimamente, de hecho, empresas referentes dentro del mundo de la informática como Meta (antiguamente denominada Facebook), han invertido grandes cantidades de dinero en dicha tecnología. Dentro de los metaversos hay distintos, dependiendo de la empresa que lo desarrolla, utilizando criptomonedas diferentes como método de pago. Se trata de un proyecto que aún no ha sido instaurado en ningún entorno del metaverso, aunque hay un proyecto de lotería *blockchain* denominado MylottoCoin con una estructura similar. Se trata de una investigación del desarrollo de este, pudiendo en un futuro desarrollar un *frontend* para instaurarlo en el metaverso que mejor se ajuste en dicho momento al proyecto. Por lo tanto, desarrollaremos el back de la *app* y un pequeño *front*, para poderlo verlo de forma visual, aunque en lo que se centrará el proyecto es en el *backend*.

Respecto a la estructura de la lotería se utilizará un formato de tokens no fungibles, como participaciones de esta. Los tokens no fungibles (NFT) son tokens únicos intercambiables por otro criptoactivo o coleccionables que al igual que el metaverso han ido cogiendo popularidad. Estos van a servir como si fueran boletos de lotería y su hash como el número con el que se participará. Estos boletos llevarán un identificador además del hash para que el usuario pueda elegir el que desee. Y estarán limitados a gusto del propietario, ya que, será él quien se encargue de lanzar la lotería. Una

vez que se haya cerrado el sorteo en curso, se abrirá de nuevo la posibilidad de comprar boletos para un nuevo sorteo.

Para extraer los números aleatorios se podría realizar mediante diferentes procedimientos. El primero es la programación en Solidity del oráculo de aleatoriedad VRF, el cual extrae los números de manera aleatoria. El segundo método, que se puede implementar en el proyecto, es a partir de la computación cuántica. Utilizando el simulador cuántico IBM Quantum Experience y el IBM Qiskit, que se trata de librerías en Python para poder extraer los valores aleatorios a través del simulador. Además también existen librerías ya instauradas en Keccak-256 que nos sirve para codificar el hash del bloque en el que estamos, lo modificamos y ya nos queda como resultante un número aleatorio, para tener una mayor entropía también le podemos aplicar más librerías como la de timestamp y hacer el módulo de ese conjunto de datos entre el número máximo de boletos vendidos.

Tras haber realizado un estudio de los distintos tipos de loterías y juegos de azar a lo largo de la historia, hemos decidido implementar un sistema, a partir del cual, siempre habrá un ganador del sorteo. Esto se realizará mediante la extracción de un número al azar. Al realizarse con los hashes y ser unos números muy grandes y con unas posibilidades de ganar ínfimas, se decide que siempre haya un ganador. Esto se puede realizar generando un ID para cada NFT, es decir, para cada boleto de los que se pongan a la venta y con un oráculo de aleatoriedad o mediante computación cuántica se extrae un boleto al azar, siendo ese el ganador, aunque también se puede llevar a cabo utilizando las librerías mencionadas anteriormente. Se han estudiado otros mecanismos, como podría ser un mecanismo de aproximación de un número aleatorio, pero el ir comparando cada hash de cada boleto, generaría un gran costo de gas. Al observar esa problemática se decide implementar el método *random* entre los números que se jueguen. Dependiendo del número de jugadores habrá más recompensa u otra, ya que, el premio será un porcentaje de la cantidad recaudada de la venta de boletos. Además cuantos menos jugadores, será menor el premio, pero habrá una mayor probabilidad de ganar. Ya que, en caso de que jugasen solo dos usuarios, la probabilidad de ganar el sorteo es del 50%, mientras que si juegan cien es del 1%. Por tanto, no puede ser el mismo premio si juegan solo dos usuarios, que si juegan cien.

Tras un exhaustivo estudio no se han encontrado referencias de sistemas similares dentro del metaverso, mientras que en el mundo *blockchain* se han encontrado referencias de un proyecto similar denominado MylottoCoin, por lo que se estima que dentro del metaverso puede tener un gran desarrollo como modelo de negocio de cara al desarrollador y más lúdico de cara al usuario.

Memoria

Necesidades y objetivos

Necesidad detectada y justificación de su resolución

Tras haber estado estudiando y trabajando en el mundo del *blockchain*, se realizó un estudio por diferentes plataformas del metaverso. Del cual, se apreció que en la mayoría había un casino con juegos de azar típicos de los casinos, pero no había una lotería dentro de dichas plataformas. A partir de esa idea se empezó a trabajar y pensar en un sistema de lotería innovador que pudiera tener cabida en el mundo del *blockchain*. Visto esta oportunidad de poder llevar una idea al mercado y sacar una rentabilidad de dicha idea, se empezó a diseñar como implementar la lotería con las tecnologías que están actualmente en auge.

Con el auge de los tokens no fungibles, se decide que cada uno de ellos sea un boleto de lotería y que su hash sea el número identificador con el que se juega. Se han estudiado diferentes métodos para elegir un ganador que sea premiado. El primer método que se ha propuesto es por aproximación, ya que, como dicho número es una cantidad muy grande, se decide que el ganador sea el boleto con el *hash* más cercano al número premiado, haciendo así que siempre haya un ganador del sorteo y haciéndolo más atractivo de cara al consumidor. Otros métodos que se están estudiando es el obtener un número aleatorio, a partir de la computación cuántica, aunque debido a la complejidad para poder anexionarlo al proyecto, se ha abortado esta propuesta. Otro de los métodos que se han propuesto es la obtención de un número aleatorio del exterior a partir del oráculo VRF de Chainlink, aunque al final el método utilizado son las librerías y los códigos de encriptación keccak256 para sacar un número aleatorio que sea el ganador.

Dado que las criptomonedas y el metaverso son entornos económicos y sociales que se encuentran en auge y viendo la cantidad de jugadores que se encontraban en los casinos del metaverso, se ha desarrollado este proyecto, con ánimo de lucro. También se ha desarrollado para que algunos de los jugadores puedan llegar a sacar una gran recompensa en forma de criptomonedas. Estas después se podrán mantener o cambiar por dinero físico, dependiendo del deseo de los usuarios ganadores. Aunque no solamente se estima que será exitosa viendo la cantidad de personas en los casinos virtuales, sino que, basándonos en la cantidad de personas que juegan a lotería físicamente, se estima que tendrá una gran popularidad y más siendo un proyecto de lotería *blockchain*, el cual, a día de hoy todavía no hay una gran cantidad de proyectos instaurados en la industria. Tras haber estado investigando, se ha podido encontrar que hay loterías como MyLottoCoin, Lotto247, LottoLand: de las cuales Lotto247 o LottoLand, son proyecto de loterías

tradicionales adaptadas a las criptomonedas y a *blockchain*, mientras que el proyecto que más se asemeja a este es MyLottoCoin, debido a la estructura y estándares que utilizan en el desarrollo. Ya que, se trata de una app con código abierto que podemos encontrar en GitHub. Aunque los métodos de juego de dicha lotería son más parecidos al Euromillón.

Requisitos técnicos y funcionales

Se requiere tener un dispositivo con conexión a red, para poder acceder a la *testnet* donde se desarrolla la lotería, así como estar *logeado* en dicha plataforma y tener activos suficientes para poder adquirir un NFT como participación para poder jugar. Además se deberá tener la cartera de criptomonedas relacionada con la plataforma en la que se desarrolla el juego para poder realizar las transacciones de compra del cupón, como en caso de ganar la recompensa en criptomonedas, poder realizar la transacción directamente a su cartera, donde se le ingresarán las criptomonedas premiadas.

Para poder instaurarlo en los distintos metaversos que actualmente hay, se debe desarrollar un *frontend* y pagar con dinero físico o criptomonedas lo que cueste la adquisición de un terreno en el metaverso, que en la actualidad, en los principales metaversos, se encuentran en unos precios muy elevados. Esto se deja para un futuro proyecto, a partir del cual, el desarrollador elija en que metaverso le conviene más desarrollar el *front* de la aplicación.

Respecto a los requisitos técnicos del ordenador en el que se ejecute, al no ser un programa muy pesado, cualquier ordenador con unas prestaciones mínimas, que puedan soportar un sistema operativo con conexión a internet podrá ejercer un uso óptimo del programa. También dependerá en parte, del desarrollo de *front-end* que ejecute el cliente y en la plataforma en la que lo acabe instaurando.

Objetivos del proyecto

El objetivo principal del proyecto es el diseño de la implementación de una lotería online con tecnología *blockchain*. En este sistema de lotería se utilizará un formato de tokens no fungibles como participaciones de la lotería.

Tras haber estado estudiando el entorno de las criptomonedas y el metaverso, se ha observado una oportunidad de lucrarse, a partir del desarrollo de una lotería. Al poder hacerse de manera descentralizada a la que puedan acceder desde todos los lugares del mundo con un criptoactivo común. Evitando así el cambio de divisas entre los diferentes países.

Este proyecto se puede llegar a convertir en un modelo de negocio dentro de las criptomonedas y en el metaverso, que son modelos de desarrollo económico y social innovadores que está experimentando un gran auge.

Bases de desarrollo del proyecto

Historia de la lotería

Para poder entender mejor el proyecto, vamos a hacer un breve paso por la historia para ver el motivo por el que se crearon las loterías, como eran en su origen y como han ido evolucionando con el paso del tiempo. Este apartado sobre todo refleja como siempre hay un mismo motivo e idea de negocio para crear una lotería y como a lo largo de los años siempre la población ha sido propensa a jugar a la lotería a cambio de obtener una recompensa fácilmente. Los motivos de nuestro proyecto son los mismos, que los que se observan lo largo de la historia, el llevarlo a un modelo de negocio, del cual, poder recaudar dinero.

La lotería surgió en el S.II a.C en China. Donde se jugaba a juegos similares al bingo. Aunque se cree que viene de China por esos juegos de azahar otros documentos datan de su origen por oriente medio, con juegos de azahar similares al del origen chino, con el objetivo de tener una recompensa en caso de que acertasen un número.

Teorías aparte, ya en la Antigua Roma existía una lotería consolidada y organizada. y durante las Saturnalia (ceremonia romana de la época en honor a Saturno) se arrojaba a la muchedumbre gran número de tablillas en las que habían sido inscritos diversos dones, premios y trofeos.

En la Roma del siglo I los juegos de azar eran una de las ocupaciones favoritas del pueblo y se utilizaban para todo. En las fiestas lupercales, entre otras actividades, hubo una lotería del amor que emparejaba a los adolescentes durante un año.[1]

En ese tipo de loterías se introducían en una caja los nombres de los muchachos y los de las muchachas en otra, y se sacaba de ambos sendos nombres, quedando así emparejados por la suerte un par de adolescentes.[1]

Los así unidos asistían juntos a las fiestas, bailes y ocasiones alegres de la ciudad. La lotería decidía el destino, el azar unía a las personas. El futuro dependía de la buena o la mala suerte.[1]

Las primeras loterías modernas surgieron en Europa hacia el siglo XV. La más antigua de la que se tiene constancia fue celebrada en 1446 por Margaret van Eyck, viuda del pintor flamenco; si bien la primera a nivel estatal fue organizada por la reina Isabel I de Inglaterra en 1567 para financiar la modernización de la armada. [1]

Así, la lotería se extendió en Europa como una forma popular de alimentar los ingresos de las arcas públicas. Emplear los beneficios de la lotería para la financiación de diversos proyectos estatales fue una práctica habitual en esta época. La llevaron a cabo los padres fundadores de Estados Unidos o diversos monarcas franceses, y sirvió para sufragar la fundación de algunas de las principales universidades de EE. UU., como Harvard, Yale, Dartmouth o Brown.[1]

En España hubo que esperar hasta 1763 para que se celebrara el primer sorteo de lotería nacional, la denominada real lotería de Madrid y sus reinos. Esta tuvo lugar en Madrid y fue creada por Carlos III. Esta consistía en la selección aleatoria de cinco números entre el 1 y el 90 y aquella persona que tuviera esa serie de números era la premiada[1]. El ministro marqués de Esquilache, trajo a Madrid desde Nápoles a José Peya, uno de los hombres más entendidos en la materia de juegos de azar, que siguió el modelo de su tierra: el de la llamada lotería beneficiata, que no era otra cosa que la actual “lotería primitiva”. Poco después, en 1771, se celebró el primer sorteo de la lotería mexicana aún bajo control español, convirtiéndose en la primera lotería del continente latinoamericano.

En ese mismo año, intervinieron por vez primera en los sorteos los niños de la doctrina del colegio de San Ildefonso, vistieron a la usanza napolitana: túnica blanca y peluca rizada.

La lotería caló tan hondamente en el pueblo que se convirtió en asunto de masas. En Madrid se jugaba a la lotería en todas las capas sociales

La lotería actual o lotería moderna, como fue designada para diferenciarla de la primitiva se remonta, en España, a la invasión francesa de 1808. Siguiendo el ejemplo mexicano, Fernando VII juntos con las cortes reunidas en Cádiz decidieron implantar esta nueva forma de lotería en el territorio ya controlado por los españoles. Así pues, a medida que las tropas francesas se retiraban, el territorio en el que se jugaba la lotería se ampliaba hasta que acabó por cubrir la totalidad del país[1].

La edición extraordinaria de Navidad tuvo que esperar poco: en 1813 ya se celebró la primera lotería navideña, aunque no fue hasta veinte años después que comenzó a realizarse con cierta frecuencia. El nombre de sorteo de Navidad no se impuso hasta 1897. Pero esta lotería no se celebraba como se hace hoy día, pues los bombos que giran y después dejan caer el número agraciado no se implantaron hasta principios del siglo XX. [1]

Bases matemáticas de la lotería

Teoría de la probabilidad

En este proyecto de lotería online en el metaverso utilizando las criptomonedas como divisas, no nos podemos olvidar de mencionar la teoría de probabilidad que es la base para poder asignar un ganador de la lotería.

Esta teoría se da bajo un fenómeno aleatorio, es decir, que bajo unas mismas condiciones iniciales, no se puede predecir cual va a ser el resultado.

La teoría de probabilidades se ocupa de asignar un cierto número a cada posible resultado que pueda ocurrir en un experimento aleatorio, con el fin de cuantificar dichos resultados y saber si un suceso es más probable que otro.[3]

En uno de los estudios de nuestro proyecto, se realizará un suceso aleatorio entre un rango definido, entre dos números. Estos números serán ser aleatorios siempre que el rango se mantenga. Por ejemplo, en una serie de números del 1 al 100, con un rango definido de 5 unidades, podrían salir como resultados válidos de los números del 13 al 18 como del 66 al 71, es decir, que los números serán aleatorios en un rango mantenido. Si hay algún número dentro de ese rango será el ganador. Dado este modelo de lotería puede haber tantos ganadores como amplio sea el rango de números premiados.

La creación de los números al crearse tanto en un oráculo de aleatoriedad como en una máquina externa cuántica siempre va a ser sobre el mismo entorno y sobre las mismas condiciones, por lo tanto, cada número tendrá exactamente las mismas probabilidades de salir ganador que otro.

Esperanza matemática de la lotería

La esperanza matemática es la relación entre el premio obtenido y probabilidad de acertar. La esperanza matemática se va a basar en Premio x Probabilidad.

Si la esperanza matemática es 1, el juego es «justo». Por ejemplo, apostar 1 euro a que una moneda sale cara o cruz, si el premio por acertar son 2 euros, y si se pierde, 0 euros. La esperanza del juego es $2 \cdot (1/2) = 1$. Entonces, consecuentemente con la teoría de juegos, podría pagar el euro para jugar o para rechazar jugar, porque de cualquier manera su expectativa total sería 0.

Si la esperanza matemática es menor que 1, el juego es «desfavorable para el jugador». Un sorteo que pague 500 euros a 1 euro pero en el que la probabilidad de acertar sea de 1 entre 1.000, la esperanza matemática es $500 \cdot (1/1.000) = 0,5$.

Si la esperanza matemática es mayor que 1, el juego es «favorable para el jugador», todo un «chollo» para el jugador. Un ejemplo sería un juego en el que se paga 10 euros por 1 euro invertido por acertar el número que va a salir en un dado, en donde hay una probabilidad de acertar es de 1 entre 6. En este ejemplo el valor de la esperanza matemática es $10 \cdot (1/6) = 1,67$ y por tanto en esas condiciones es juego «beneficioso» para el jugador.

En las definiciones formales el cero suele ser el «juego justo», y los valores negativos o positivos indican «positivo o negativo para el jugador»

Tipos de loterías en España

Lotería Nacional

La lotería nacional se creó en 1812 y fue la denominada lotería moderna. Se juega mediante la compra de cupones. Cada cupón contiene un número que es el que se juega. El número ganador se determina, mediante la extracción de cada cifra del número de un bombo diferente. Por lo tanto hay un bombo para los millares, otro diferente para las centenas otro diferente para las decenas y otro diferente para las unidades. Por lo tanto se tratan de extracciones independientes sin remplazo ni relación entre las cifras. Tras realizar la esperanza matemática, contando con una probabilidad de 0.0000016666 de ganar el premio gordo y un premio de 60000 euros la esperanza matemática es de aproximadamente 0,1. Por lo tanto es altamente desfavorable que el usuario gane. Por eso se suele decir que las loterías son un impuesto del gobierno al desconocimiento de las matemáticas.

Primitiva y Bonoloto

La Primitiva y la Bono Loto consiste en elegir 6 números del 1 al 49, además de otro número complementario y uno distinto del 0 al 9 para determinar el reintegro.[5]

Se destina a premios el 55% de la recaudación, distribuyéndose esta cantidad entre cinco categorías de premios más el reintegro.[5]

Para hallar la probabilidad de realizar la combinación ganadora, se aplica la regla de Laplace. Esta regla saca la probabilidad ganadora realizando el cociente entre los casos favorables y los casos posibles. En este caso es una combinación de 49 elementos escogidos en grupos de 6. Por lo tanto se trata de : [5]

$$C_{49,6} = \binom{49}{6} = \frac{49 \cdot 48 \cdot 47 \cdot 46 \cdot 45 \cdot 44}{6!} = 13.983.816$$

Por lo tanto ahora aplicamos Laplace de 1 combinación ganadora entre 13983816 y nos proporciona una probabilidad de 0.0000000715% de acertar la combinación ganadora. Si se aplica la esperanza matemática teniendo en cuenta que el premio de media son unos 6000000 euros, el resultado es de 0.128. Una cifra muy inferior a 1. Por lo que se antoja casi imposible acertar dicha combinación.

Hay más categoría de premios, con premios menores y con esperanzas algo mayores, pero siempre con una esperanza inferior a 1. Por lo que se encuentra el jugador en una situación desfavorable.

Gordo de la primitiva

El Gordo de la Primitiva consiste en elegir 5 números del 1 al 54 de la 1ª matriz (PRONOSTICOS) y 1 número del 0 al 9 de la 2ª matriz (NÚMERO CLAVE), para Apuestas Sencillas. Las probabilidades de ganar son las siguientes: [6]

$$\text{Boletos posibles : } \binom{54}{5} \cdot \binom{10}{1} = 31.625.100$$

La probabilidad de ganar es el cociente entre una combinación ganadora entre 31.625.100 posibles, es decir, 0.0000000316 % de acertar. Aplicando la esperanza matemática se trata de tomando como premio medio, ya que, sigue el mismo patrón que la primitiva siendo el premio el 55% del bote recaudado, unos 2500000 euros, se halla una esperanza de 0.07. Por lo tanto se antoja, también casi imposible conseguir el gordo. Se tienen otras categorías de premios, pero estos con menores premios, pero con una esperanza algo mayor que la de conseguir el gordo. Aun así el jugador siempre se encuentra en una situación desfavorable.

Euromillones

Euromillones es el sorteo europeo que reparte los premios más grandes, llegando a superar con creces los 100 millones de euros en algunos botes, pero ello hace que este sorteo tenga que tener una complicación añadida: el enorme número de combinaciones posibles.[7]

El juego consiste en acertar 5 números de una tabla de 50 (Del nº 1 al nº 50) y además acertar 2 números (estrellas) de una tabla de 9 (Del nº 1 al nº 9) . Es decir, para tener derecho al primer premio hay que acertar 7 números (5 + 2).[7]

En Euromillones se destina a premios el 50% de la recaudación, distribuyéndose esta cantidad entre 12 categorías de premios. [7]

Realizando las combinaciones y aplicando Laplace se pueden ir hallando las probabilidades de que te puedas obtener un premio u otro. Las probabilidades de obtener la combinación ganadora es la siguiente:

$$\text{Boletos posibles : } \binom{50}{5} \cdot \binom{9}{2} = 76.275.360$$

Por lo tanto la probabilidad de acertar es el cociente entre una única combinación ganadora entre 76275360 de probabilidades. El resultado obtenido es 0,000000013% de probabilidades de ganar. En este caso la esperanza matemática depende del premio, si el premio es por ejemplo de 100000000 de euros, saldría favorable de cara al usuario. Ya que por euro invertido en caso de obtener el premio obtienes una gran rentabilidad.

Quiniela y el Quinigol

La Quiniela consiste en pronosticar sobre 3 opciones, el resultado de un partido de fútbol entre 2 equipos.[8]

En La Quiniela se destina a premios el 55% de la recaudación, distribuyéndose esta cantidad entre cinco categorías más la categoría especial. [8]

En las quinielas, si hacemos una apuesta sencilla, tenemos que hacer frente a 3 elevado a la 14 de casos posibles, ya que en cada uno de los catorce partidos tenemos tres posibles resultados: 1, X, 2. Por lo tanto, hay que dividir nuestra apuesta (1) entre todas las posibilidades (3¹⁴), con lo que para llevarse el pleno hay una probabilidad de 1 entre casi 5 millones, si jugáramos sin tener en cuenta que algunos resultados son más probables que otros, debido a las diferencias entre los equipos de fútbol en juego. [8]

El Quinigol es un juego que se basa en resultados de partidos de fútbol, como La Quiniela. Pero a diferencia de ésta, en la que hay que acertar si un equipo gana, empata o pierde, en el Quinigol lo que hay que acertar es el resultado de los partidos en base al número de goles que marca cada equipo. [8]

En esta apuesta al igual que en la quiniela se destina el 55% de la cantidad recaudada a los premios. Y obtienen premio las apuestas con 6, 5, 4, 3 y 2 resultados acertados. [8]

Algunos consideran que las apuestas deportivas NO deberían figurar entre los juegos de azar ya que en las loterías (Lotería Nacional, Primitiva, Once y similares) todos los números tienen las mismas probabilidades de salir, no hay favoritos ni lesionados, todas las bolas están fabricadas con el mismo presupuesto y no existe el factor campo. Sin embargo, en las quinielas todos esos parámetros se pueden evaluar porque en el deporte hay muchísimos elementos de análisis que el apostante puede tener en cuenta. [8]

Cupón de la Once

El cupón de la ONCE es la lotería de la Organización Nacional de Ciegos Españoles, tiene distintas modalidades: Cupón diario (de lunes a jueves), Cuponazo (viernes), Supercupón fin de semana (domingos) y el último en aparecer el Combo. [9]

En este tipo de loterías se fija de antemano la cantidad establecida para los premios. En el caso del cuponazo los premios son los siguientes:

- 1 premio de 6.000.000 € a las cinco cifras y serie.
- 119 premios de 35.000 € a las cinco cifras.
- 1.080 premios de 600 € a las cuatro últimas cifras.
- 10.800 premios de 60 € a las tres últimas cifras.

- 108.000 premios de 6 € a las dos últimas cifras.
- 1.080.000 premios de 2,5 € a la última cifra (Reintegro a la última cifra).

Al igual que en el caso de la Lotería Nacional, hay que tener en cuenta las series en juego. Cada serie consta de 100.000 números comprendidos desde el 0 al 99.999 y en los sorteos especiales se entrega además un premio especial si coincide también con la serie.[8]

Por ejemplo, acertar el “Cuponazo” significa una probabilidad de 1 entre 15 millones (150 series x 100.000 números).[9]

Ventaja de una lotería blockchain frente a las loterías tradicionales

Las loterías tradicionales presentan una serie de problemáticas bastantes comunes, que se suelen dar habitualmente, sobre todo se suelen dar cuando hay una gran cantidad de dinero como recompensa de por medio. Los problemas son los siguientes:

- Hacienda: En el caso de la lotería físicas, Hacienda requisa el 20% de premios superiores a 40000 euros. Por debajo de dicha cifra, están exentos de tributación. No solamente queda aquí el pago a Hacienda, ya que al ser un beneficio en la declaración de la Renta, tendrás que tributarla y pagarás aparte en función de la contabilidad que hayas tenido en el año laboral. En el caso de las criptomonedas, se tributa sobre el beneficio que hayas obtenido en el último año laboral. Por lo que si has tenido pérdidas debido a una mala inversión y ganas la lotería tendrás que pagar menos impuestos que en el modelo físico. Si tienes unos beneficios inferiores a 6000€ deberás pagar el 19%, la franja hasta 50000€ un 21% y en adelante un 23%. La principal ventaja es si has tenido pérdidas, para contrarrestar el beneficio del premio de la lotería, en cualquier otro caso, pagará menos impuesto jugando un sorteo regulado por el estado.
- Falsificación de boletos: En las loterías *blockchain*, la falsificación de un boleto, que en el proyecto será un NFT es imposible, ya que no se puede introducir un bloque en el medio de la cadena. Además descifrar el hash de un bloque debido a la encriptación de este resulta prácticamente imposible.
- Caducidad de los premios: En una lotería física tradicional, la reclamación del premio tiene normalmente una fecha de expiración, por ejemplo en la lotería de Navidad son tres meses de plazo para reclamarlo. En cambio en una lotería *blockchain* digitalizada, el proceso de pago se realiza de forma automática, al usuario asociado al cupón ganador. De tal forma, que no tendrá que preocuparse para reclamar el premio.

- Compartición de boletos: Esta problemática surge cuando se juega “un boleto a medias”. Este común problema, con las loterías *blockchain* no se puede, ya que, va asociado el premio a un usuario y a una cartera específica, en la que se abonará el premio.
- Otra ventaja es que, al ser una lotería digital, el boleto no se puede extraviar o romper, como ocurre en ocasiones con los boletos físicos. Por lo que siempre que puedas acceder a tu cuenta, tendrás acceso

Al margen de los beneficios anteriores, la lotería *blockchain*, presentará las siguientes diferencias con el modelo tradicional:

- Un código abierto y descentralizado en vez de un negocio centralizado en una entidad determinada
- Automatización de los procesos en vez de los trámites manuales y en ocasiones el tener que desplazarte hasta una sede física para cobrar o solucionar el problema.
- Premios en función de la recaudación en tiempo real con un coste prácticamente nulo, en vez de tener un gasto de personal operativo excesivo.
- Se trata de una lotería que se puede jugar a nivel global, a diferencia de las físicas que solo se pueden jugar a nivel nacional.
- Mayor seguridad debido a los estándares y que todas las operaciones quedan registradas en el *ledger*. Por lo tanto, así se evita el blanqueo de capital.
- Se realiza el pago de premios automáticamente sin necesidad de depender de terceros factores, como el tener que visitar la sede física y depender de ellos.
-

Herramientas y tecnologías a utilizar

Blockchain

Definición

Blockchain es un tipo de libro de registros compartido e inmutable. Lo que se escribe en la cadena de *blockchain* queda registrado, garantizando la seguridad, integridad y disponibilidad. En ocasiones el contenido de la cadena se puede encontrar codificado para una mayor seguridad, garantizando la confidencialidad del registro. En esta tecnología no pueden existir dos registros idénticos, ya que cada bloque además de contener las transacciones e información contine los vínculos con los bloques posterior y anterior, por lo que tiene un puesto inamovible dentro de la cadena.

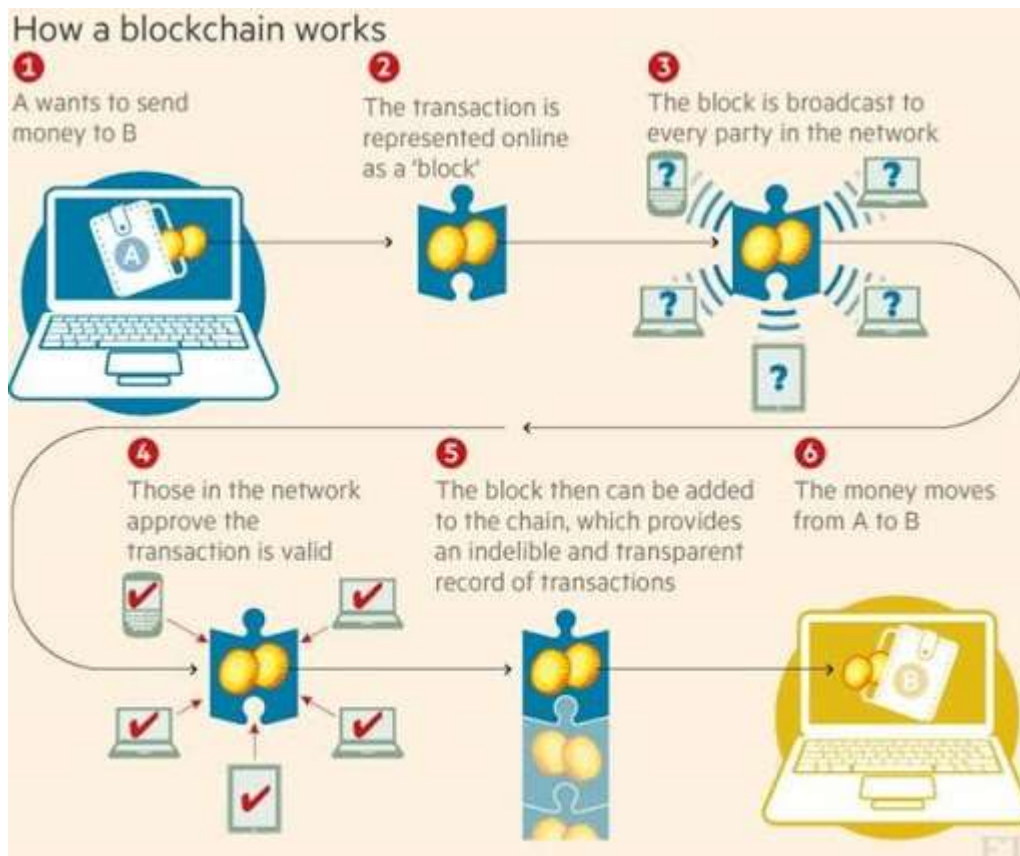
Blockchain trabaja descentralizadamente, es decir, que es administrada por distintos computadores que forman una red *peer to peer*, donde cada computador se trata de un nodo de la red. Cada uno de los computadores tiene una copia del libro de registros de las transacciones para evitar un único punto de fallo y todas las copias en cada nodo se actualizan simultáneamente [10]. Además para que se pueda proceder a realizar una transacción, esta debe estar validada por cada uno de los nodos de la red.

Cómo funciona blockchain

Dentro de *blockchain* hay dos tipos de bloques, las transacciones individuales y los bloques. Las transacciones individuales consisten en un encabezado y datos que pertenecen a las transacciones dentro de un periodo de tiempo establecido [11]. Cada bloque contiene una marca de tiempo, datos y una función 'hash'. Se trata de un identificador único para todo el contenido del bloque. Un hash es un algoritmo matemático que asigna datos de longitud arbitraria a una salida de longitud fija. Esta función devolverá siempre el mismo resultado cuando se le introduzcan exactamente los mismos datos. Si varía alguno de los datos aunque sea mínimamente el resultado de la función cambiará por completo. Esta función es unidireccional, ya que no puedes sacar ninguno de los datos a partir de la inversa de los datos. Es decir, que no se podrá hallar algunos de los factores a partir del resultado y de otro de los factores.

Una vez que se ha creado el primer bloque, cada uno de los nuevos bloques utiliza el hash del bloque anterior para calcular su propio hash. Para que un nuevo bloque pueda ser agregado a la cadena, debe ser validado por cada uno de los nodos de la red de *blockchain*. A esto se le llama proceso de validación o consenso.

Una vez que se agrega un nuevo bloque a la cadena, se considera inmutable, ya que no se puede cambiar. Si se intenta intercambiar con algún otro bloque de la cadena, los hashes de los bloques posterior y anterior también se verán modificados, por lo que interrumpirán el estado compartido del libro de registros o *ledger*. Cuando alguno de los nodos reporta un fallo y no hay un consenso en la red no se podrán agregar nuevos bloques a la cadena hasta que se solucione el problema y haya consenso entre todos los nodos. En la mayoría de los casos, el nodo causante del fallo se descarta y se repite el proceso de validación.



1. Funcionamiento del blockchain

Elementos clave de blockchain

- **Tecnología de libro mayor distribuido:** Todos los participantes de la red tienen acceso al libro mayor distribuido y a su registro inmutable de transacciones. Con este libro mayor compartido, las transacciones se registran solo una vez, eliminando la duplicación del esfuerzo que es típico de las redes de negocios tradicionales.[12]
- **Registros inalterables:** Ningún participante puede cambiar o falsificar una transacción una vez grabada en el libro mayor compartido. Si el registro de una transacción incluye un error, se debe añadir una nueva transacción para revertir el error, pero ambas transacciones serán visibles. [12]
- **Smart Contracts:** Para acelerar las transacciones, un conjunto de reglas, llamado *Smart Contracts*, se almacena en el *blockchain* y se ejecuta automáticamente. Un contrato inteligente puede definir las condiciones para las transferencias. [12] Un *Smart Contract* es un tipo especial de instrucciones que es almacenada en la *blockchain*. Y que además tiene

la capacidad de autoejecutar acciones de acuerdo con una serie de parámetros ya programados. Todo esto de forma inmutable, transparente y completamente segura[13].

Tipos de blockchain

Blockchain pública

Se trata del primer tipo de *blockchain* que existió. Este tipo de *blockchain* mantienen abierto al público sus datos, el software y su desarrollo, de forma que cualquier persona puede revisar, auditar, desarrollar o mejorar los mismos.

Para garantizar la integridad y seguridad de la red, se llevan a cabo medidas de seguridad como la tolerancia a fallos bizantinos en la programación, protocolos de consenso robustos, protecciones DDoS o contra ataques de 51% o doble gasto. [14]

La tolerancia a fallos bizantinos se trata de acatar y defender un conjunto de reglas determinadas acordadas por los actores, en este caso, cada uno de los nodos de la red [15]. Se trata de un método de seguridad complejo, ya que, no implica ninguna restricción [16]. Este método de seguridad se asocia en *blockchain* con el proceso de validación, por lo que todos los nodos deben aceptar y acatar las normas de la red. Para ello, se debe iniciar cada proceso con un estado no decidido y se debe garantizar un medio de comunicación. Una vez que se ha informado a todos los nodos estos deben pasar a un estado de decidido o no decidido y una vez decididos, se totalizan las decisiones y gana el estado con mayor cantidad de decisiones a favor.

Un ataque DDos, se trata de un ataque que inhabilita la red durante un periodo de tiempo específico. Esto se causa realizando peticiones o conexiones empleando un gran número de ordenadores o direcciones IP. Estos ataques consisten en consumir los recursos computacionales de la red para colapsar el servicio, alteración de información de configuración de la red, alteración de información de estado, tales como interrupción de sesiones TCP, obstrucción de medios de comunicación entre usuarios de un servicio y la víctima y explotación de vulnerabilidades del servicio o de partes del sistema para lograr que el mismo deje de funcionar [17].

Para realizar el ataque del 51%, basta con tener un nodo minero y tener la participación mayoritaria en la red de la *blockchain*, ya que para que se lleven a cabo las acciones en las reglas se debe tener el consenso de la mayoría de los nodos. En redes grandes se dificulta el poder realizar dicho ataque a una red, ya que poseer la mayoría de ella se antoja difícil. Un ataque de este tipo le permitiría al atacante obtener la mayor cantidad de recompensas por minerías de la red. Esto

gracias a que controla la mayoría del poder minero, y por tanto el sistema le recompensará de forma proporcional [18].

Otro caso que puede tener lugar es la capacidad del atacante de realizar ataques de doble gasto. Es decir, la capacidad de modificar el historial de la *blockchain* para recuperar monedas gastadas y tener la capacidad de volverlas a gastar. Estas dos primeras acciones son los principales motivos para controlar la mayoría del poder minero de una *blockchain*. Pues le permiten obtener ganancias de forma completamente deshonestas y nadie puede detenerlos en ese objetivo.[18]

Las características de este tipo de redes de *blockchain* son las siguientes:

- Las *blockchain* públicas permiten que cualquier persona pueda formar parte de la misma. Bien sea como usuario, minero o administrador de un nodo, las personas pueden acceder a la red y formar parte de ella sin restricción alguna. [14]
- El funcionamiento de la red es completamente transparente y abierto. Los datos de la *blockchain* desde sus inicios están disponibles para todos sin restricciones. Cualquier persona puede revisar o auditar el funcionamiento de la red y su software.[14]
- No existen entidades centralizadas. Las redes públicas son completamente descentralizadas y no existe una autoridad central que regule su funcionamiento. [14]
- El mantenimiento económico de la *blockchain* depende del sistema integrado en la misma. Generalmente este sistema económico depende de la minería y el cobro de comisiones por cada transacción que se realice dentro de la red. [14]

Las redes de *blockchain* públicas además de ser más propensas a recibir ataques, las transacciones pueden ser rastreadas en caso de que la billetera vaya asociada directamente a un usuario, ya que se podrán ver todos los movimientos que ha realizado dicho usuario.

Para mantener la integridad de la red los propios mineros piden que se les pague una comisión en cada transacción, ya que, son ellos los que aseguran la red de los ataques externos.

Otra gran desventaja de este tipo de redes es el alto consumo energético. Esta desventaja suele ser más apreciada en las redes que utilizan *Proof of Work* [18]. Una red que utiliza este protocolo es aquella que verifica cada operación antes de dar acceso al cliente a cada transacción o bloque.

Blockchain privada

Una red de *blockchain* privada generalmente tiene las mismas características y su funcionamiento es similar a las redes públicas, pero a diferencia de éstas, las *blockchain* permissionadas dependen de una unidad central que controla todas las acciones dentro de la misma. Esta unidad central es la que permite dar acceso a los usuarios, además de controlar sus funciones y permisos dentro de la *blockchain*.

Son redes convenientes para organizaciones, empresas o negocios que buscar tener un acceso limitado y privado a los registros; con el fin de mantenerlo fuera del alcance del público. Sin embargo, las operaciones o transacciones que se realizan son validadas por los mismos participantes.

Al ser redes privadas, suelen ser más pequeñas, lo que suele venir asociado a un mayor rendimiento de la red. Otra gran ventaja es que es confiable y es menos propensa a recibir ataques, ya que, todos los usuarios de la red están identificados, a diferencia de las públicas, que pueden ser anónimos. Por último a diferencia de la red pública no existen comisiones, ya que, los creadores de bloques no lo son por incentivos económicos sino por formar parte de la red, así pues no exigen una retribución en forma de criptomonedas y por lo tanto los usuarios no deben pagar comisión para usar la red [18].

La principal diferencia con la red pública es que no es descentralizada, ya que, los registros se encuentran totalmente con acceso cerrado. Y es administrado por una sola organización o entidad según corresponda.

Blockchain híbrida

En estas *blockchain*, la participación en la red es privada. Es decir, el acceso a los recursos de la red es controlado por una o varias entidades. Sin embargo, el libro de contabilidad es accesible de forma pública. Esto significa que cualquier persona puede explorar bloque a bloque todo lo que sucede en dicha *blockchain*.

Las características de estas redes son:

- El acceso a la red está restringidos a elementos que solo pueden ser autorizados por el resto de las unidades de control. [18]
- El acceso al libro de transacciones o cualquier otro medio de información generado por la *blockchain* es público.[18]
- No existe minería ni criptomonedas. El consenso de la red se da por otros medios que aseguran que los datos son correctos. [18]

Es parcialmente descentralizado lo que conlleva a un mejor nivel de seguridad y transparencia. [18].

Casos de uso de blockchain

Descentralización

La descentralización es clave en la tecnología *blockchain*, pues no hay un intermediario que todo lo vea y que además nos vaya a cobrar por ese servicio. Al final cada entidad tiene su base de datos que a la vez está conectada a su red de *blockchain* y entre las mismas entidades se validan, sin la necesidad de una entidad central. [19]

Criptografía

Gracias a la criptografía *blockchain* es seguro, programable e inmutable. *Blockchain*, por lo tanto, es una base de datos segura, que es compartida entre varias partes y es distribuida. Todo esto con el fin de registrar y permitir las transacciones de cualquier tipo.[19]

Mantenimientos de registro

- Certificados: Así mismo, las redes de *blockchain* también podrían utilizarse para asegurar que los certificados no se falsifiquen, ni se modifiquen, pues una vez dentro de la red *blockchain* estarán 100% seguros.[19]
- Seguimiento de procedencia: Todo lo que tiene que ver con denominaciones de orígenes es otro tema que se podría manejar con mucha eficiencia a través de redes *blockchain*. [19]
- Intercambios: Para realizar cualquier intercambio, ya sea, financiero o de propiedades de particulares. Debido a que las normas de intercambio quedarán grabadas en los *Smart Contracts*, evitando así conflictos entre las partes.

Automatización de flujos de trabajo

La tecnología *blockchain* puede utilizarse para optimizar los procesos y conectar en una sola base de datos toda la información necesaria para todos los participantes de la cadena de suministro. En este caso se pueden utilizar los *smart contracts*. [19]

Tokenización

La tokenización es la representación de un valor que tenemos en el mundo físico real pero en digital, en una red *blockchain*, por lo tanto, se puede *tokenizar* todo lo que se desee. Desde criptoactivos hasta inmuebles.

La tokenización se utiliza mucho para pagos internacionales, fideicomisos, bonos, micropagos y todo el tema de financiación.[19]

Colecciones e inversiones

Casas de subastas que comercian con arte han adoptado la tecnología NFT como una forma de permitir que los coleccionistas e inversores obtengan propiedad de obras de arte sin necesariamente tener posesión física de ellos. Ya sea que se trate de un JPG de gastos o una obra maestra de Picasso, ahora puedes comprar, coleccionar e invertir en NFTs basados en *blockchain* a través de plataformas de trading.[20]

Solidity

Se trata de un lenguaje de programación a alto nivel, cuyo uso está orientado a la programación *blockchain*, diseñado en la programación de los contratos inteligentes o *Smart contracts*. La sintaxis del lenguaje está basada en ECMAScript, es decir, es similar a la de C++ o JavaScript. Otras similitudes con C++ es que también es un lenguaje orientado a objetos y acepta la herencia y polimorfismos de los contratos inteligentes, que en C++ equivalen a las clases.

Fue creado en 2014 por diferentes desarrolladores del proyecto de Ethereum. Por lo tanto, este lenguaje trabaja exclusivamente con las redes de Ethereum. Aunque debido a la similitud con otras redes como la de Binance o Polygon, se podrán desarrollar contratos inteligentes en dichas redes.

Dentro del lenguaje podemos encontrar diversas librerías y estándares como por ejemplo las de Chainlink que son los estándares que se utilizan y se explican posteriormente.

Una ventaja de este lenguaje es que el código fuente de los programas que son realizados con el mismo, puede ser accedido públicamente desde la red de *blockchain*. Incluso estando en *bytecode*, es posible descompilarlo y obtener una muestra bastante clara del código fuente original.

Chainlink

Se trata de un proyecto de oráculos descentralizados, cuyo origen se remonta a 2014. Este proyecto fue desarrollado por Steve Ellis, Ari Juels y Sergey Nazarov en los orígenes sobre los *Smart Contracts*. La idea era la de crear estructuras descentralizadas capaces de crear enlaces de comunicación entre parámetros del mundo real y las *blockchain* públicas.

En 2017 los investigadores describieron los protocolos de los oráculos descentralizados que se ejecutarán en la red de *blockchain*, aunque no fue hasta el 1 de junio de 2019 cuando se lanzó la red principal de Chainlink.

Protocolo de Chainlink

Chainlink trabaja sobre una red de nodos, llamados Chainlink Nodes. Cuya finalidad es la de poder ejecutar un programa capaz de vigilar los datos provenientes de un evento en el mundo real y a partir de dichos datos poder nutrir de información a los *Smart Contracts* de la red Ethereum.

Chainlink, para asegurarse que la información es veraz, obtiene datos de diferentes nodos de forma aleatoria. Tras captar la información se genera un consenso entre las respuestas obtenidas, tomando como válida la respuesta indicada por la mayoría de los nodos. Chainlink usa la teoría de juegos, además de incentivos / desincentivos para evitar malas prácticas o manipulación maliciosa de datos. El incentivo para los operadores de nodos es dar respuestas correctas a cambio de una pequeña compensación económica. Mientras más exacta es la información, mejor es la compensación económica, por lo que se crea un ambiente en el que la certeza de datos es recompensada.[21] Por el contrario, la manipulación de datos desde los nodos acarrea lo contrario, suponiendo una especie de multa y la degradación en la confianza de dicho nodo.

Para incentivar que cada nodo realice su trabajo correctamente y poder incentivarlos a no cometer ningún fraude en la red, se utiliza el token Link. Se trata de un token ERC-20, es decir, que funciona sobre toda la red de Ethereum. Con cada nodo instalado y solicitud de información atendida de forma correcta, los nodos y sus operadores van acumulando tokens Link que son entregados en compensación por su trabajo.

Un propietario de tokens Link puede intercambiar estos tokens por otros tokens, criptomonedas o fiat. También, como en cualquier otra criptomoneda, se puede hacer HODL de estos tokens y apostar por su subida de precio. [21]

Dentro del protocolo de Chainlink, cada tarea a realizar es llevada a cabo por una serie de *Smart contracts* como se describe a continuación:

- En primer lugar, un contrato de reputación, el cual asigna una reputación a cada nodo dentro de la red teniendo en cuenta la calidad de información que ofrece.
- En segundo lugar, un contrato de coincidencia de pedidos, el cual acumula información de los nodos, los servicios que solicitan información, el tipo de información solicitada, parámetros de consulta y las fuentes de datos a analizar.

- Y finalmente, un contrato de agregación, que se encarga de recopilar todas las respuestas de los nodos, analizar los datos y ofrecer la respuesta final al solicitante de dicha información.

Privacidad en Chainlink

En Chainlink hay dos problemas fundamentales respecto a la privacidad de la información. En primer lugar, no se quiere que el oráculo tenga un acceso muy elevado a la información de los usuarios que interactúan con el mismo, junto a los servicios con los que interactúa. Y, en segundo lugar, se busca que las sesiones y la información enviada y recibida por el oráculo, no permita la trazabilidad de dichos datos desde su origen hasta el destinatario, lo que obviamente permitiría identificar plenamente a las partes que hacen uso del oráculo.

Pese a que en *blockchain* todo puede ser pseudoanónimo (o completamente anónimo), no pasa lo mismo fuera de este ecosistema. Cada vez que te conectas a Internet comienzas a dejar rastros de tu identidad, o de elementos que pueden llevar a las autoridades (o quien se interese) hacia tus datos y tu persona real [21]. Por lo tanto, esto provoca que indirectamente que la privacidad de los datos de los datos del usuario esté en un riesgo constante cada vez que se conecta a internet.

Beneficios de Chainlink

Pese a que como se ha matizado en el apartado, la privacidad es uno de los gaps que se encuentra en este proyecto y a que, es posible que en se pueda realizar un ataque a la red Sybil, que consiste en la introducción de datos falsos a través de los oráculos, manipulando la información de la red, hay una serie de beneficios que se pueden destacar.

- Es un oráculo descentralizado con gran capacidad de extensión. El protocolo puede proporcionar información de casi cualquier cosa, algo que es posible gracias a su infraestructura extensible.[21]
- El sistema está completamente descentralizado y de hecho está construido teniendo en mente un funcionamiento centrado en la seguridad y la correctitud de los datos que se otorgan.[21]
- La instalación de los nodos Chainlink es sencilla de realizar, lo que permite la masificación de la red de nodos de esta red.[21]
- El modelo económico del protocolo permite que cualquier con un nodo dentro de la red pueda obtener buenas ganancias prestando los servicios que solicitan quienes usan el protocolo.[21]
- El sistema es completamente software libre y puede ser verificado en todas sus instancias de funcionamiento.[21]

Oráculos

Definición de oráculos

Los oráculos *blockchain* son servicios ofrecidos por terceros que proporcionan a los *Smart contracts* información externa. Sirven como puente entre las *blockchains* y el mundo exterior.

Los bloques de *blockchain* y los *smart contracts* no tienen acceso a datos de tipo off-chain (es decir, datos que son externos a la red). Sin embargo, para la ejecución de muchos acuerdos contractuales, resulta vital disponer de información relevante sobre el mundo exterior.

Cabe señalar que un oráculo *blockchain* no es la fuente de información en sí misma, sino la capa que consulta (*queries*), verifica y autentifica fuentes de datos externas, y luego transmite dicha información.[22]

Tipos de Oráculos

Software: Se tratan de oráculos que extraen la información del exterior, proporcionada por un usuario. Estos datos son enviados a los *Smart Contracts*, para el que fueron programados. A partir de estos datos, el *Smart Contract*, lleva a cabo las acciones que fueron programadas anteriormente.

Hardware: Se tratan de oráculos que extraen información en tiempo real. Estos dispositivos hardware brindan a los usuarios la capacidad de monitorear cadenas de suministro completas a través de una *blockchain*.

Así, estos oráculos incluyen sensores IoT, RFID o lectores de códigos de barras. La información que capturan se lleva a un sistema de suministro basado en *blockchain*, lo que brinda a los usuarios un conjunto completo de información[22].

Entrantes y salientes: Los oráculos entrantes, son aquellos que proporcionan información desde el exterior hacia la *blockchain* o *smart contract*. Son muy empleados por empresas y compañías que desean automatizar distintas acciones que dependen de la introducción de información.[22]

Por otro lado, los oráculos salientes, son aquellos que brindan a la red la posibilidad de enviar datos al mundo real.

De Consenso: Son oráculos que se basan en el consenso para asegurar una información confiable y que no pueda ser manipulada. Estos oráculos, recogen información de diferentes fuentes, ya que, si solo la adquiriese de una única fuente, sería poco fiable e insegura.

Centralizados y descentralizados: Los oráculos centralizados son aquellos que están controlados, únicamente por una unidad central. Esto hace que si dicha unidad proporciona información errónea, se vea afectado indirectamente al *Smart Contract*, debido a que realizará las acciones programadas sobre una información errónea. Se podría decir que el *Smart Contract* depende directamente de la información que le proporciona el oráculo. El principal problema con los oráculos centralizados es la existencia de un único punto de falla, lo que hace que los contratos sean menos resistentes a las vulnerabilidades y los ataques.

Los oráculos descentralizados aumentan la veracidad de la información ya que no procede de una única fuente. El contrato inteligente consulta múltiples oráculos para determinar la validez y precisión de los datos; esta es la razón por la cual los oráculos descentralizados también pueden denominarse oráculos de consenso[22].

Los oráculos descentralizados también pueden ser útiles en los mercados de predicción, donde la validez de un determinado resultado puede verificarse por consenso social.[22]

Si bien los oráculos descentralizados apuntan a lograr la falta de confianza (*Trustlessness*), es importante tener en cuenta que, al igual que las redes *blockchain trustless*, los oráculos descentralizados no eliminan completamente la confianza, sino que la distribuyen entre muchos participantes. [22]

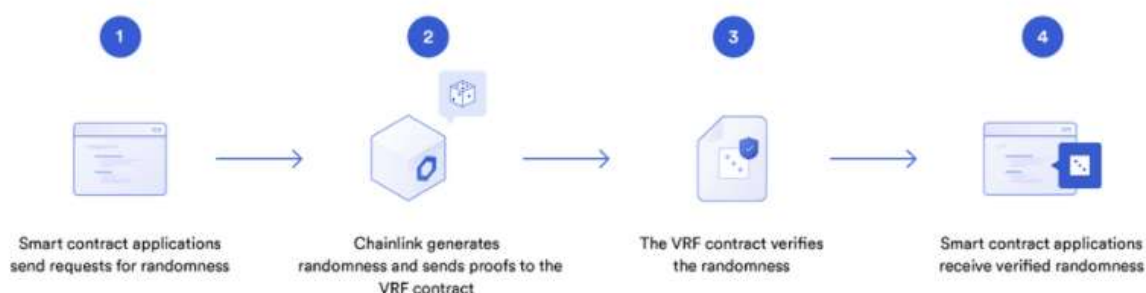
Oráculos específicos de un contrato: En la mayoría de los casos hay un oráculo, para cada *Smart Contract*, aunque hay empresas que diseñan un oráculo para varios *Smart Contracts*. Este tipo de oráculo es más costoso de desarrollar y mantener y lento a la hora de ejecutarse que uno creado específico para un *Smart Contract*. Ya que uno creado de cero y específico para ya un *Smart Contract* determinado es más flexible y optimizado a la hora de desarrollarse.

Oráculos humanos: Se trata de uno de los oráculos más seguros, ya que, la información la extraes de una persona con grandes conocimientos en un campo determinado. Esta persona a la que se consulta debe ser de confianza y en ocasiones conviene contrastar la información con otros especialistas en el mismo campo.

Números aleatorios

Se han estudiado diferentes métodos de extracción de un boleto aleatorio para el sorteo. Aunque al final debido a los costes en Wei, se realizará la obtención del boleto premiado mediante el módulo del hash del bloque en el que se realiza el sorteo. Las distintas metodologías de estudio para la extracción de un boleto aleatorio son las siguientes:

Oráculo de aleatoriedad VRF de Chainlink



2. Como trabaja el VRF

El oráculo de aleatoriedad de Chainlink VRF es necesario para muchos desarrolladores de *Smart Contracts* que quieran generar valores aleatorios imparcialmente. Esto llevado a cabo en nuestro proyecto, es útil a la hora de poder comprar los NFT, asignarle a cada uno un número aleatorio dentro de un rango y para conseguir sacar el premio entre los números jugados.

El VRF de Chainlink está diseñado específicamente para los contratos inteligentes y ofrece la fuente de aleatoriedad más segura a la hora de generar NFTs únicos, experiencias dinámicas en juegos y otras características dentro de diversas aplicaciones. El VRF de Chainlink funciona combinando los datos del bloque (que aún se desconocen cuando se realiza la petición) con la clave privada del oráculo para generar tanto un número aleatorio como una prueba criptográfica. Esto garantiza el alto nivel de integridad del número aleatorio que se obtenga porque la prueba criptográfica solo puede generarse si el proceso del VRF es a prueba de intentos de manipulación. De este modo, los usuarios pueden estar seguros de que cualquier aplicación que obtenga la aleatoriedad a través del VRF de Chainlink funcionará de forma justa. [23]

Estos oráculos se utilizan para defender con éxito un contrato inteligente contra los adversarios que buscan robar los fondos retenidos por ese contrato. Los desarrolladores de contratos inteligentes que utilizan la aleatoriedad como entrada clave también deberían ver la manipulación de esa aleatoriedad como un riesgo crítico.

Las compensaciones clave y riesgos de seguridad que creemos que los desarrolladores de contratos inteligentes que consideran una fuente de aleatoriedad para el contrato inteligente deben tratar de evitar incluyen los siguientes:

- Inaccesibilidad o incompatibilidad con contratos inteligentes
- Manipulación por el operador centralizado del generador de números aleatorios
- Explotación por parte de los mineros de una cadena de bloques como uno de los usuarios de la aplicación
- Requisitos de confianza irrazonablemente altos de los usuarios finales de la aplicación

Chainlink VRF funciona mediante la combinación de datos de bloque que aún se desconocen cuando la solicitud se realiza con la clave privada precomprometida del nodo de Oracle para generar un número aleatorio y una prueba criptográfica. Cada oráculo usa su propia clave secreta cuando genera aleatoriedad. Cuando el resultado se publica en la cadena junto con una prueba, se verifica en la cadena antes de enviarse al contrato de un usuario. Confiar en las capacidades de verificación de pruebas y firmas ampliamente aceptadas de una cadena de bloques permite que los contratos consuman solo aleatoriedad que también ha sido verificada por el mismo entorno en cadena que ejecuta el contrato en sí.[24]

El beneficio fundamental de usar Chainlink VRF es su aleatoriedad verificable. Incluso si un nodo se ve comprometido, no puede manipular ni proporcionar respuestas sesgadas: la prueba criptográfica en cadena fallaría. El peor de los casos es que el nodo comprometido no devuelva una respuesta a una solicitud, que será visible de inmediato y para siempre en la cadena de bloques. Los usuarios ya no confiarían en los nodos que dejan de responder y/o no proporcionan aleatoriedad con una prueba válida. Incluso en el escenario improbable de que un nodo se vea comprometido, su aleatoriedad resultante no se puede manipular. Los nodos comprometidos solo pueden retener una solicitud, sin dar respuesta, por lo que serían penalizados financieramente utilizando las próximas capacidades de *staking* de Chainlink y eliminados de consultas futuras que habrían pagado tarifas por su aleatoriedad.[24]

Un beneficio adicional de Chainlink VRF es que, a medida que más usuarios lo utilizan, aumentan las tarifas de usuario pagadas a los operadores de nodos, lo que crea un incentivo para que los operadores de nodos brinden tantas garantías de seguridad como sea posible. La seguridad cripto-económica a través del *staking* probablemente sea algo que los usuarios pidan aumentar con el tiempo y lo justifiquen con sus tarifas cada vez más altas, que pagan por seguridad adicional. Esto conduce a un recurso global compartido que está respaldado por las tarifas de los usuarios y permite a los usuarios adicionales que habrían gastado fondos en crear su propia solución RNG implementar esos mismos fondos para aumentar la seguridad colectiva de un recurso compartido para todo el ecosistema de *blockchain*. [24]

La clave secreta VRF es un número que el oráculo elige de la distribución uniforme en $\{0, \dots, \#secp256k1-1\}$, de forma criptográficamente segura. (*secp256k1* es la curva elíptica utilizada por la criptografía de ethereum). Asociada con esta clave secreta hay una clave pública, que se utiliza para identificar el oráculo. El oráculo registra la clave pública con la maquinaria en cadena, junto con el identificador de trabajo de enlace de cadena al que responderá.[24]

Cuando un contrato de consumo realiza una solicitud de aleatoriedad, transmite un registro de Ethereum solicitando la salida VRF correspondiente del oráculo especificado en la solicitud del consumidor. Al ver este registro, el oráculo construye la salida de la siguiente manera:[24]

Primero, " codifica la entrada a la curva ", obteniendo una muestra aleatoria criptográficamente segura de secp256k1, utilizando los datos del bloque impredecible y la clave pública del oráculo como entradas. Hace esto mediante el hash recursivo de las entradas usando keccak256 hasta que la salida es menor que el orden del campo base de secp256k1 (" p " en esa descripción de secp256k1), y es la ordenada x de algún punto (x,y) en secp256k1 (es decir, $y^2 = x^3 + 7$ en el campo base). (x,y) es entonces el hash de la entrada a la curva. [24]

Luego multiplica (x,y), como un punto de la curva secp256k1 por su clave secreta para obtener un punto γ . El hash keccak256 de γ , como uint256, es la salida VRF. Genera una prueba de que γ es el mismo múltiplo de (x,y) que la clave pública del oráculo del generador secp256k1. Esta prueba es muy similar a una firma de Schnorr : Primero, muestrea de manera segura un nonce n de $\{0, \dots, \#secp256k1-1\}$, como lo hizo con su clave secreta, luego calcula $u = n \times g$, donde g es el generador secp256k1 y toma la dirección Ethereum de u . Luego calcula $v = n \times (x,y)$. A continuación, hash juntos (x,y), su clave pública VRF, γ , la dirección de u, y v , toma el resto de ese módulo hash $\#secp256k1$ y lo llama c. Finalmente, calcula $s = nc \times k$ módulo $\#secp256k1$, donde k es su clave VRF secreta. La prueba es entonces su clave pública, γ , c , s , y su semilla de entrada. Envía esto de vuelta a la maquinaria VRF en cadena, que verifica la prueba , y envía la salida de vuelta al contrato de consumo , suponiendo que la prueba se verifique. [24]

Para verificar la prueba, el contrato sería idealmente:

- Compruebe que la clave pública y γ son puntos válidos secp256k1 [24]
- Verifique que la dirección del punto $c \times pk + s \times g$ coincida con la dirección de u , (donde pk es la clave pública de Oracle)[24]
- Calcule el "hash para curvar" (x,y) a partir de la clave pública y la semilla, y verifique que el hash de h, su clave pública, γ , la dirección de u y $c \times \gamma + s \times (x,y)$ coincide con c .[24]

Aplicabilidad en nuestro proyecto

Nuestro proyecto al ser un sistema de loterías *blockchain* utilizando como activos las criptomonedas. Se podrían utilizar los oráculos VRF, tanto para asignarle los hashes a cada uno de los NFTs, como a la hora de elegir el ganador entre los diferentes participantes. Se tomará cada uno de los tokens no fungibles como una participación de la lotería. Aunque debido a los costes como se indica anteriormente, se realizará mediante el módulo del hash del bloque en el que se realiza el sorteo.

Computación cuántica

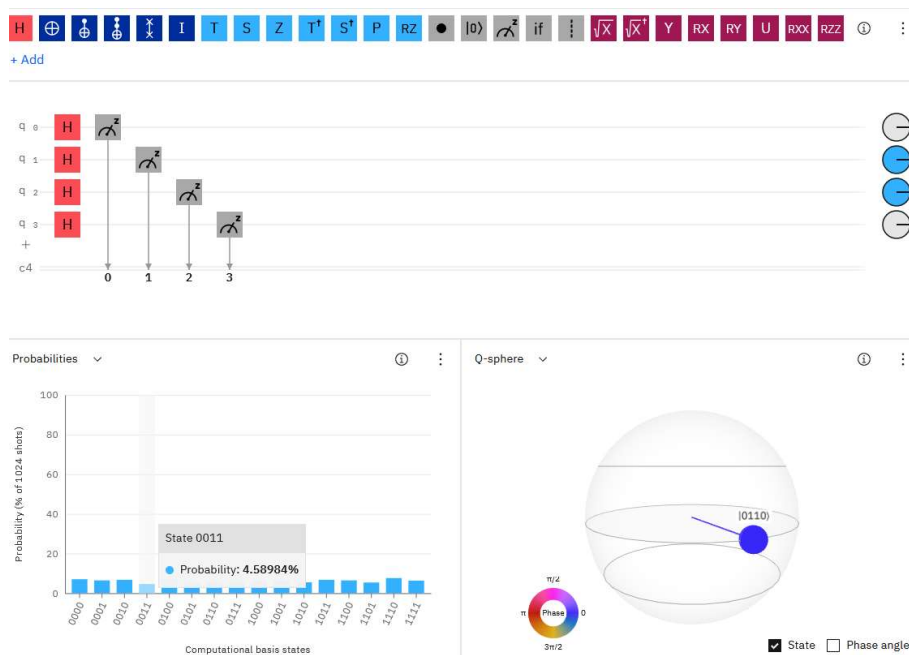
Otra metodología estudiada, fue la extracción de un número mediante la computación cuántica.

La computación cuántica se basa en los principios de la superposición de la materia y el entrelazamiento cuántico para desarrollar una computación distinta a la tradicional.[25] A diferencia de la computación tradicional o actual donde cada bit puede estar en un estado discreto y alternativo a la vez, la unidad fundamental de almacenamiento es el bit cuántico o qubit, donde cada bit cuántico puede tener múltiples estados simultáneamente en un instante determinado, así reduciendo el tiempo de ejecución de algunos algoritmos de miles de años a segundos.

En la computación tradicional, un bit es la mínima unidad de información pero, para representarlo, se utiliza la ausencia o la presencia de miles de millones de electrones en un diminuto transistor de silicio.

La computación cuántica pretende utilizar un principio básico de la mecánica cuántica por el cual todas las partículas subatómicas (protones, neutrones, electrones, etc.) tienen una propiedad asociada llamada spin. El spin se asocia con el movimiento de rotación de la partícula alrededor de un eje. Esta rotación puede ser realizada en un sentido, o el opuesto. Si por ejemplo tomamos como bit al spin de un protón, podemos usar una dirección como 1 y otra como 0. Estos bits, tomados a partir del spin de las partículas han recibido el nombre de qubits. Por ejemplo, si se toman dos bits, sus estados posibles son cuatro: 00, 01, 10, 11. Son necesario cuatro pares de bits para representar la misma información que un solo par de qubits con comportamiento ambiguo.

La aplicabilidad en el proyecto de la computación cuántica se debe a la extracción de números aleatorios, ya que dependiendo en el entorno que estén los qubits, pueden sacar un valor u otro. Haciendo casi imposible que el entorno sea exactamente el mismo y así mismo añade dificultad, en caso de que se quiera averiguar el bloque que saldrá premiado.



Circuito de aleatoriedad formado en un simulador de computación cuántica

Aplicabilidad en nuestro proyecto

Se ha estudiado la opción de realizar la extracción del boleto ganador mediante computación cuántica. Como se observa en la imagen superior, un qubit se comporta de forma diferente dependiendo en el estado que se encuentre. Al igual que se observa que en realidad, no todos los números tienen las mismas probabilidades de salir. Lo que se estudió es que mediante un circuito poder realizar las extracciones de los números aleatorios, aunque se descartó esta idea debido a la complejidad de integración con el resto de los componentes.

Librerías de Solidity

También podemos sacar números aleatorios a partir del hash del bloque cifrándolo con las librerías predefinidas que nos proporciona Solidity. Para poder cifrar el hash del bloque en nuestro proyecto utilizamos el keccak256. Esta librería proviene de las librerías SHA3. Las librerías keccak son las últimas librerías de seguridad instauradas en *blockchain*. Estas librerías se basan en la captación de un conjunto de datos finito. Este conjunto de datos pasa a través de unas puertas XOR y mediante unas funciones SHA3, comprobando la autenticidad de ese conjunto de datos.

Si el conjunto de datos introducido y cifrado con la librería sufre una mínima modificación, la salida que tendrá será totalmente diferente y aleatoria. Por lo tanto, en nuestro proyecto jugamos con eso para sacar un número aleatorio, modificar la dirección del bloque y a partir de ahí realizar el módulo con el número máximo de boletos producidos.

Este ha sido el método utilizado debido a los costes de los otros métodos, tanto de gas en el caso de los oráculos, como la complejidad de introducir la computación cuántica y los medios que requieren en un proyecto llevado a una mayor escala.

NFT

Un NFT o Token No Fungible es un certificado digital de autenticidad que mediante la tecnología *blockchain*, la misma que se emplea en las criptomonedas, se asocia a un único archivo digital. Cada uno de estos tokens son exclusivos, indiviso, transmisible y con la capacidad de demostrar su insuficiencia. Estos tokens, al ser no fungibles, no son reemplazables.

Los NFT se ejecutan por medio de la tecnología *blockchain* o de cadena de bloques. Es igual a la tecnología de las criptomonedas, que trabajan mediante una red de ordenadores transferidos, con bloques o nodos enlazados y asegurados usando criptografía. Cada bloque se junta a un bloque anterior, así como una fecha y datos de transacciones, y por diseño son resistentes a la modificación de datos.

A los NFT, se les otorga una especie de comprobante digital de validez, una serie de metadatos que no se van a poder cambiar. En estos metadatos se asegura su veracidad, se guarda el valor de partida y todos los logros o transacciones que se hayan hecho, y también a su autor.

Los NFTs aunque son únicos y no se pueden cambiar directamente con un token idéntico, el dueño de un NFT puede decidir venderlo contra una criptomoneda regular. Esto llevado a cabo en el mundo real (No virtual), se asemejaría a comprar un artículo coleccionable único. Esa exclusividad de ser único es lo que realmente le da valor a dicho producto.

Si el precio ofrecido por el comprador potencial coincide con la proyección del vendedor, este último puede aceptarlo y recibir el monto respectivo en criptomonedas. La venta se completa una vez que el vendedor ha transferido el NFT a la billetera *blockchain* del comprador.



3. Ejemplos NFTS

Tecnología blockchain Ethereum orientada a los NFTs

Como se describe anteriormente, cada NFT es único y no se puede compartir ni dividir entre varios participantes. Esto es debido a que representa un producto único digital.

Un NFT se controla a través de un identificador único y con metadatos que no puede replicar otro token. Cuando un NFT se crea a partir de un *Smart Contract*, se le asigna un propietario y se gestiona el proceso de transferencia de ese NFT al usuario. Este proceso se trata de la creación de un bloque, donde se valida la información a través de distintos estándares y por último se graba la información en la red de *blockchain*.

Para demostrar que has comprado un NFT se puede realizar a través de la clave pública del creador del token, además el propio identificador del token y la clave privada del comprador demuestran que ese NFT es el original.

En caso de que se desee crear un NFT se debe demostrar que eres el creador de dicho NFT, se debe determinar cuántas réplicas se van a vender, cada vez que se venda ese NFT, el creador va a recibir un incentivo, es decir, una comisión en el precio de la reventa y saber que no se necesitan intermediarios para poner a la venta un NFT, es decir, que se puede poner a la venta en cualquier red *Peer to Peer*.

La seguridad con los NFTs es importante por ese motivo, cuantos más bloques tenga la cadena de *blockchain* donde se almacenan los NFTs, más segura es la cadena. Si el NFT se creó en el bloque n.º 350 y un hacker intentara robar su NFT modificando sus datos, la huella digital de todos los

bloques posteriores cambiaría. Eso significa que cualquier persona que ejecute el software Ethereum podría detectarlo de inmediato y evitar que suceda.

La mayoría de los NFT se construyen utilizando un estándar consistente conocido como ERC-721. Sin embargo, existen otros estándares, con lo que también se pueden crear NFTs. El estándar ERC-1155 permite tokens semifungibles que son particularmente útiles en el ámbito de los juegos. Y más recientemente, se ha propuesto EIP-2309 para hacer que la acuñación de NFT sea mucho más eficiente, debido a que en una transacción se pueden crear varios NFTs.

Criptomonedas

El origen de las criptomonedas se basa en la criptografía y de la generación de una clave pública y privada. La clave pública es visible para todo el mundo y sirve como comprobante para saber que quien lo ha firmado es quien dice ser, mientras que la clave privada es aquella que solo la puede ver quien ha realizado una transacción. Un ejemplo sería una transacción bancaria, el número de cuenta se asemejaría a la clave pública mientras que el pin de la tarjeta para realizar una operación sería la clave privada.

Los conceptos de claves públicas y privadas nacieron bastante antes de las criptomonedas, aunque son la base de estas. En 1976, Diffie y Hellman, crearon la pareja de claves. Para que este sistema sea lo suficientemente robusto contra ataques de criptoanálisis, las claves han de ser de una longitud mínima de 1024 bits, siendo recomendable, en los casos que sea posible, utilizar claves de 2048 bits. El algoritmo RSA, además de permitir cifrar con la clave pública y descifrar con la privada, permite cifrar con la clave privada y descifrar con la pública. Este modo de cifrado no proporciona confidencialidad ya que cualquiera puede descifrar un mensaje cifrado con una clave secreta al poder obtener siempre la componente pública de su interlocutor, sin embargo el hecho de cifrar un mensaje con la clave secreta de un usuario implica una identificación del usuario al igual que lo hace una firma, por lo que este proceso se conoce con el nombre de firma digital.

Aunque el origen de las criptomonedas como se conocen a día de hoy data de 2008, con el origen del Bitcoin. Cuando Satoshi Nakamoto crea un sistema de dinero electrónico. Este sistema tenía una estructura *peer to peer*, es decir, que todos los nodos están en igualdad y consecuentemente es una estructura descentralizada. Con ello consiguió la anulación de doble gasto de comprobación, evitar terceros de confianza o emisores de monedas y todos los integrantes de la red eran anónimos y estaban bajo distintos seudónimos.

El Bitcoin tiene conserva la misma estructura que en 2008, tratándose de una red descentralizada con n nodos. Cada nodo tiene una copia del histórico de todas las transacciones que se han realizado en la red y cada transacción se tiene que aprobar por todos los nodos de la misma.

Una de las principales ventajas del Bitcoin es el de tener una base de datos descentralizada. Cada actualización a la base de datos distribuida contiene un conjunto de transacciones, agrupadas en bloques. Estos bloques se relacionan entre si a través de la técnica criptográfica ($\text{Hash}(n-1)$). Es decir, cada bloque contiene un hash, que es la cantidad de datos de tamaño variable, normalmente de gran tamaño, que es resumido en una huella digital electrónica de tamaño fijo. Por lo tanto aplicando la función criptográfica nos encontramos que el bloque n contiene el hash del bloque anterior. Con cambiar un bit de un hash de un bloque al ir encadenados se modificaría drásticamente la huella digital y los hashes de los siguientes bloques de la cadena.

Para que los mineros, que son los creadores de los bloques, puedan seguir intentando conseguir más bloques, se aplica un protocolo de consenso denominado Nonce. Estos bloques se descubren acertando el hash del bloque n , que es menor o igual que el hash objetivo. Estos hashes se realizan sacando números al azar hasta descubrir el hash.

Dado que la minería es cada vez más complicada, es casi imposible ver unidades «hash por segundo» en las redes modernas de *blockchain*. Hoy en día, se necesitan dispositivos con mayor potencia para resolver estos problemas que van desde decenas de *megahash* por segundo.[31]

Por ejemplo, una potencia de procesador de 10 MH/s puede generar 10 millones de combinaciones diferentes de números en un segundo para encontrar el hash que coincida con todos los parámetros establecidos por la red.[31]

Sin embargo, existen diferentes factores que determinan la tasa de hash. Incluso la elección de un algoritmo de minería afecta al parámetro. También es importante saber cómo reaccionan otros dispositivos con diferentes algoritmos. Mientras que algunos de estos proveen la máxima capacidad con redes que presentan un algoritmo de SHA, el resultado puede ser inferior cuando la misma red emplea el algoritmo de Script.[31]

En caso de que dos mineros descubriesen un hash en el mismo momento la cadena de bloques se bifurcaría. En este caso se escogería siempre la cadena más larga, es decir que si dos mineros obtienen el hash del bloque dos a la vez se generaría una cadena con el bloque 2 y otra con el bloque 2', pero si en la cadena del bloque 2 se descubre un bloque 3 antes que en la cadena 2', se deshacen las transacciones de la cadena más corta y se unen a la otra cadena.

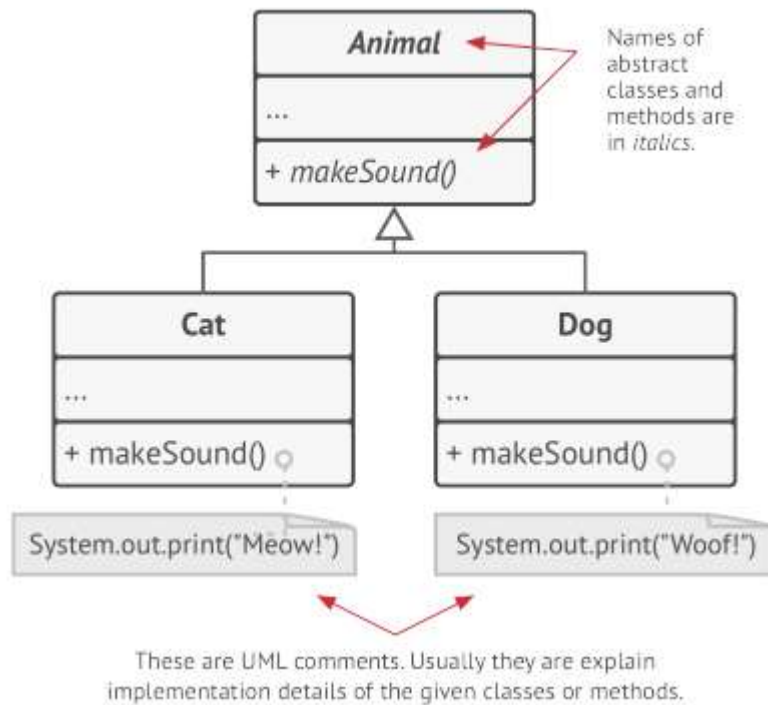
OpenZeppelin

OpenZeppelin es una librería del lenguaje de programación Solidity. Esta librería ha sido desarrollada sobre una base sólida de código examinado por una comunidad de programadores de Solidity. Como la mayoría de las librerías en los diferentes lenguajes de programación, nos permite la reutilización de código, con solamente importar las librerías correspondientes.

Estas librerías nos permiten:

- Realizar esquemas de códigos sencillos y flexibles basado en los diferentes roles de usuarios.
- Desarrollar componentes de Solidity reutilizables para poder desarrollar *Smart Contracts* de forma personalizada y complejos sistemas de manera descentralizada.
- Implementaciones de estándares como el ERC720, ERC 721 y ERC-1155 que son los utilizados en el proyecto.
- Controlar quien puede realizar alguna acción o modificación en nuestro programa y que acciones puede realizar.
- Se proporcionan herramientas genéricas, que incluyen matemáticas sin desbordamiento, verificación de firmas y sistemas de pago sin confianza.

Los contratos que se han desarrollado con estas librerías utilizan la herencia. Esto se utiliza cuando de una clase padre dependen otras clases. Es decir, que estas clases hijas tendrán los métodos del padre, pero también podrán tener los suyos propios. Dentro de la herencia se encuentra el polimorfismo, que es una propiedad de ella. Con el polimorfismo, un método en la clase padre puede tener diferentes funcionalidades en las clases hijas. En la siguiente ilustración se puede ver un ejemplo de este.



4. Herencia y polimorfismo

Hooks

En ocasiones para extender un contrato principal, se deberán anular varias funciones relacionadas, lo que conduce a la duplicación de código y a una mayor probabilidad de errores.

Para evitar ese tipo de situaciones se crearon los *hooks*, que se tratan de funciones que se llaman antes o después de que se lleve a cabo alguna acción. Proporcionando un punto centralizado para conectar y ampliar el comportamiento original.

El uso de *hooks* conduce a un código más limpio y seguro, sin tener que depender de una comprensión profunda de las partes internas y de los métodos de los padres.

Para evitar fallos en el uso de estos se deben seguir la siguiente regla:

- Cada vez que anule el *hook* de la clase padre, se debe volver a aplicar el atributo virtual al *hook*. Eso permitirá que los contratos secundarios agreguen más funcionalidad a este.

Librerías que se utilizan

ERC-20

Un *Smart Contract* importando la librería ERC-20 realiza un seguimiento de los tokens fungibles, cualquier token es exactamente igual a cualquier otro token; ningún token tiene derechos o comportamientos especiales asociados con ellos. Esto hace que los tokens ERC-20 sean útiles para cosas como un medio de cambio de moneda, derechos de voto y participación. [34]

Para sacar partido a esta librería utilizaremos la herencia, para aprovechar ya los métodos de la clase padre que viene implementada en ella, tanto para una implementación más básica como para las excepciones.

Una curiosidad del lenguaje de programación de Solidity, es que no se puede trabajar con números decimales, por lo que, en principio no se podría ejecutar transacciones tan exactas como demandan los consumidores. Para ello se proporciona en esta librería un método preestablecido denominado `decimals`, que nos permite rellenar un input para saber con cuantos decimales se va a realizar la transacción. Así ya, podremos realizar una transacción con decimales.

En el *Smart Contract* que se ha desarrollado para la creación del token, se otorga la posibilidad de elegir el número de tokens que se van a crear, así como el nombre y la iniciales del mismo. Además dependiendo de la red en la que se haya introducido el token, se podrá mirar el contrato de creación. de este, como las operaciones que se han realizado con este. En mi caso, está en la red de la empresa y se puede observar en el Epirus Explorer con el número de contrato y estando vinculado con la cuenta de la empresa. En esta web se pueden ver los detalles del token como aparece en la ilustración inferior.

JaimeTello1

 Contract	0xcebb038b46f55bd544c6426b06bb4bc7d770256
 Total supply	100 JTS1
 Holders	2 Addresses
 Transfers	2 Transfers
 Decimals	18
 Token type	ERC-20

ERC-721

En situaciones como bienes raíces , derechos de voto o coleccionables , donde algunos artículos se valoran más que otros debido a su utilidad, rareza, etc. ERC721 es un estándar para representar la propiedad de tokens no fungibles , es decir, donde cada ficha es única.

Esta librería se utiliza para poder observar los distintos tokens con sus atributos, que serán únicos. Estos tokens únicos se podrán conservar o se podrán intercambiar como si fueran otro tipo de divisa más. Para restringir que cuentas pueden tener cada uno de los NFTs y si los pueden almacenar podemos configurar accesos de control a los NFTs.

A diferencia de los tokens fungibles, y la librería ERC-20, con esta librería no podemos trabajar directamente con los decimales, ya que cada activo es único y no se puede particionar.

Además esta librería contine las extensiones estándares de metadatos IERC721Metadata, así como un mecanismo de metadatos para cada token. De ahí viene el método `_setTokenURI`, que se utiliza para almacenar los metadatos de un elemento. La información de cada elemento está incluida en los metadatos, no en la cadena. Por lo tanto, cambiando las reglas de la cadena, se pueden modificar los metadatos subyacentes.

ERC-1155

El estándar ERC-1155, se basa en la combinación del ERC20 , ERC721 y ERC777, con el fin de un ahorro de gas y que sea independientemente fungible. Es decir, que ese token, ya sea una moneda o un NFT se pueda intercambiar individualmente por dinero o por otro activo.

En comparación con el estándar ERC-721 incluye la función `_balanceof` que es capaz de comprobar el balance de la cuenta con al que vas a interactuar. Esto para proyecto con múltiples tokens, ya que, te permite implementarlos todos en un mismo contrato, por lo que supone el tener un gran ahorro de gas. Otra diferencia con el ERC-721 son las funciones `balanceOfBatch` y `safeBatchTransferFrom` que al igual que el `_balanceof` que hacen que la consulta de múltiples saldos y la transferencia de múltiples tokens sea más simple y menos gaseosa.

En nuestro caso con estas funciones podemos transferir los premios a la cuenta del ganador y obtener la comisión que nos quedamos nosotros.

Además con este estándar podemos personalizar los boletos de las diferentes loterías, para que se distinga una lotería de otra, a través de la función `uri`, pasándole, la dirección del boleto.

Otra de las ventajas que tiene utilizar este token es que contiene la función `safeTransferFrom`, que permite que las transferencias a contratos externos puedan revertirse. Además evita también que se llamen desde funciones sin ningún tipo de funcionalidad y que puedan hacer que se quede bloqueado el contrato entero.

Tras haber estado estudiando el desarrollo de la lotería con una combinación de los estándares ERC-20 y ERC-721, hemos decidido por sencillez y por los métodos que ya vienen implementados en el ERC-1155 elegir esta última. Ya que también como hemos indicado anteriormente hemos podido observar que era el modelo más eficiente respecto a la mayor problemática del proyecto que son las pérdidas en el consumo de gas.

Para asegurarnos que los métodos que se ejecutan en la interfaz están llevando a cabo las funciones correctamente utilizamos la interfaz IERC-165. Este sirve para estandarizar la versión de contrato que se va a utilizar con el estándar utilizado, haciendo que sea posible que el estándar soporte la interfaz que implementa. Además, se utilizan las interfaces asociadas al propio estándar ERC-1155, como son la IERC1155, IERC1155MetadataURI. Por lo que el estándar ERC-1155 implementa dichas interfaces y éstas se combinan con el IERC-165 para que pueda soportar dichas interfaces y se pueda ejecutar el programa correctamente.

Líneas futuras

Entorno front-end de desarrollo del proyecto

Metaverso

El metaverso es una red online persistente y masivamente escalable de mundos virtuales interconectados, diseñada para la interacción en tiempo real, donde la gente puede trabajar, interactuar socialmente, hacer negocios, jugar e incluso crear. Utiliza tecnologías y virtualización avanzadas (RA, RV, sensores hápticos, etc.) para sumergir totalmente al usuario en el mundo virtual. Esto significa que el usuario puede interactuar en directo con un mundo que siempre está ahí y al que puede acceder cuando quiera.[36]

Este modelo que sigue en vías de desarrollo ha ido evolucionando y grandes compañías del sector informático como Meta (antiguo Facebook) o Microsoft han desarrollado su propia red en vistas al futuro, como un modelo de negocio económico y social.

Esta red se ha ido desarrollando desde Web1 que la interacción solo era a través de emails y formularios, pasando por Web2, en la cual, ya se podía realizar una interacción social, a través de plataformas web, permitiendo al usuario elegir, que hacer, cuando y como. Por último se sigue desarrollando la Web3, la cual, se basa en su predecesora e integra los mundos del metaverso y criptomonedas, así como la relación entre diferentes mundos virtuales. Este desarrollo no solamente ha sido a nivel de software, sino que, a nivel hardware, se necesita una mejor conexión de red y sobretodo unas GPUS mucho más potentes para poder procesar toda la información.

Dentro del metaverso se siguen trabajando en aspectos como puede ser un marco jurídico y legal, en el cual, se establezcan unas normas y en caso de infringirlas haya unas penalizaciones, que puedan llegar a ser tomadas en el mundo físico. Esto va relacionado con el tema de las estafas en los pagos y cobros en estas plataformas y con la relación de identidades entre el metaverso y el mundo físico real. Ya que, se pueden realizar estafas en el metaverso, con falsificaciones de identidades y realizando transacciones a cuentas falsas, sin obtener ningún servicio o producto a cambio. Estas transacciones se realizan en distintos tipos de criptomonedas.

Áreas de aplicación del metaverso

- **Plataforma (mundo) para la interacción social:** Los usuarios pueden ver a otros a través de espacios y mundos virtuales, interactuar entre ellos y también celebrar reuniones sociales. Este enfoque lleva a los medios sociales al siguiente nivel, ya que pasa de compartir información de forma asíncrona a una combinación de interacción asíncrona y síncrona (en directo).[36]

- **Mercados y modelos de negocio digitales:** Esto también permitiría la creación de mercados totalmente digitales y transacciones puramente digitales en el metaverso. Un ejemplo serían las subastas, que podrían vivirse desde cualquier parte del mundo. Esto también permitiría nuevos modelos de negocio digitales disruptivos que prevén un mundo totalmente digital y la monetización de esos mundos digitales.[36]
- **Espacios para el arte y la cultura:** La creación de equivalentes digitales de eventos, museos o exposiciones de arte en Internet podría permitir a muchas más personas de todo el mundo consumir arte y cultura de una forma completamente nueva. Además se podrían crear nuevas obras artísticas a través de los Tokens No Fungibles, dando más libertad a la gente para innovar, diseñar y crear nuevas obras de arte.
- **Entornos de trabajo aumentados y virtuales:** En lugar de ver a las personas en una pantalla con unos bloques de vídeo, se podría estar en el mismo espacio virtual, haciendo una lluvia de ideas, escribiendo en una pizarra e incluso cambiando el espacio según sea necesario. Los espacios de trabajo aumentados combinarían estas funciones y permitirían a las personas participar virtualmente en una reunión física. Esto significaría tener hologramas en la sala, experimentando las personas y los avatares al mismo tiempo, y pudiendo interactuar como si estuvieran allí mismo.[36]
- **Educación y escuelas:** En este sector también es de gran ayuda el metaverso, ya que, se podría interactuar sobre algunos conocimientos en realidad aumentada. Un ejemplo sería el poder explicar el Sistema Solar a los niños, se realizaría de manera más amena y fácil, pudiendo verlos en una realidad aumentada, dando una sensación de estar en el espacio u otro planeta.

Críticas al metaverso

Aunque tiene una infinidad de virtudes el metaverso, también surgen algunas dudas. La principal cuestión por resolver es el tema de la privacidad de la información y el uso de ella por parte de la empresa desarrolladora. Otra crítica que surge es por las posibles falsificaciones de identidad, ya que, realmente estás interactuando con avatares u hologramas, por lo tanto no sabes quien está controlando dicho avatar u holograma realmente.

Otro problema es que la empresa desarrolladora tendría todo el poder sobre el mundo virtual. Significando que podría crear un mundo virtual con sus propias leyes, su ecosistema económico, sus identidades y saltarse las normas y la gobernanza del mundo real.

Metaversos más importantes

1. Decentraland

Se trata de la plataforma más importante entre las distintas que existen en el metaverso. Esta plataforma recrea un mundo virtual, en el cual los usuarios pueden adquirir propiedades, construir en ellas e interactuar con otros usuarios. Cuenta con su propio *token*

para poder comprar, que es denominada MANA (perteneciente a Ethereum). Además, cuenta con el token LAND, un NFT basado en el estándar ERC-721 de Ethereum. LAND permite a los usuarios demostrar y mantener la propiedad digital sobre los terrenos dentro del Metaverso, haciendo posible la monetización.



5. Metaverso Decentraland

2. The Sandbox

The Sandbox es una plataforma parecida a Decentraland, la cual, permite a sus usuarios construir sus propios mundos virtuales, acceder a aplicaciones interactivas y comercializar diferentes objetos. Al tener su propio token y sus propios NFTs como Decentraland se trata de una red descentralizada. El token con el que se juega y se comercia es el SAND.



6. Metaverso The Sandbox

3. Somnium Space

A través de Somnium Space, los usuarios pueden crear y personalizar sus propios objetos y avatares. Y negociarlos como NFTs, para monetizar el juego y sus experiencias. Este metaverso *blockchain* también permite negociar con tierra digital, donde los usuarios pueden construir todo lo que imaginan, abriendo las puertas hacia una nueva economía digital. Esta plataforma se diferencia de las otras, en elq que puedes conectar experiencias 2D con experiencias en 3 dimensiones, siendo esto útil para poder negociar y apreciar mejor los distintos NFTs.

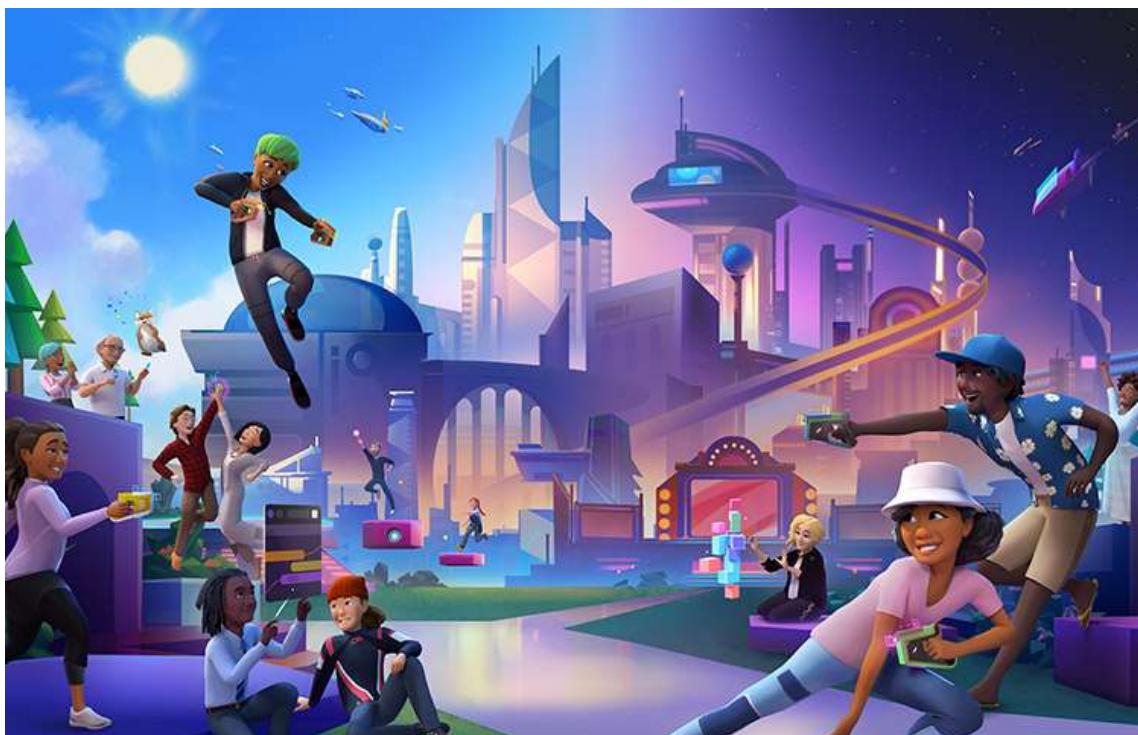


6. Metaverso Somnium Space

4. Horizon Worlds

Se trata de un metaverso creado por Meta (denominado anteriormente Facebook). Este metaverso está orientado a la realidad virtual y para poder acceder a él deberemos adquirir unas gafas de realidad aumentada, llamadas Oculus Quest. Actualmente sigue en periodo de pruebas y desarrollo y por el momento, no se ha planteado utilizar una metodología *blockchain*, aunque viendo el auge de los otros metaversos *blockchain*, se encuentra en estudio el introducirla.

En este metaverso, los avatares pueden viajar entre mundos creados por los usuarios, al tiempo que interactúan virtualmente con los habitantes de cada uno de estos lugares.[39]



7. Metaverso Horizon Worlds

Entono web

Además de poderlo desarrollar en el metaverso, se podría desarrollar en una página web utilizando las herramientas y códigos de programación ya existentes, por lo que, a continuación solo se van a plantear algunos métodos y herramientas con las que el cliente podría desarrollar un front-end y que son los más comunes.

1. HTML

Se trata de un lenguaje de programación que permite crear sitios web, a partir de la implementación de elementos, etiquetas y atributos. Sobre todo se utiliza para desarrollar la estructura a las páginas web.

2. CSS

Se trata de un lenguaje a partir el cual, se crea la estética de la página web. Es decir, que funcionalmente se encarga del diseño y la apariencia de la web desarrollada, con el fin que la web atraiga a más usuarios. Su uso es complementario tanto con HTML como con JavaScript.

3. JavaScript

Se trata de un lenguaje de programación multiplataforma e interpretado. Cuenta con diversas librerías para añadir funcionalidades de conexión de la web y estéticas. Con este lenguaje, sobre todo, se suele utilizar para la visualización de contenido a través de botones,

implementaciones de menús y animaciones y visualizaciones de imágenes. En la mayoría de las ocasiones se combina con CSS para completar el diseño de la web.

- **NodeJS**

Se trata de un entorno de tiempo de ejecución de JavaScript. Se trata de un entorno de código abierto, multiplataforma y que se ejecuta solo en el servidor. Sirve para crear sitios web dinámicos muy eficientes, escritos con el lenguaje de programación JavaScript.

- **Deno**

Se trata de otro entorno de tiempo de ejecución de JavaScript y TypeScript, que compite tanto directamente a nivel de uso en las empresas como por funcionalidades con NodeJS. La principal diferencia con NodeJS es el lenguaje de desarrollo, ya que, NodeJS está desarrollado en C++ mientras que Deno está desarrollado en Rust.

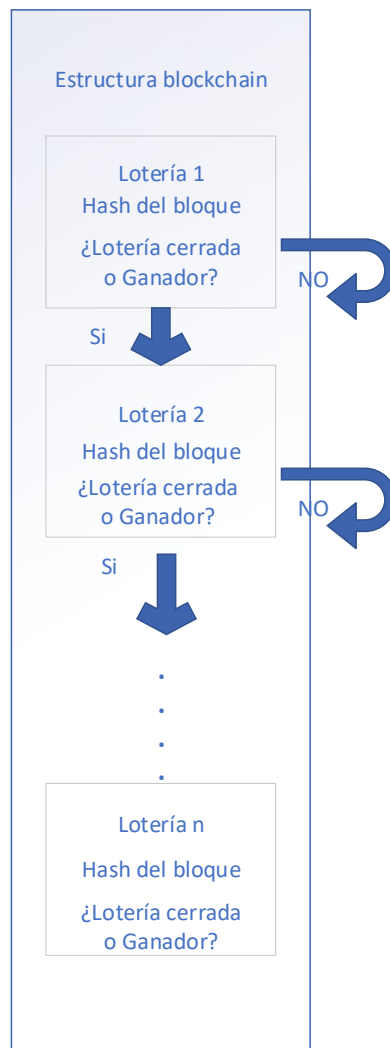
- **React**

Se trata de una biblioteca de JavaScript en código abierto diseñado para desarrollar las interfaces de usuarios de las páginas web. Es de código libre, aunque Facebook se encarga de mantener el código.

Arquitectura del proyecto

La arquitectura del proyecto se basa en el estudio del desarrollo de una lotería *blockchain*. Esta recurre a las criptomonedas como forma de pago y a los NFTs como participaciones de la lotería. Cada lotería tendrá un tipo de NFT, aunque dentro de la misma lotería serán los NFTs idénticos pero con direcciones diferentes.

La secuencia de la lotería se basa en una cadena de bloques. En cada bloque se desarrollará un sorteo de lotería que vendrá determinado por dos factores, el primero es que haya un ganador que entonces el bloque se cierra y comienza uno nuevo con un nuevo sorteo o a elección del propietario que tendrá en su mano cuando parar la lotería y realizar el sorteo. Por lo tanto, el cierre del bloque estará estrictamente relacionado con la elección de un ganador en la lotería del bloque anterior. La apertura del segundo bloque se dará cuando se *mintee*, es decir, se cree el primer boleto de lotería del segundo sorteo, que se prolongará hasta que el propietario decida cerrar la lotería.



Flujo del funcionamiento del programa

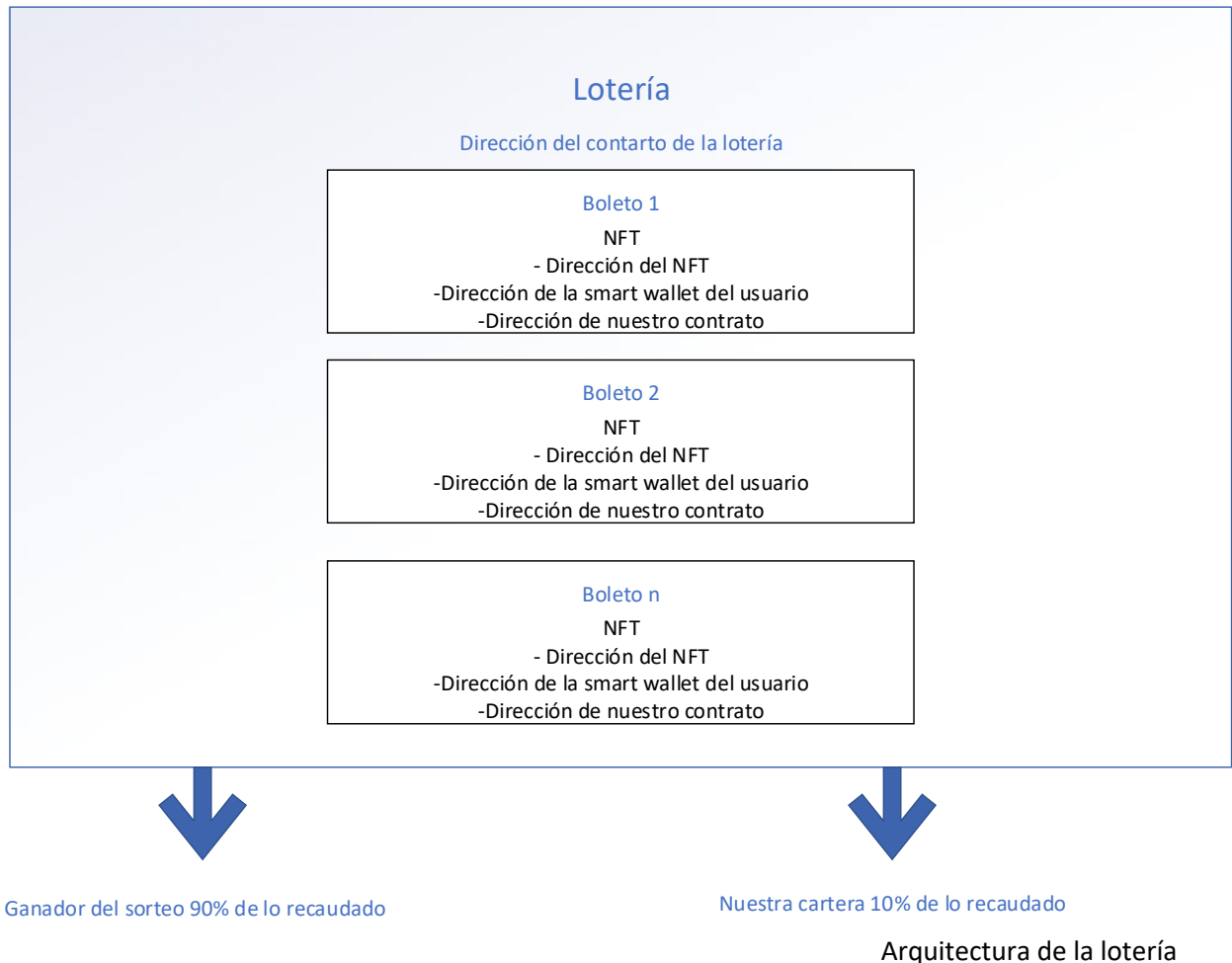
Respecto a la arquitectura de programación me baso en el estándar de Chainlink ERC-1155, el cual engloba diferentes estándares con el ERC-20 y el ERC-721, los cuales se crearon específicamente para la generación de nuevos tokens. Con el ERC-1155 como cito anteriormente, conseguimos una mayor eficiencia a la hora de producir un token nuevo, es decir, que cuando se genera un nuevo boleto de lotería (un NFT), el coste de gas será menor que con los otros estándares. Además la gran cantidad de métodos que incluye este estándar ya predefinido nos permite reutilizar una gran cantidad de código, con los beneficios, de saber que es un código ya testado y con un óptimo funcionamiento. Además, para que los métodos de del estándar funcionen en conjunto correctamente, he utilizado la interfaz IERC-165, compatible solo con los estándares ERC-20, ERC-721, ERC-777 y el ERC-1155. Por lo que la estructura de la creación de los NFTs se realiza con los métodos exclusivamente del estándar.

Cada lotería tendrá un identificador y podrá utilizar un NFT diferente respecto a otras loterías cambiando a la imagen que apunta el NFT, por lo que permite tener tanto a las loterías identificadas como a los ganadores de dichas loterías identificados.

Para obtener el ganador de la lotería utilizamos las librerías de Solidity keccak256, con el que codificamos la dirección del bloque, la dirección de origen y un número aleatorio con la librería timestamp, a partir de ahí restamos uno al bloque por lo que obtendremos una cantidad de salida totalmente aleatoria. Una vez tenemos dicho número hacemos el módulo entre la cantidad de boletos vendidos y el número aleatorio, que nos definirá cual es el ganador del sorteo.

Para que el programa sea algo más seguro, se ha desarrollado una función para intentar filtrar que no haya bots. Esto se realiza restringiendo las direcciones con las que puedas interactuar, en mi caso, hago una restricción para que no puedan jugar desde otro contrato externo. Es decir, que solo pueda interactuar una dirección de usuario, no un contrato. Esta función se realiza porque la mayoría de los bots juegan desde contratos externos, por lo que si se cumple una simple comparación entre que la dirección externa sea igual a una de las internas, la cuenta con esa dirección ya no podría jugar.

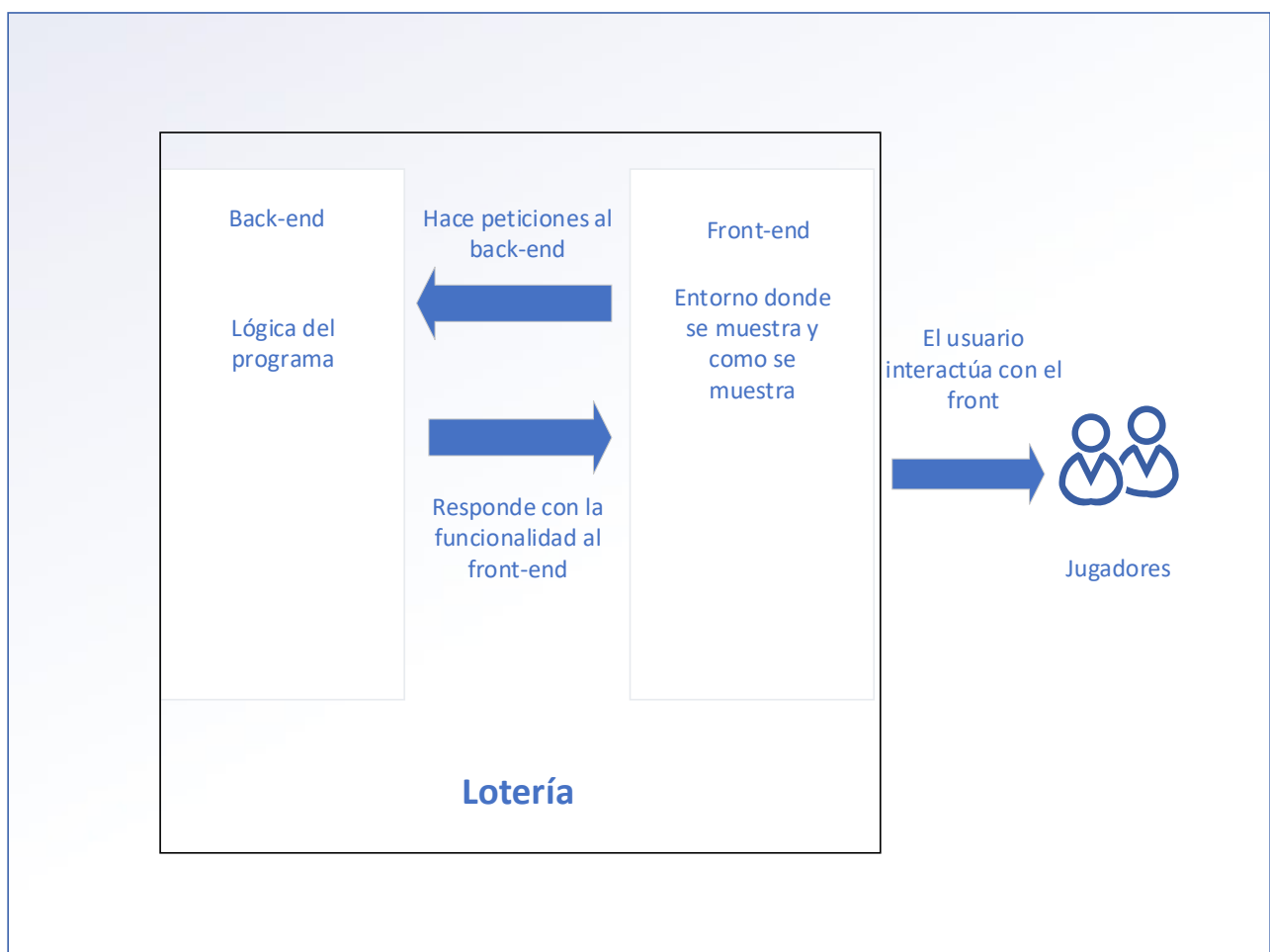
La lotería se pagará en dólares y cada cupón valdrá 12\$. Esto lo hemos hecho para que el público que juegue tenga una relación visual, aunque después esa cantidad de dólares se convertirá a Ethers. De esos 12\$ como administrador de la lotería me quedo con 1,2\$ es decir, un 10% y la otra cantidad es la que se juega. En caso de haber 10 jugadores, se jugarían 120\$ de los cuales yo me quedo con 12\$ y al ganador le tocarían 108\$. Este ingreso se realizaría directamente a la cartera con la dirección correspondiente de la compra del cupón.



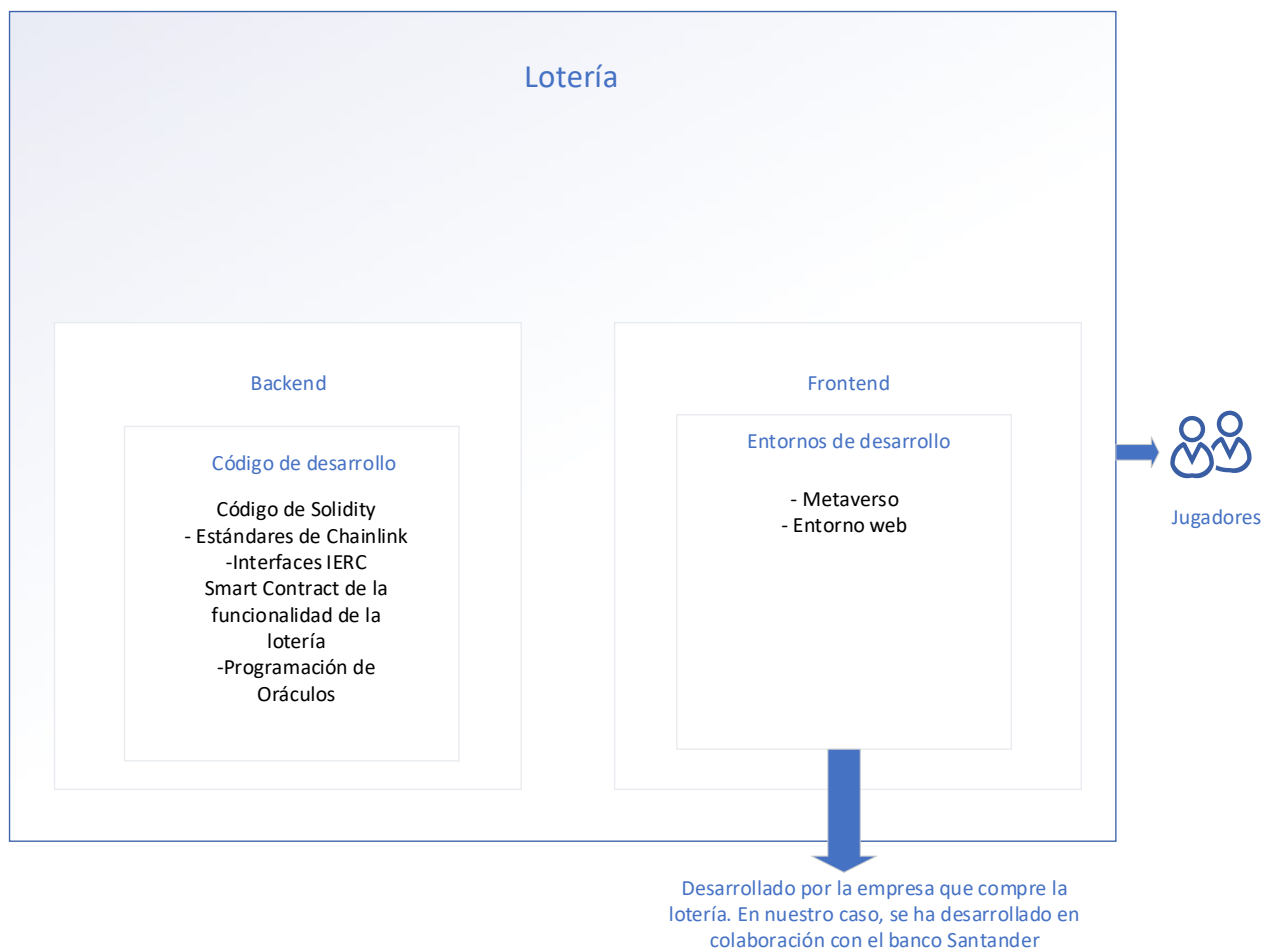
Para captar información del exterior, utilizamos un oráculo de Solidity. Esto lo utilizamos para poder realizar el cambio de dólares a Ethers. Este valor cambia constantemente, ya que el valor de las criptomonedas varía, por lo que el precio del cupón podrá variar el valor en criptomonedas, ya que 12\$, no será una cantidad constante de Ethers. Si el Ether baja, será correspondiente a una cantidad mayor de Ethers, mientras que si sube serán menos.

Por lo tanto, la estructura del programa viene definida por las interfaces y librerías, en las cuales, se da forma a la estructura así como ciertas funcionalidades como comprobaciones o la creación y destrucción de los NFTS, mientras que en el contract Lottery realizamos las funcionalidades de seguridad de bots, la ejecución del propio sorteo eligiendo el ganador y las funciones de transferencia de premio al usuario ganador y la transferencia nuestra de la comisión del bote recaudado.

También se creará, en colaboración con el banco, un *front-end* de prueba sencillo con vistas a que ya el comprador de la lotería desarrolle un nuevo *front* o lo instaure en uno de los metaversos ya preestablecidos, que es el fin, con el que se desarrolla esta lotería.



Estructura general de la aplicación



Estructura detallada de la aplicación

Fases de desarrollo del proyecto

Para el desarrollo del proyecto se han seguido diferentes etapas:

- La primera etapa fue una lluvia de ideas, sobre que proyecto podía realizar, relacionado con el trabajo ejercido en la empresa. Aunque la idea surgió relacionada a los bonos que iba a sacar el banco Santander relacionados con las criptomonedas y las redes *blockchain* que estaban probando dentro del banco.
- Una vez, se definió el tema del TFG, llegó la etapa de formación sobre el mismo, debido a la falta de conocimientos que tenía sobre el tema. Para ello se me recomendó el libro *Mastering Ethereum* y el curso de programación de criptomonedas, llamado *Cripto zombies*. Además desde el banco también se me proporcionó información y formación en como utilizar las herramientas *blockchain* y distintas librerías y herramientas propias del lenguaje de programación *Solidity*.
- La siguiente etapa fue la de estudio de métodos de extracción de los números aleatorios, en la cual se plantearon diversas opciones, como los métodos ya mencionados anteriormente.
- Una vez elegido como sacar el número aleatorio para saber cuál era el ganador, se comenzó con la fase de desarrollo del *back-end*, que fue identificar que librerías eran las más convenientes para desarrollar el proyecto, se comenzó utilizando la *erc20* y creando un token

y combinarla con la ERC-721 para desarrollar la lotería. Estudiando las diferentes librerías, se decidió abortar el proyecto y utilizar la ERC-1155, debido a sus interfaces y su mayor eficacia en el costo de gas.

- Una vez definidas las librerías y estándares, ya si que se paso a realizar el propio desarrollo de la funcionalidad de la lotería.
- Una vez que estuvo desarrollado el código, se estudió el hacer un *front-end* o implantarlo en el metaverso. La implantación en el metaverso no se llevó a cabo debido a motivos económicos, mientras que para realizar una página web, el equipo de *blockchain* del banco se puso en contacto con el departamento de desarrollo de *front-end*, que tienen modelos de web ya implementados y escogieron uno de base, lo modificaron y ha sido el que se ha utilizado en el proyecto. Como se menciona anteriormente, este *front-end*, ya es a elección del comprador del negocio de la lotería.
- La última etapa fue la elección, sobre que red de prueba de Metamask se iba a implementar, la cual se eligió Rinkeby debido a las pruebas de transacciones ya realizadas sobre esa misma red, anteriormente.
- Por último se realizó la compilación del código y se comprobó que las funciones se ejecutaban de manera correcta y había conectividad con el *front-end* y Metamask.

Diseño e implementación de la lotería

Partiendo de la arquitectura definida del proyecto, se comienza la etapa de diseño y desarrollo, en la que, se van observando las diferentes alternativas mencionadas y se eligen las que se creen más adecuadas.

El primer paso fue el ir probando las funciones de los distintitos estándares de OpenZeppelin, en los cuales se observa la creación por separado de un token y de un NFT con los estándares ERC-20 y ERC-721. Investigando sobre los diferentes estándares, encontramos el que utilizamos que es el ERC-1155, el cual incluye funcionalidades de ambos estándares y se observa que utilizando dicho estándar es el método más fácil de implementar la lotería.

La lotería se divide en contratos inteligentes, interfaz IERC-165 y las asociadas a ERC-1155, que son la IERC-1155, IERC-1155MetadataURI y IERC1155Receiver y la librería address. Con esta librería de Solidity restringimos las operaciones con cuentas externas, es decir, restringirá el acceso a las cuentas externas del *Smart contract*, a las cuentas externas que intenten interactuar, contratos en construcción, una dirección con un contrato inteligente creado o una dirección que haya tenido un contrato en algún momento creado, aunque en la actualidad ya no lo tenga operativo. Por lo que se restringe en gran parte de las llamadas a solo direcciones de las *smart wallets*. Por otra parte,

nos encontramos con los contratos e interfaces basados en el ERC-1155, que son los que dan la estructura a la lotería y a los boletos de la lotería, que incluye la creación de los NFTs y la quema de los mismos y seguridad en las transacciones, mientras que el último contrato ya es código propiamente desarrollado, en el cual incluyo la funcionalidad al programa.

Dentro de la librería las funciones más destacadas serían las siguientes:

La primera función es para comprobar si es un contrato comprobando la longitud de la cuenta.

```
function isContract(address account) internal view returns (bool) {  
  
    uint256 size;  
    assembly {  
        size := extcodesize(account)  
    }  
    return size > 0;  
}
```

La segunda función importante de la librería es para que el coste en gas se cobre al destinatario y por lo tanto, el dueño del contrato no cargue con ese coste de dichas transacciones. Esto hace una comprobación si se puede hacer cargo del gas además del coste del boleto el destinatario. En caso de que no pueda hacerse cargo, anula la operación.

```
function sendValue(address payable recipient, uint256 amount) internal {  
    require(address(this).balance >= amount, "Address: insufficient balance");  
  
    (bool success, ) = recipient.call{value: amount}("");  
    require(success, "Address: unable to send value, recipient may have reverted");  
}
```

Otra función importante de la librería es la función Call. Esta sirve para que el destinatario no pueda revertir las operaciones. De ellas en la librería hay diferentes variantes, las cuales incluyen el intentar revertir llamadas al contrato o revertir operaciones con devoluciones de tokens. También las variantes incluyen el envío de diferentes mensajes de error dependiendo de lo que se quiera revertir. Aunque se han instaurado todas las versiones en el código que incluye la librería la función de la que parten es la siguiente.

```
function functionCall(address target, bytes memory data) internal returns (bytes memory) {  
    return functionCall(target, data, "Address: low-level call failed");  
}
```

Las funciones adjuntas anteriormente son las que se consideran más principales dentro de la librería address, siendo estas las que brindan la seguridad al programa y por lo que utilizamos dicha librería ya instaurada en Solidity y que se pueden encontrar en la siguiente dirección de Github.

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol>

Respecto a lo primero que comenzamos a desarrollar es la interfaz IERC1155Receiver. Esta interfaz la hemos utilizado para que solo puedan devolvernos un tipo de token. Esta interfaz se llama al final de la función SafeTransferFrom. Para aceptar la transacción, esta función debe devolver lo siguiente: ``bytes4(keccak256("onERC1155Received(address,address,uint256,uint256,bytes)"))``

Para forzar a que nos tengan que devolver el tipo byte4 utilizamos la siguiente función.

```
function onERC1155Received(
    address operator,
    address from,
    uint256 id,
    uint256 value,
    bytes calldata data
) external returns (bytes4);
```

La siguiente interfaz que utilizamos es otra del estándar ERC-1155 que es la IERC-1155. Esta interfaz es obligatoria si se quiere desarrollar un contrato con el estándar ERC-1155. Se encarga del tema de las transacciones y el tipo de token que se transfiere, con diferentes aprobaciones. De esta interfaz las principales funciones y eventos son los siguientes.

- En primer lugar, nos encontramos con los eventos de transferencia para una y múltiples operaciones que salta cuando el valor del token es transmitido. Para ello nos encontramos las siguientes funciones

```

event TransferSingle(address indexed operator, address indexed from, address indexed to, uint256 id, uint256 value);

/**
 * @dev Equivalent to multiple {TransferSingle} events, where `operator`, `from` and `to` are the same for all
 * transfers.
 */
event TransferBatch(
    address indexed operator,
    address indexed from,
    address indexed to,
    uint256[] ids,
    uint256[] values
);

/**
 * @dev Emitted when `account` grants or revokes permission to `operator` to transfer their tokens, according to
 * `approved`.
 */
event ApprovalForAll(address indexed account, address indexed operator, bool approved);

```

- En segundo lugar tenemos el evento que se lanza cuando se le cambia la dirección de la imagen de cada NFT

```

event URI(string value, uint256 indexed id);

```

- Una vez que se declaran los eventos, comienzan a desarrollarse las funciones de la interfaz. En primer lugar se encuentran las funciones de comprobación de cuentas, ya que, la dirección de la cuenta con la que interactúa nuestro contrato no puede ser 0 y que los ids y las cuentas tengan la misma longitud.

```

function balanceOf(address account, uint256 id) external view returns (uint256);

/**
 * @dev xref:ROOT:erc1155.adoc#batch-operations[Batched] version of {balanceOf}.
 *
 * Requirements:
 *
 * - `accounts` and `ids` must have the same length.
 */
function balanceOfBatch(address[] calldata accounts, uint256[] calldata ids)
    external
    view
    returns (uint256[] memory);

```

- Las siguientes funciones destacables dentro de la librería son las que restringen las llamadas de los contratos que realizan las peticiones. Estos no pueden ser direcciones 0, deben haber sido aceptadas por las funciones definidas anteriormente en la librería setApprovalForAll y isApprovedForAll, además la cuenta debe tener al menos la cantidad de tokens, en el caso de nuestro proyecto de 0,0071 Ether + el coste del gas, que será el coste de comprar un

cupón, para poder realizar una transacción y por último se debe haber implementado bajo las librerías IERC1155Receiver-onERC1155Received.

```
function setApprovalForAll(address operator, bool approved) external;

/**
 * @dev Returns true if `operator` is approved to transfer ``account``'s tokens.
 *
 * See {setApprovalForAll}.
 */
function isApprovedForAll(address account, address operator) external view returns (bool);
```

```
function safeTransferFrom(
    address from,
    address to,
    uint256 id,
    uint256 amount,
    bytes calldata data
) external;
```

```
function safeBatchTransferFrom(
    address from,
    address to,
    uint256[] calldata ids,
    uint256[] calldata amounts,
    bytes calldata data
) external;
```

Como todas las librerías anteriores proceden de la interfaz IER-165, se declara un contrato con la única función que tiene la interfaz IERC-165, para que se puedan desarrollar e integrar el resto de las interfaces. Además, esta interfaz se crea para que los contratos puedan heredar y limita el gas a 30000 Weis.

Las transacciones no se pueden ejecutar automáticamente, por lo que debemos implementar el contrato de contexto para poder acceder a los datos de pago entre el comprador del boleto y el propietario del contrato, así como también en las transacciones de los pagos de los premios.

Una vez se definieron los contratos anteriores, continuamos con las interfaces que necesitamos en el proyecto, por lo tanto, se define la interfaz IERC1155MetadataURI y los métodos no definidos anteriormente de las interfaces ERC165 y IERC1155. La interfaz IERC1155MetadataURI, aunque es opcional, se implementa debido a la imagen del NFT que vamos a crear como boleto. Ya que, la imagen del boleto debe ser constante en la lotería y no puede variar hasta que esta, se encuentre cerrada.

En este contrato utilizando las diferentes librerías cabe a destacar diferentes métodos. El primero que se va a definir es el `__safeTransferFrom`. Sirve para realizar las diferentes comprobaciones para que las direcciones de envío no puedan ser 0, ya que, en ese caso, se perdería la emisión del

token y que esté implementado con las interfaces IERC1155Receiver-onERC1155Received. También debe devolver un valor específico y emite un evento de transferencia si realiza la función correctamente. Muy parecida a esta función es la `_safeBatchTransferFrom`, la cual se diferencia de la anterior porque trata con transacciones entre *Smart contracts*, en vez de con direcciones de otras carteras.

```
function _safeTransferFrom(
    address from,
    address to,
    uint256 id,
    uint256 amount,
    bytes memory data
) internal virtual {
    require(to != address(0), "ERC1155: transfer to the zero address");

    address operator = _msgSender();

    _beforeTokenTransfer(operator, from, to, _asSingletonArray(id), _asSingletonArray(amount), data);

    uint256 fromBalance = _balances[id][from];
    require(fromBalance >= amount, "ERC1155: insufficient balance for transfer");
    unchecked {
        _balances[id][from] = fromBalance - amount;
    }
    _balances[id][to] += amount;

    emit TransferSingle(operator, from, to, id, amount);

    _doSafeTransferAcceptanceCheck(operator, from, to, id, amount, data);
}
```

Para poder generar los NFT de los boletos utilizamos las funciones `_mint` y `_mintbatch`. Con estas funciones también realizamos las comprobaciones similares a las de las funciones anteriores, pero con las cantidades de NFTs creados y que la cantidad sea igual al identificador, ya que se van asignando los identificadores a los NFTs, en función que se crean, por lo que deben coincidir. Además se comprueba si se utiliza la interfaz IERC1155Receiver-onERC1155Received y emite un evento si realiza correctamente la función.

```
function _mintBatch(
    address to,
    uint256[] memory ids,
    uint256[] memory amounts,
    bytes memory data
) internal virtual {
    require(to != address(0), "ERC1155: mint to the zero address");
    require(ids.length == amounts.length, "ERC1155: ids and amounts length mismatch");

    address operator = _msgSender();

    _beforeTokenTransfer(operator, address(0), to, ids, amounts, data);

    for (uint256 i = 0; i < ids.length; i++) {
        _balances[ids[i]][to] += amounts[i];
    }

    emit TransferBatch(operator, address(0), to, ids, amounts);

    _doSafeBatchTransferAcceptanceCheck(operator, address(0), to, ids, amounts, data);
}
```

Respecto a las funciones de `_burn` y `_burnbatch`, son idénticas a las de `_mint` y `_mintbatch`, pero con la diferencia que en vez de, añadir el NFT al array, resta el NFT en el que se encuentra el array al recorrerlos con el bucle `for`. Por lo tanto, al contrario de las anteriores estas además de hacer las mismas comprobaciones sirven para destruir el NFT una vez ya transmitido. Esto sirve para que el token no pueda ser repetido varias veces, es decir, que una vez entregado al cliente destruimos el lote de nuestro montón de la lotería y lo tiene solo el cliente y no se puede volver a crear un token con la misma dirección.

Para que se pueda realizar, implicando la creación y destrucción de los boletos, se deben cumplir las funciones de comprobación `beforeTokenTransfer`, `_doSafeTransferAcceptanceCheck`, `_doSafeBatchTransferAcceptanceCheck` y `_asSingletonArray`. Con ellas se realiza las siguientes comprobaciones:

- Que en una sola transacción que tanto el id como la cantidad que se transfiere solo sea 1.
- Que no se realice una transferencia de 0 tokens, obteniendo un gasto de gas innecesario
- Que no se destruyan 0 tokens, debido también al motivo que habría un gasto de gas innecesario
- Que ninguna de las direcciones, tanto la de emisor como la del receptor, sea 0.

```
function _beforeTokenTransfer(
    address operator,
    address from,
    address to,
    uint256[] memory ids,
    uint256[] memory amounts,
    bytes memory data
) internal virtual {}

function _doSafeTransferAcceptanceCheck(
    address operator,
    address from,
    address to,
    uint256 id,
    uint256 amount,
    bytes memory data
) private {
    if (to.isContract()) {
        try IERC1155Receiver(to).onERC1155Received(operator, from, id, amount, data) returns (bytes4 response) {
            if (response != IERC1155Receiver(to).onERC1155Received.selector) {
                revert("ERC1155: ERC1155Receiver rejected tokens");
            }
        } catch Error(string memory reason) {
            revert(reason);
        } catch {
            revert("ERC1155: transfer to non ERC1155Receiver implementer");
        }
    }
}
```

```
function _doSafeBatchTransferAcceptanceCheck(
    address operator,
    address from,
    address to,
    uint256[] memory ids,
    uint256[] memory amounts,
    bytes memory data
) private {
    if (to.isContract()) {
        try IERC1155Receiver(to).onERC1155BatchReceived(operator, from, ids, amounts, data) returns (
            bytes4 response
        ) {
            if (response != IERC1155Receiver(to).onERC1155BatchReceived.selector) {
                revert("ERC1155: ERC1155Receiver rejected tokens");
            }
        } catch Error(string memory reason) {
            revert(reason);
        } catch {
            revert("ERC1155: transfer to non ERC1155Receiver implementer");
        }
    }
}
```

```
function asSingletonArray(uint256 element) private pure returns (uint256[] memory) {
    uint256[] memory array = new uint256[](1);
    array[0] = element;

    return array;
}
```

Para permitir el manejo de ciertas funciones específicas declaramos el contrato Ownable. De estas, destaco la función de transferOwnership, para modificar el propietario de la lotería y la función de owner con el modificador de onlyOwner. Los modificadores o *modifiers* en inglés sirven como prerequisites para que la función se pueda ejecutar. Por lo tanto pueden cambiar el comportamiento de dicha función.

```
function owner() public view virtual returns (address) {
    return _owner;
}
```

```
modifier onlyOwner() {
    require(owner() == _msgSender(), "Ownable: caller is not the owner");
    _;
}
```

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    _setOwner(newOwner);
}
```

Por último, fuera del contrato y ya antes de realizar el desarrollo propio del código (no reutilizado de estándares, librerías en interfaces) cabe destacar también la programación de la interfaz AggregatorV3Interface. Esta interfaz contiene la programación de un oráculo para poder coger el valor de los dólares y Ethers y poder hacer la conversión entre ellos.

```

interface AggregatorV3Interface {
    function decimals() external view returns (uint8);

    function description() external view returns (string memory);

    function version() external view returns (uint256);

    function getRoundData(uint80 _roundId)
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        );

    function latestRoundData()
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        );
}

```

El contrato inteligente denominado Lottery es el desarrollo de la funcionalidad de la lotería, definiendo aspectos fundamentales, como, cuál será el ganador de la lotería, el estado de esta, los diferentes tipos de loterías, la creación de los boletos, las transacciones de pago y las imágenes de los boletos de cada lotería. Además contendrá un intento de restricción para no interactuar con bots y funciones reservadas para que el propietario pueda crear boletos gratuitamente cuando quiera y que pueda sacar los ingresos cuando también lo desee el propietario de la lotería. Estas funciones se desarrollan en caso de que haya habido algún hackeo y la función de crear los boletos por si se desea en un futuro cambiar las funcionalidades y se necesitan hacer diversas pruebas técnicas.

En primer lugar, se guardan en variables realizando un mapping correspondiente a cada variable las imágenes de NFTS, los números de boletos generados en cada lotería, los ganadores de las loterías, el número de loterías jugadas hasta el momento, el precio del ticket y el porcentaje de beneficio que vamos a tener en la lotería. El precio del ticket va seguido de 8 ceros, debido a que es así como el oráculo almacena el valor.


```

contract Lottery is ERC1155, Ownable {
    using Strings for uint256;

    //Guarda cada imagen para cada NFT. Para poder modificar cada NFT
    mapping(uint => string) public typeToURI;

    //Guarda los boletos, Para tener en cuenta cuantos boletos se han mintado y consecuentemente el premio variará.
    mapping(uint => uint256) public typeToAmountMinted;

    //Guarda ganador loteria
    mapping(uint => address) public typeToWinner;

    //Cuántas loterías se han realizado
    uint public nLottery;

    //Precio del ticket
    uint256 public ticketPrice = 1200000000; //12$
    //Porcentaje de lo que nos llevamos nosotros
    uint public feePercentage = 10;
}

```

Por motivos de seguridad se define a continuación un enumerado, que permite cambiar el estado de la lotería a cerrado cuando lo necesitemos.

Una vez definas las variables y el enumerado se define el constructor del contrato en el que inicializaremos el número de loterías, la imagen de cada NFT para la lotería que se va a jugar y la dirección en hexadecimal de donde el oráculo coge la información.

```

constructor() {
    nLottery += 1;
    typeToURI[nLottery] = "ipfs://QmRegAu9doxopUQAodKDPBT96kjpEd75rFLH5oohYmnBhg/";

    priceFeed = AggregatorV3Interface(0x8A753747A1Fa494EC906cE90E9f37563A8AF630e);
}

```

En caso de que un usuario introduzca una cantidad, declaramos una función receive() para que la lotería no se quede estancada, si recibe ethers sin llegar a comprar el cupón. En tal caso se le devolverá la cantidad de ethers al jugador.

Como regla general, la mayoría de las llamadas al contrato de la lotería que provengan de contratos externos suelen ser de bots, que intenten hackear el juego. Por eso restringimos esas llamadas para que no se puedan hacer desde los contratos externos con la siguiente función.

```

modifier noBots() {
    require(msg.sender == tx.origin, "No bots!");
    _;
}

```

Además se han desarrollado también dos funciones, comprobando que el estado de la lotería es on, es decir, que esté activa y que es correcto el valor de los ethers que hay en cada transacción.

```

modifier isContractState(ContractState contractState_) {
    require(contractState == contractState_, "Lottery is Paused!");
    _;
}

/**
 * Correcta cantidad de Ether en cada transacción.
 */
modifier correctValue(uint256 expectedValue) {
    require(expectedValue >= msg.value, "Ether value sent is not correct!");
    _;
}

```

Para coger los valores y poder realizar conversiones entre dólares y ethers, utilizamos el oráculo de la interfaz AggregatorV3Interface y le pasamos la dirección de donde cogerá el valor. Utilizando el valor que devuelve la función y la propia interfaz se desarrollan las funciones de conversión de ethers a dólares y viceversa.

```

* Precio actual eth. Ligado a los cambios de ETH a $
*/
function changePriceFeed(address newFeed) public onlyOwner {
    priceFeed = AggregatorV3Interface(newFeed);
}

function getETHtoUSD() public view returns(int) {
    (,int price,,) = priceFeed.latestRoundData();
    return price;
}

function getPriceETH() public view returns(uint256) {
    int price = getETHtoUSD();
    return (ticketPrice * 1000000000000000000) / uint256(price);
}

```

Para crear los boletos desarrollamos la función mint, la cual tiene en cuenta, funciones anteriores como la restricción a los bots, el estado de la lotería, que se debe encontrar activa y los ethers totales que tenemos en función de los boletos vendidos.

En esta función si el id de la lotería es correcto y no hay un ganador ya definido, se añaden al array minters los nuevos ids de los participantes. También se utiliza el estándar ERC-1155, para la creación de los boletos, pasando a esa función las direcciones de destino, origen, la cantidad y los bytes de memoria inicializados a nada, porque en un origen la lotería consume menos de un byte.

```

function mint(address _to, uint _id, uint _amount) external payable isContractState(ContractState.ON) correctValue(getPriceETH() * _amount) noBots() {
    require(_id == nLottery, "Invalid Lottery ID!"); //nlottery cantidad loterias que llevamos
    require(typeToWinner[_id] == address(0), "This lottery is closed!"); //Que no haya ganador ya
    for (uint i; i < _amount; i++) { //Lista de ids de participantes
        minters.push(_to);
    }
    typeToAmountMinted[nLottery] += _amount;
    _mint(_to, _id, _amount, "");
}

```

Se crea la función mintReserved en caso de que se desee realizar alguna prueba y se necesite crear gratuitamente boletos, para ello esta función llama al método _mintbatch del estándar ya definido ERC-1155.

Para cambiar el estado de la lotería cuando se desee de on a off, sin que se haya definido un ganador se define una función que modifica el valor del enumerado que define el estado de la lotería. Esta función, al igual que la de mintReserved, será solo de prueba para el propietario, ya que, en un caso normal, la lotería se cerrará cuando se defina un ganador.

```
function mintReserved(uint256[] memory ids, uint256[] memory amounts) external onlyOwner {
    _mintBatch(owner(), ids, amounts, "");
}

/**
 * Cambiar estado de la lotería
 */
function setContractState(uint contractState_) external onlyOwner {
    require(contractState_ < 2, "Invalid Contract State!");
    if (contractState_ == 0) {
        contractState = ContractState.OFF;
    }
    else {
        contractState = ContractState.ON;
    }
}
```

Para comenzar una nueva lotería, primero debe estar definido un ganador, por lo que, en la función para comenzar la lotería se hace lo primero dicha comprobación. Además se incrementa el contador de número de loterías, se asigna la dirección de la imagen a los boletos y se eliminan los boletos que había creados de la lotería anterior.

```
/**
 * Nueva lotería.
 */
function newLottery(string memory newUri) external onlyOwner {
    require(typeToWinner[nLottery] != address(0), "A winner has not been chosen yet for this lottery!");
    nLottery += 1;
    typeToURI[nLottery] = newUri;

    delete minters;
}
```

La siguiente función es la que elige el ganador del sorteo. Es decir, es aquella que llama a la función que saca un número aleatorio y entre todos los boletos y el ganador será la dirección asociada a la posición del boleto ganador boleto ganador.

Una vez que se ha elegido el ganador se llama a las funciones de pago para realizar las transacciones a la cuenta ganadora.


```

/**
 * Elige ganador
 */
function pickWinner() external onlyOwner {
    require(typeToWinner[nLottery] == address(0), "A winner has already been chosen for this lottery!"); //Comprueba que no sea dirección 0
    uint256 winnerNumber = randomLotteryNumber(typeToAmountMinted[nLottery]); //Cuantos NFTs que se han mintado
    address winner = minters[winnerNumber];

    uint256 balance = accountBalance();
    require(balance > 0, 'No Funds to withdraw, Balance is 0');
    _withdraw payable(owner()), (balance * feePercentage) / 100;
    _withdraw payable(winner), accountBalance(); //Para evitar que se queden eth sueltos

    typeToWinner[nLottery] = winner;
}

```

La función randomLotteryNumber va asociada a la de pickWinner, ya que es la que calcula el número aleatorio para elegir al ganador. Como se ha comentado en apartados anteriores se han estudiado diferentes métodos para poder hallar el número aleatorio. Aunque al final, se ha realizado de la forma aparentemente más simple. Este número aleatorio se calcula a partir de la codificación del hash del bloque y modificándolo en una unidad. Lo que consecuentemente debido a las propiedades de la librería keccak256 nos genera el número aleatorio. Una vez que lo ha calculado, se haya el resto entre el número máximo de boletos generados.

```

function randomLotteryNumber(uint max) public view returns(uint256) {
    return uint256(keccak256(abi.encodePacked(
        tx.origin,
        blockhash(block.number - 1),
        block.timestamp,
        _msgSender()
    ))) % max;
}

```

Las siguientes funciones son para la visualización de los NFTs y por si queremos asignar una nueva dirección a cada imagen de los NFTs, es decir, de los boletos.

```

function updateTypeUri(string memory newUri, uint typeId) external onlyOwner {
    require(typeId <= nLottery, "URI requested for invalid token type");
    typeToURI[typeId] = newUri;
}

//Visualización de cada imagen de cada NFT
function uri(uint256 typeId) public view override returns (string memory) {
    require(typeId <= nLottery, "URI requested for invalid token type");
    return typeToURI[typeId];
}

```

Por último nos encontramos con las funciones de pago. La primera es la función que devuelve los ethers que tiene el contrato. La segunda es en caso de tener algún problema y que el dueño tuviera que lanzar esta función y sacar a su dirección de cartera todos los fondos del contrato y la última es la que se ejecuta para pagar al ganador el premio que haya en el contrato, menos la comisión que nos llevamos nosotros y la que utilizamos para ingresar nosotros los beneficios en nuestra cartera.

```

//Para ver los eth en el contrato
function accountBalance() public view returns(uint256) {
    return address(this).balance;
}

//Quita todos los fondos del contrato
function withdraw() public onlyOwner {
    uint256 balance = accountBalance();
    require(balance > 0, 'No Funds to withdraw, Balance is 0');

    _withdraw(payable(owner()), balance);
}

//Nos envía el porcentaje de beneficio a nuestra cartera y para pagar el premio
function _withdraw(address payable account, uint256 amount) internal {
    (bool sent, ) = account.call{value: amount}("");
    require(sent, "Failed to send Ether");
}

```

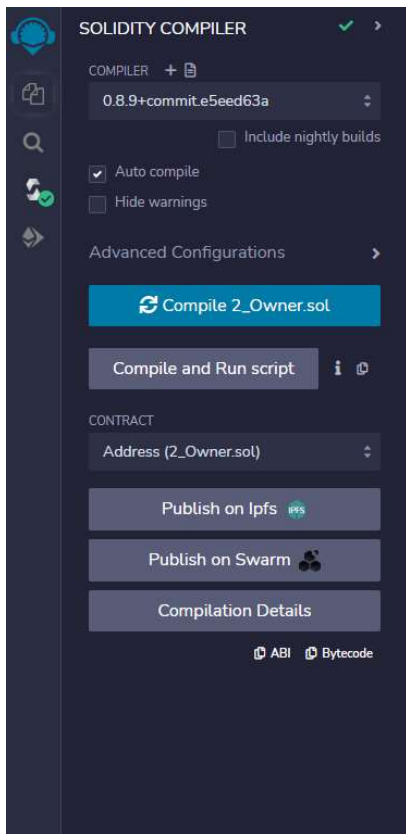
Con estas funciones concluye el desarrollo tanto de la estructura como funcional de la lotería. Para desarrollar un *front-end* sencillo se buscará a un cliente, para llevarlo a cabo y así simular el modelo de negocio real el cual se pretende llevar a cabo. Ya que lo que se trata es que se pueda implementar en múltiples plataformas y con cliente diferentes.

Compilación, ejecución y conclusiones

Compilación y ejecución del código

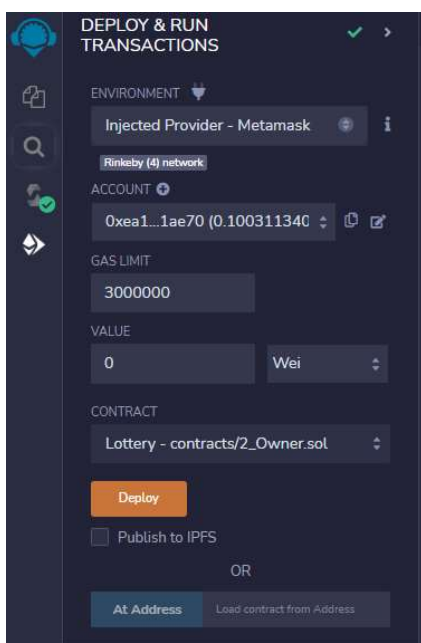
El código se comprende de una parte del *back-end*, en la que se desarrolla la parte lógica del programa y el *front-end*, que es el que hace las peticiones al *back-end* y se encarga de mostrar la información así como el estilo del entorno. En nuestro caso, nos hemos encargado de desarrollar la parte del *back-end*, mientras que el front, lo ha desarrollado el equipo de desarrollo del Banco Santander.

En primer lugar, para poder ejecutar el programa deberemos tener instalada la extensión de Metamask en el navegador y tener una cuenta asociada. En nuestro caso se va a probar con la red de prueba de Rinkeby, a partir del cual podemos simular una cartera con Ethers, pero de forma gratuita. Una vez que la tengamos configurada, nos dirigimos al compilador de remix, conectándonos a la URL <https://remix.ethereum.org/> . Una vez que estemos dentro del compilador nos dirigimos al *namespace* donde hemos desarrollado el proyecto. Una vez que estamos en el *namespace* correspondiente, nos dirigimos al apartado de compilación, como se puede observar en la siguiente imagen.



Se recomienda compilarlo con la versión 0.8.9

Una vez que lo hemos compilado, deberemos dirigirnos al apartado de ejecución, en el cual deberemos elegir la opción de injected provider-metamask en el entorno. También deberemos elegir el contract lottery.sol y aunque normalmente se establece automáticamente la cuenta de metamask, comprobar que es la de nuestra cartera. Una vez tengamos ajustados los parámetros, se hará el deploy del contract. Se abrirá nuestra cartera de metamask y deberemos aprobar la transacción. Esta transacción como todas, se podrá ver en etherscan.



The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, showing the 'Lottery - contracts/2_Owner.sol' contract being deployed to the Rinkeby testnet. The 'GAS LIMIT' is set to 3000000 and the 'VALUE' is 0 Wei. The 'CONTRACT' dropdown shows 'Lottery - contracts/2_Owner.sol'. The 'Deploy' button is highlighted. Below the deployment panel, a list of 'Deployed Contracts' shows 'LOTTERY AT 0x007...4506f (BLOCK)'. The main editor displays the Solidity code for the lottery contract, including functions like 'newLottery', 'pickWinner', 'randomLotteryNumber', 'updateTypeUri', and 'uri'. The bottom status bar shows a successful deployment: '[block:11354323 txIndex:2] from: 0xea1...1ae70 to: Lottery.(constructor) value: 0 wei data: 0x088...8672f logs: 1 hash: 0xf36...872e4 creation of Lottery pending...'. On the right, a sidebar shows the 'Contract nuevo' details for 'https://remix.ethereum.org', including 'Tarifa estimada de gas' (0.01136639 RinkebyETH) and 'Total' (0.01136639 RinkebyETH).

The screenshot shows the Etherscan interface for the Rinkeby Testnet. The 'Transaction Details' page is displayed, showing the transaction hash '0xd08530b48ffab607ec333859e7285e8cdb1760284e1af4a4454d5fd9b23d6c9b'. The transaction status is 'Success' with '1 Block Confirmation'. The transaction occurred '6 secs ago (Sep-10-2022 06:00:55 AM +UTC)'. The 'From' address is '0xea13d5285a48942bc9f78adfed11749848f1ae70'. The 'To' is '[Contract 0xd79e8e1c8dee589e5059eb198875744044937cd9 Created]'. The 'Value' is '0 Ether (\$0.00)'. The 'Transaction Fee' is '0.011366392536372456 Ether (\$0.00)'. The 'Gas Price' is '0.000000002500000008 Ether (2.500000008 Gwei)'. A link 'Click to see More' is provided at the bottom.

Una vez que se ha creado el contrato de la lotería con su hash y dirección correspondiente, se deberá modificar el apartado de *front-end* el contractAdres, para poder conectar el *front-end* con ese contrato de esa lotería específica.

```
const contractAddress = '0x5CdF8100c11f8d77c7C46D3a6b99aaee2b427A0D';

document.getElementById('total-supply').style.display = 'none';
document.getElementById('minted').style.display = 'none'

window.onload = function() {
```

Una vez cambiada la dirección y guardado, se deberá instalar a través del terminal de Visual Studio, en el directorio del *front-end* del programa, la dependencia `http-server` a través del comando `npm install --global http-server`. Una vez que esté instalado se deberá ejecutar el comando `http-server`. En nuestro caso elegimos conectarnos a la IP: `http://127.0.0.1:8080` que será la dirección del servidor web donde se encuentra el front-end y ya la parte visual de la lotería.

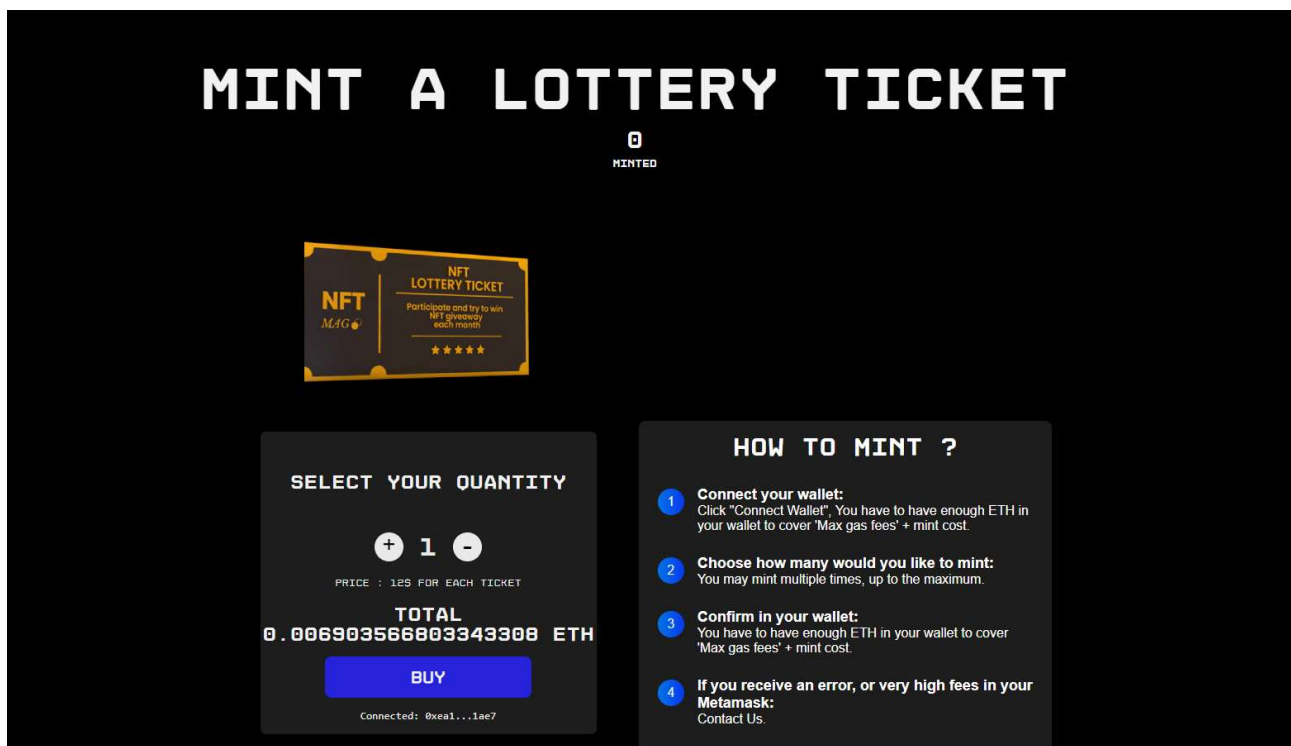
```
PS C:\Users\JAIME TELLO> cd desktop
PS C:\Users\JAIME TELLO\desktop> cd tfg
PS C:\Users\JAIME TELLO\desktop\tfg> cd front
PS C:\Users\JAIME TELLO\desktop\tfg\front> cd frontend
PS C:\Users\JAIME TELLO\desktop\tfg\front\frontend> cd frontend
PS C:\Users\JAIME TELLO\desktop\tfg\front\frontend\frontend> npm install --global http-server
```

```
PS C:\Users\JAIME TELLO\desktop\tfg\front\frontend\frontend> http-server
Starting up http-server, serving ./

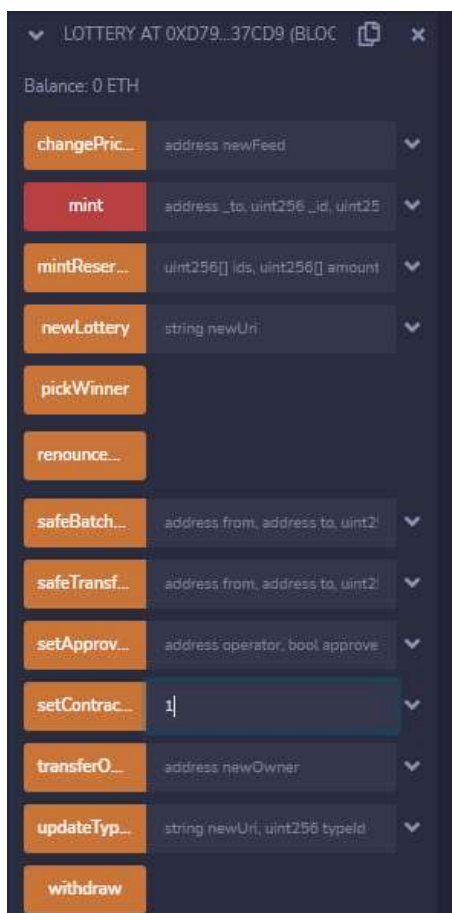
http-server version: 14.1.1

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://192.168.56.1:8080
  http://192.168.99.1:8080
  http://192.168.1.41:8080
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
```

En el *front-end*, ya solo nos deberemos preocupar, de tener la cuenta de Metamask conectada, que en nuestro caso al haber ido accediendo antes, se conecta ya automáticamente. Además antes de comenzar a jugar deberemos dirigirnos al remix, y ejecutar la función `setContractState`, pasándole un 1 para abrir el sorteo.



Una vez esté abierta ya la lotería se podrá proceder ya a la compra de los boletos. Que en la web se muestra cuantos boletos han sido creados y por tanto, si hay muchos jugadores jugando y si vas a tener mayor o menor probabilidades de conseguir el premio.

Para cerrar el sorteo ejecutamos la función `pickWinner`. En este caso de ejemplo al solo haber 1 boleto creado, seré yo quien gane y las direcciones en el Metamask de envío de premio y la comisión del 10% del beneficio que se lleva el dueño del contrato de la lotería, se realizarán a la misma dirección de la *smart wallet*. Una vez realizadas las transferencias, ya se dará por cerrada la lotería y habrá que crear una nueva, pasándole a la función `newLottery` una nueva dirección de contrato o realizando la ejecución del *back-end* y repitiendo el proceso.

Conclusiones sacadas del proyecto

Este proyecto de investigación y desarrollo llevado a cabo en colaboración con el Banco Santander se ha realizado una investigación sobre el mundo del *blockchain*, en el que a través del programa desarrollado de la lotería, se pudo ver el funcionamiento y la estructura de la cadena de bloques, cada uno con su hash y como se va ligando un bloque a otro. Además se han estudiado diferentes métodos de extracción de números aleatorios, entre ellos diferentes oráculos o la extracción de números a través de un computador cuántico, aunque debido a la complejidad y los diferentes fallos de resultados lógicos que no salieron con el oráculo VRF, se decidió implementar a través de las librerías de Solidity. Todo esto se suma a la complejidad de aprender, Solidity, un nuevo lenguaje de programación enfocado a *blockchain*, que aunque tiene una lógica como C++ o Java orientada a objetos, se deben tener en cuenta otros parámetros como las direcciones de los bloques o los hashes. Además cada lenguaje tiene sus propias librerías, las cuales hay que entender la funcionalidad que tienen, para saber cuál utilizar.

Todo este proyecto, que se ha llevado a cabo, no se podría haber realizado sin el apoyo del Banco Santander, para el que me encuentro trabajando día a día, ya que, ellos fueron quienes me propusieron la idea y han sido quienes durante meses me han estado impartiendo una formación en la materia y me han proporcionado un desarrollo de un *front-end* para que se pueda ver de manera visual.

Economía

En el apartado económico hay que destacar el coste de los boletos y el beneficio en porcentaje sobre el precio de compra de los boletos. Este precio va a ser constante, independientemente del cambio de valor a los que constantemente está sometido el ether y el dólar. El coste que tendrá para el comprador será de 12\$ cada cupón que al cambio en ethers son 0,0078. El premio que se entrega al cliente será el 90% de la recaudación de la totalidad de la venta de los boletos.

Al ser una aplicación en la web, dependiendo del cliente que nos compre la aplicación y en que plataforma la instaure, dependerán los beneficios y el coste que pueda llegar a tener. Un ejemplo es el metaverso de Decentraland, que hoy en día es uno de los más conocidos. En este metaverso, el precio de las parcelas es bastante elevado, de hecho, en esta plataforma se vendió la parcela virtual más cara de la historia por un valor de 618,000 MANA que al cambio en dólares son 3,5 millones de dólares. Esta moneda se devaluó a la semana su precio pasó a rondar la mitad de lo pagado y a día de hoy el precio mínimo ronda los 4 ethers que al cambio son unos 14.700 dólares.

Se estima que en la aplicación desarrollada, si está instaurada en una de las grandes plataformas del metaverso pueda haber loterías con una cantidad de jugadores en torno a los 5000, que si lo

extrapolamos a beneficio para el dueño del contrato serían 6000 dólares de beneficio y 54000 dólares para el ganador de la lotería.

En este proyecto al poderlo desarrollar en colaboración con el banco, nos facilitan de forma gratuita el equipo de desarrollo de front-end, para poder demostrar el funcionamiento de la aplicación, pero, en caso de ser un cliente externo se le vendería la aplicación por una cantidad de dinero fija y además un porcentaje del beneficio que obtenga en dichas loterías, estos podrían variar en función de la plataforma en la que se vaya a instaurar.

Otra opción que se ha contemplado es la contratación de un equipo de front-end, informándonos, tanto en el banco como en la web encontramos que el precio medio de la contratación de un equipo de desarrollo de front-end, es entorno a los 60€/hora. Esto incluiría a 2 personas desarrollando el proyecto. Por lo que he visto que ha tardado en desarrollar el *front* el equipo formado por una persona más otra de apoyo en situaciones puntuales, calculo que en 5 días trabajando 8 horas diarias se podría completar una web similar a la que nos han proporcionado. Esto supondría un costo de unos 2400€. Que si tiene éxito, con que jueguen 2000 jugadores ya habríamos cubierto gastos y ya comenzaríamos a tener beneficio con el jugador 2001. Esto al poderse desarrollar en entornos web y estar al alcance de todo el mundo, se estima que el número de jugadores que haya en cada sorteo sea bastante superior a 2000 jugadores.

Respecto al coste de desarrollo de la aplicación ha sido solo el coste del libreo Mastering Ethereum, ya que, el resto de los programas para desarrollar el código son gratuitos. Respecto a los medios físicos, no he tenido que adquirir ningún equipo o componente específico para realizar el proyecto. El único coste en el proyecto sería como comento anteriormente lo que cueste el instaurarlo en una plataforma o si lo quieres instaurar tú mismo, lo que cueste un equipo de desarrollo de front-end y lo que cueste instaurarlo en alguna plataforma del metaverso o en la web.

Anexo

Glosario

Palabra Técnica	Definición
App	Diminutivo de aplicación
Back/back-end	Es la parte o rama del desarrollo web encargada de que toda la lógica de una página funcione
Blockchain	Cadena de bloques de criptomonedas
Front/front-end	Es la parte o rama del estilo y diseño de una web
Hardware	Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.
Ledger	Registro de todas y cada una de las operaciones y transacciones que se realizan dentro de una blockchain
Mintear	Dentro de los estándares de Chainlink, relativo a generar un token
Namespace	Contenedor abstracto o el entorno creado para alojar una agrupación lógica de identificadores únicos o símbolos
NFT	Token No Fungible
Nonce	Es un número de cuatro bits que se agrega a un bloque cifrado o encriptado en una cadena de bloques
Peer to peer	Red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí
<i>Proof of Work</i>	Prueba de trabajo
<i>Queries</i>	Consultas/Peticiones
Smart Contract	Programas informáticos diseñados para ejecutarse automáticamente a medida que las personas o empresas involucradas en un acuerdo van cumpliendo con las cláusulas del mismo
smart wallets	Cartera o billetera virtual de criptomonedas
Staking	Es parte del mecanismo que utilizan las criptomonedas que funcionan con una «prueba de participación» para verificar las transacciones realizadas
Token	Unidades de valor que se le asignan a un modelo de negocio
Trustless	Sin confianza
Trustlessness	Sin confianza
Software	Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Diferencias con MylottoCoin

MylottoCoin	Nuestra lotería
Sean los jugadores que sea, siempre hay las mismas probabilidades de ganar, mientras que el premio si varía en función de los jugadores	Pocos jugadores, menor premios, pero será mayor la probabilidad de ganar. En función que haya más jugadores habrá un mayor premio pero irán bajando las probabilidades
Utiliza el Oráculo para obtener información de Powerball. Por lo que depende de esa página externa	El sorteo se celebrará independientemente de las páginas externas al elegir un ganador por sorteo de los jugadores
Código abierto en GitHub	Código en un repositorio privado en GitHub
Precio del cupón 0.19 BNC, aprox 56\$, que podrá variar en función del valor de la criptomoneda	Utilización del oráculo para un precio fijo de 12\$, que a día de hoy son 0,0069 Ethers
Solo utilizan el estándar ERC-20 y su interfaz IERC-20	Utilización del estándar ERC-1155 y las interfaces asociadas. Además se utiliza la interfaz ERC-165
Aplicación completa, llevando a cabo el <i>front-end</i> y el <i>backend</i>	Centrado principalmente en el <i>backend</i> , con un <i>front</i> , que en este caso, solo sirve para que se vea de forma visual el <i>back</i>

Agradecimientos

En primer lugar me gustaría agradecer a los amigos y a mi familia, que son las personas cercanas que durante esta etapa, han estado apoyándome y ayudándome día a día. En segundo lugar me gustaría agradecer la ayuda, colaboración y por supuesto, en hacer los días de clase mucho más divertidos a los compañeros con los que he estado compartiendo la mayoría del tiempo estos años de la universidad.

Por otra parte agradecer a todo el claustro de profesores, que hemos tenido y que nos han ido impartiendo las asignaturas, por la dedicación, paciencia y profesionalidad que han demostrado en todo momento. Cabe destacar también el agradecimiento a Maite, por haberme llevado el TFG y por el tiempo dedicado, para ayudarme a elegir el tema y las correcciones sobre el mismo.

También agradecer al Banco Santander los medios dados, así como la asistencia técnica para poder llevar a cabo el proyecto y por los conocimientos que he ido adquiriendo en el último año y que espero poder seguir aprendiendo y trabajando con ellos durante más años.

Por último, agradecer a esos departamentos de la universidad, destacando el departamento de prácticas en empresas, que tienen menos visibilidad pero que sin ellos no es posible llevar a cabo el grado.

Bibliografía

1 ¿De dónde viene la lotería?

<https://elordenmundial.com/de-donde-viene-la-loteria/>

2022, El Orden Mundial en el Siglo XXI. Creative Commons BY-NC-ND

2 Historia de la lotería

<https://curiosfera-historia.com/origen-e-historia-de-la-loteria-inventor/>

© CurioSfera Historia 2022

3 ESPERANZA MATEMÁTICA DE LAS LOTERÍAS

<https://www.estadisticaparatodos.es/taller/loterias/esperanza.html>

Copyright Titapg 2008

4 LOTERIAS DE ESPAÑA

<https://www.estadisticaparatodos.es/taller/loterias/loterias.html>

Copyright Titapg 2008

5 PRIMITIVA Y BONO LOTO

<https://www.estadisticaparatodos.es/taller/loterias/primitivas.html>

Copyright Titapg 2008

6 EL GORDO DE LA PRIMITIVA

<https://www.estadisticaparatodos.es/taller/loterias/gordo.html>

Copyright Titapg 2008

7 EUROMILLONES

<https://www.estadisticaparatodos.es/taller/loterias/euromillon.html>

Copyright Titapg 2008

8 La Quiniela y el Quinigol

<https://www.estadisticaparatodos.es/taller/loterias/quinielas.html>

Copyleft Titapg 2008

9 La ONCE

<https://www.estadisticaparatodos.es/taller/loterias/once.html>

Copyleft Titapg 2008

10 Blockchain

<https://www.computerweekly.com/es/definicion/Blockchain>

TechTarget, [S.A de C.V 2021 - 2022](#)

11 ¿Cómo funciona Blockchain? Te explicamos la tecnología que está transformando las finanzas

<https://es.euronews.com/next/2021/10/21/como-funciona-blockchain-te-explicamos-la-tecnologia-que-esta-transformando-las-finanzas>

Copyright © euronews 2022

12 ¿Qué es la tecnología de blockchain?

<https://www.ibm.com/es-es/topics/what-is-blockchain>

IBM 2021

13 Smart Contracts: ¿Qué son, cómo funcionan y qué aportan?

<https://academy.bit2me.com/que-son-los-smart-contracts/>

Bitcoinforme S.L. 2022

14 Cuántos tipos de blockchain existen

<https://academy.bit2me.com/cuantos-tipos-de-blockchain-hay/>

Bitcoinforme S.L. 2022

15 ¿Qué es la Tolerancia a Fallas Bizantinas (BFT)?

<https://academy.bit2me.com/que-es-tolerancia-fallas-bizantinas-bft/>

Bitcoinforme S.L. 2022

16 ¿Qué son los ataques DoS?

<https://academy.bit2me.com/que-son-ataques-dos/>

Bitcoinforme S.L. 2022

17 ¿Qué es un Ataque del 51%?

<https://academy.bit2me.com/que-es-un-ataque-del-51/>

Bitcoinforme S.L. 2022

18 Tipos de blockchain: pública, privada e híbrida

<https://www.bsmexecutive.com/diferencias-entre-blockchain-publica-privada-e-hibrida/>

BSM GLOBELERS SL, B67370601 Copyright 2022 Blockchain School for Management

19 Blockchain, casos de uso más allá de las criptomonedas

<https://www.eae.es/actualidad/noticias/blockchain-casos-de-uso-mas-alla-de-las-criptomonedas>

EAE Octubre 2022

20 Los mejores casos de uso de blockchain

<https://kriptomat.io/es/blockchain/los-mejores-casos-de-uso-de-blockchain/>

Kriptomat 2021

21 ¿Qué es Chainlink?

<https://academy.bit2me.com/que-es-chainlink-link/>

Bitcoinforme S.L. 2022

22 Guía sobre los Oráculos Blockchain

<https://academy.binance.com/es/articles/blockchain-oracles-explained>

2022 Binance Academy

23 El VRF de Chainlink ya está disponible en Binance Smart Chain, trayendo consigo la aleatoriedad verificable a los desarrolladores de BSC

<https://medium.com/chainlink-community/el-vrf-de-chainlink-ya-est%C3%A1-disponible-en-binance-smart-chain-trayendo-consigo-la-aleatoriedad-e044ab789a4e>

Chainlink Community 2021

24 Chainlink VRF: On-Chain Verifiable Randomness

<https://blog.chain.link/chainlink-vrf-on-chain-verifiable-randomness/>

2021 SmartContract Chainlink Ltd SEZC

25 QUÉ ES LA COMPUTACIÓN CUÁNTICA

<https://www.iberdrola.com/innovacion/que-es-computacion-cuantica>

© 2022 Iberdrola, S.A. Reservados todos los derechos.

26 ¿Qué es quantum computing?

<https://www.ibm.com/es-es/topics/quantum-computing>

IBM 2022

27 Qué es un NFT y cómo funciona

<https://www.elmundo.es/como/2021/12/13/61b7100d21efa0c5488b4595.html>

El Mundo 2022

28 Qué es NFT

<https://www.arimetrics.com/glosario-digital/nft>

Arimetrics 2021

29 ¿Qué es NFT? Todo lo que necesita saber sobre los Tókenes No Fungibles

<https://ecommerce-platforms.com/es/glossary/what-is-nft>

2022 ecommerce-platforms.com | operado por Reeves and Sons Limited

30 Tókenes no fungibles (NFT)

<https://ethereum.org/es/nft/#what-are-nfts>

febrero de 2022 Ethereum.org


31 Explicación: ¿Qué es el hash en blockchain?

<https://learn.bybit.com/es/blockchain-es/que-es-hashing-blockchain/>

[Bybit Learn](#) 2021

32 Los conceptos de encapsulación, herencia, polimorfismo y composición de la programación orientada a objetos

<https://picodotdev.github.io/blog-bitix/2021/03/los-conceptos-de-encapsulacion-herencia-polimorfismo-y-composicion-de-la-programacion-orientada-a-objetos/>

Copyright © 2022 - 

33 Extending Contracts

<https://docs.openzeppelin.com/contracts/4.x/extending-contracts>

© OpenZeppelin 2017-2022

34 ERC20

<https://docs.openzeppelin.com/contracts/4.x/erc20>

© OpenZeppelin 2017-2022

35 ERC721

<https://docs.openzeppelin.com/contracts/4.x/erc721>

© OpenZeppelin 2017-2022

36 El metaverso explicado – Definición, introducción y ejemplos

<https://morethandigital.info/es/que-es-el-metaverso-definicion-introduccion-y-ambitos-de-aplicacion/>

© 2022 - [MoreThanDigital](https://morethandigital.info/)

37 ¿Qué es Decentraland (MANA)?

<https://x4prolab.com/que-es-decentraland-mana-criptomoneda/>

Copyright © 2021. Todos los derechos reservados. – Lab X4pro WEB DEV

38 La lista de metaversos blockchain que más dinero mueven en ventas

<https://observatorioblockchain.com/metaverso/la-lista-de-metaversos-blockchain-que-mas-dinero-mueven-en-ventas-3-800-millones/>

Jennifer Maldonado -2021

39 ¿Qué es Horizon Worlds? El primer metaverso de Facebook-Meta

<https://observatorioblockchain.com/metaverso/que-es-horizon-worlds-el-primer-metaverso-de-facebook-meta/>

Guillermo Callejo - 2022

Youtube: <https://www.youtube.com/watch?v=MM2E6we60ik> (Camino hacia el blockchain Jaime Gómez García)

Libro: Mastering Ethereum, AndreasM.Antonopoulos, Dr Garvin Wood

Bibliografía ilustraciones

- 1 <https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda>
- 2 <https://es.beincrypto.com/cardence-mejora-plataforma-multiples-servicios-oraculos-chainlink/>
- 3 <https://cdn.computerhoy.com/sites/navi.axelspringer.es/public/styles/480/public/media/image/2021/10/nft-2493697.jpg?itok=yF8DL-8B>
- 4 POO Lección 3: Conceptos principales – Beltramone Matías – Backend Developer (<https://cafecito.app/matibeltramone>) (matiasbeltramone.github.io)
- 5 <https://x4prolab.com/que-es-decentraland-mana-criptomoneda/>
- 6 [La lista de metaversos blockchain que más dinero mueven en ventas \(observatorioblockchain.com\)](https://observatorioblockchain.com)
- 7 [¿Qué es Horizon Worlds? El primer metaverso de Facebook-Meta \(observatorioblockchain.com\)](https://observatorioblockchain.com)