Desarrollo de una plataforma de crowdfunding distribuida sobre Ethereum

Autores:
Adrián Calvo María
Viktor Jacynycz García

Facultad de Informática
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo de Fin de Grado
Madrid, junio 2015

Directores:
Samer Hassan Collado
Antonio Sánchez Ruiz-Granados

Autorizamos a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el software desarrollado.

Adrián Calvo María *Viktor Jacynycz García*

Agradecimientos

En primer lugar, queremos agradecer a los directores del proyecto Samer Hassan Collado y Antonio Sánchez Ruiz-Granados por su tiempo, consejos y correcciones para la realización de este trabajo.

También queremos agradecer a los desarrolladores que contribuyen a la creación de Ethereum y la tecnología sobre la que se asienta este proyecto y a todas aquellas personas que dedican su tiempo a documentar y compartir su conocimiento sobre las mismas.

“It will be everywhere and the world will have to readjust. World governments will have to readjust.”

John McAfee, founder of McAfee, on Bitcoin Technology

Índice

Resumen

Betfunding es una plataforma de crowdfunding distribuida que usa apuestas en lugar de donaciones para promover la producción artística y el trabajo creativo.

A diferencia de otras plataformas de crowdfunding, en Betfunding son los inversores quienes proponen un proyecto y buscan creadores mediante la creación de una recompensa por su trabajo. Cualquier persona puede convertirse en creador de un proyecto si se compromete con los usuarios mediante una fianza. La evaluación de éxito o fracaso de un proyecto lo determinará un tercero de confianza designado al comienzo del proyecto, que puede ser una persona o un sistema automático.

La aplicación ha sido desarrollada sobre Ethereum, una tecnología similar a la que usa Bitcoin que permite gestionar dinero mediante criptomonedas y desarrollar aplicaciones descentralizadas. El uso de la tecnología de la cadena de bloques y criptomonedas evita la necesidad de una autoridad central de confianza que envíe y guarde el dinero cobrando comisiones. En su lugar, esta tarea se realiza a través de una red P2P de forma transparente y segura.

Palabras clave: Bitcoin, Blockchain, Cadena de bloques, Contrato inteligente, Criptomonedas, Crowdfunding, Cryptocurrencies, Ethereum, P2P, Smart contract, Solidity.

Abstract

Betfunding is a decentralized crowdfunding platform that uses bets instead of donations to support artistic production and creative work.

Unlike other crowdfunding platforms, in Betfunding the investors are the ones that suggest a project and look for creators by the creation of a bounty for their work. Anyone can become a creator of a project if she pays a deposit as a commitment to the users. The success or failure of a project will be determined by a trusted third party designated at the beginning of the project that can be a person or an automated system.

The platform has been developed on Ethereum, a Bitcoin-like technology that allows to manage money with cryptocurrencies and to develop decentralized applications. The use of blockchain technology and cryptocurrencies avoid the need for a trusted central authority to send and keep the money charging fees. Instead of that, it is done through a peer-to-peer network in a transparent and secure way.

Keywords: Bitcoin, Blockchain, Cadena de bloques, Contrato inteligente, Criptomonedas, Crowdfunding, Cryptocurrencies, Ethereum, P2P, Smart contract, Solidity.

Capítulo 1. Introducción

Muchos de los servicios que usamos a diario en Internet requieren que el usuario se conecte a un servidor para acceder a una aplicación en concreto. Por ejemplo, para acceder a páginas como Google o Facebook hay que pasar por alguno de los servidores en donde están alojados. Esto supone que **muchas de las aplicaciones y servicios online que usamos cada día están centralizados**.

La organización de la red según esta estructura conlleva varios problemas. El primer problema es el de la **seguridad**, si un servidor sufre un ataque, generalmente suele poner en peligro a todos sus usuarios. El segundo problema es el de la **confianza**, los usuarios no saben qué ocurre detrás de la aplicación una vez ha mandado sus peticiones. En una sociedad donde cada vez se confía información más sensible a estos sistemas, estos problemas pueden suponer grandes pérdidas a los usuarios.

Con la creación de Bitcoin, surgió la tecnología de la cadena de bloques que permitía a los usuarios llegar a un consenso de manera **distribuida y segura** sin necesidad de un servidor central. Años después, **Ethereum supo aprovechar esta tecnología** para permitir el desarrollo de aplicaciones distribuidas a todo tipo de usuarios con conocimientos técnicos.

El movimiento Bitcoin ha demostrado que mediante la fuerza de mecanismos de consenso y el respeto de los contratos sociales es posible un internet descentralizado, compartido a través de toda la red y simplemente con el coste de ofrecer el poder computacional de un ordenador para entrar en la red. Este sistema es **criptográficamente seguro** y basado en transacciones.

Ethereum es un proyecto que trata de construir una **plataforma para poder desarrollar este tipo de tecnología**, en la que todas las aplicaciones basadas en transacciones pueden ser implementadas. Además, no sólo provee la metodología para realizar dichas aplicaciones, sino que también ofrece programas de desarrollo para que cualquier persona pueda realizarlas.

Uno de los principales objetivos de Ethereum es facilitar las transacciones entre individuos que de otra manera no tendrían confianza entre ellos. Esta desconfianza puede ser debida a muchos factores desde una separación geográfica, cambio de moneda o incluso situación financiera del país de alguno de dichos individuos. Creando un sistema de transacciones **pseudo-anónimas,**

seguras y con un lenguaje **completo, no ambiguo y accesible para cualquier persona**, Ethereum nace para poner fin a estos problemas (1).

Por otro lado, desde la creación de las primeras empresas, el dinero necesario para que éstas salieran a flote provenía de inversores con ahorros, préstamos de entidades bancarias, o incluso de otras empresas con ganas de expandir su capital. Sin embargo, por ejemplo, Mozart y Beethoven financiaban sus conciertos y publicaciones de nuevos manuscritos a través de suscripciones de personas interesadas. Incluso La Estatua de la Libertad fue financiada por pequeñas donaciones provenientes de personas americanas y francesas. Estos dos ejemplos anteriores son solo una selección entre cientos de proyectos que han salido adelante gracias a la "**microfinanciación**" de una gran multitud (2).

Con el **crowdfunding**, un emprendedor puede conseguir una gran financiación de un **público muy extenso** en el cual cada uno aporta una **pequeña parte de la financiación total**, en vez de solicitar una pequeña cantidad de inversores sofisticados (3).

Motivaciones

Motivados por esta tecnología decidimos desarrollar una plataforma de financiación de proyectos a la que hemos bautizado como Betfunding. Esta plataforma **utiliza apuestas en vez de pujas** para incentivar la creación de proyectos innovadores, instando a los posibles inversores a que pongan dinero, no solo en proyectos que crean que se pueden realizar, sino también en proyectos con posibilidad de fracaso. Cualquier persona puede crear su proyecto y apostar en otros proyectos. Esta plataforma es **software libre, gratuita y carece de servidor centralizado.**

Objetivos

Los objetivos de este proyecto son:

- Aprender y experimentar con **aplicaciones distribuidas y criptomonedas** usando los lenguajes de programación de Ethereum.
- Promover la **creación de proyectos** a través de una **forma innovadora de crowdfunding** y **sin depender de estructuras centralizadas**.
- Incentivar el desarrollo de **software libre** utilizando una tecnología gratuita, modificable y accesible para cualquier persona.
- Sentar las bases para una gran diversidad de trabajos futuros, ya que es una tecnología innovadora y muy puntera, con gran cantidad de posibles aplicaciones.

Chapter 1. Introduction

Many of the services we use every day on the Internet require the user to connect to a server to access a particular application. For instance, to access sites like Google or Facebook you have to go through any of the servers where they are hosted. This means that **many of the applications and online services we use every day are centralized**.

The organization of the network according to this structure involves several problems. The first is a **security** problem, if a server is attacked, usually tends to endanger all users. The second is a **trust** problem, users do not know what happens behind the application once it has sent requests. In a society where increasingly sensitive to these systems are trusted information, these problems can mean big losses to users.

With the creation of Bitcoin, it emerged the blockchain technology, that allowed users to reach a **safe and distributed** consensus **without a centralized server**. Years later, **Ethereum took advantage of this technology** to enable the development of distributed applications to all users with certain technical knowledge.

The Bitcoin movement has shown that by the strength of mechanisms for consensus and respect for social contracts, a decentralized Internet is possible, shared throughout the network and simply by the cost of providing the computational power of a computer to enter the network. This system is **cryptographically secure** and based on transactions.

Ethereum is a project that aims to build **a platform to develop this technology**, in which all transaction-based applications can be implemented. Moreover, not only it provides the methodology for such applications, but also provides development programs that helps anyone who wants to develop these.

One of the main objectives of Ethereum is to facilitate transactions between individuals who otherwise wouldn't have trust between them. This distrust may be due to many factors from a geographical separation, exchange or financial situation of the country of any of these individuals. Creating a transaction system of **pseudo-anonymous**, **secure** and with a **full, unambiguous and accessible language for anyone**, Ethereum was created to stop these problems (1).

Since the creation of the first companies, the money needed to raise them came from investors, loans from banks, or even other companies wanting to expand their capital.

However, for instance, Mozart and Beethoven concerts financed their new manuscripts and publications through subscriptions from interested persons. Even the Statue of Liberty was financed by small donations from American and French people. These two examples above are just a selection from hundreds of projects that have gone ahead thanks to the "**microfinance**" of a large crowd (2).

With **crowdfunding**, an entrepreneur can get great financing from a **very wide audience** in which everyone contributes a **small part of the total funding**, instead of applying for a small number of sophisticated investors (3).

Motivations

Motivated by this technology we decided to develop a platform to which we have called Betfunding. This platform uses **bets instead of donations** to encourage the creation of innovative projects, urging potential investors to put money not only in projects that they think that can be performed, but also in projects with the possibility of failure. Anyone can create a project and bet on other projects. This platform is **free/open source software and decentralized**.

Goals

The objectives of this project are:

- Learning and experimenting with **distributed applications and cryptocurrencies** using Ethereum's programming languages.
- Promote the **creation of projects** through an **innovative platform of crowdfunding without relying on centralized structures.**
- Encourage the development of **free software** using a technology completely free, modifiable and accessible technology for anyone.
- Lay the groundwork for for a wide variety of future work, as it is an innovative technology and has many potential applications.

Capítulo 2. Estado del arte

Tanto el *crowdfunding* como las criptomonedas son **tecnologías muy recientes**. Ambos comenzaron a popularizarse a partir del año 2009 gracias a Kickstarter y Bitcoin respectivamente. A lo largo de este capítulo se repasa la historia y la evolución que han seguido ambas tecnologías hasta el día de hoy.

Crowdfunding

Se entiende por *crowdfunding* o micromecenazgo una forma de **financiación colectiva** en donde son los propios usuarios o clientes del producto a desarrollar quienes asumen el coste de su creación antes de su realización mediante pequeñas donaciones (3).

Normalmente la donación temprana a un proyecto viene acompañada de algún tipo de **recompensa para el donante**. Los modelos más comunes incluyen el envío de *merchandising* relacionado con el proyecto o la posibilidad de reservar el producto como recompensa, pero también es habitual el reparto de acciones o la devolución de lo donado a modo de préstamo pudiendo incorporar algún tipo de interés.

Aparte de para alcanzar una meta económica que permita la realización del proyecto anunciado, el modelo de financiación mediante *crowdfunding* también se emplea con regularidad para **sondear la posible aceptación** de una idea y crear una base de usuarios sólida mediante la difusión de la campaña de financiación por parte de los propios donantes interesados en que el proyecto salga adelante.

El origen del *crowdfunding* está en los sistemas de donaciones, aunque como puede verse en la figura 1, no fue hasta el año 2009 cuando comenzó a popularizarse para el público general y empezó a ser visto como **una alternativa** seria para la búsqueda de **financiación**. Dicho auge fue en parte propiciado por la situación de creciente crisis económica que limitó la emisión de préstamos por parte de los bancos hacia nuevos proyectos, y la adopción masiva de **Internet** y las **redes sociales**, facilitando la propagación de ideas y la búsqueda alternativa de financiación (4).

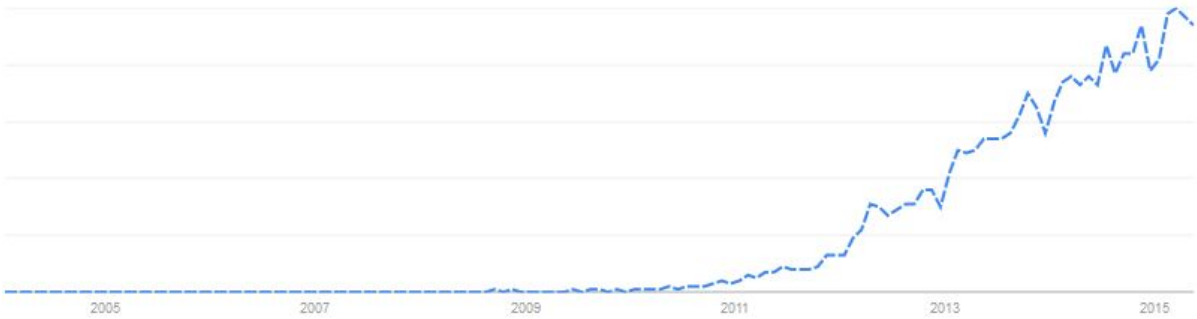


Figura 1. Búsquedas en Internet del término *crowdfunding (5)*

Actualmente el *crowdfunding* sigue siendo usado de forma mayoritaria por pequeños emprendedores que buscan una forma de crear un negocio pero no consiguen financiación para su idea por parte de un banco o prefieren **no depender de ningún fondo de inversiones**. Algunos ejemplos de financiación mediante *crowdfunding* exitosos son Star Citizen, un videojuego que lleva recaudados más de 80 millones de dólares mediante las aportaciones de más de 893.000 personas (6), Pebble Time, un *smartwatch* con una recaudación de más de 20 millones de dólares (7), o el propio Ethereum, la plataforma de desarrollo de aplicaciones descentralizadas basado en la cadena de bloques sobre la que está realizado este trabajo, cuya financiación se realizó completamente mediante bitcoins alcanzando la cifra de 31.531 BTC en 42 días, equivalentes a más de 18 millones de dólares en aquel momento (8).

Sin embargo, gracias a la popularización de la financiación colectiva durante estos últimos años también se han realizado importantes campañas de *crowdfunding* más allá de los proyectos comerciales. Algunos ejemplos notorios son la campaña a las elecciones presidenciales de los Estados Unidos del año 2008 de Barack Obama (9) o el partido político Podemos (10) en España.

Plataformas de *crowdfunding* vía web

La forma más popular de *crowdfunding* en la actualidad es la del *crowdfunding* vía web, que consiste en una página web donde la persona o grupo de personas interesados en realizar el proyecto publican su idea y buscan usuarios que les **financien a cambio de pequeñas recompensas.**

Los proyectos normalmente suelen estar agrupados en diferentes plataformas como Kickstarter (11) que facilitan la creación y gestión del mismo, permitiendo a los usuarios comenzar a recaudar fondos sin necesidad de conocimientos técnicos. Una vez el creador pública su proyecto, los usuarios pueden comenzar

a donar para que el proyecto salga adelante empleando un procesador de pagos como intermediario.

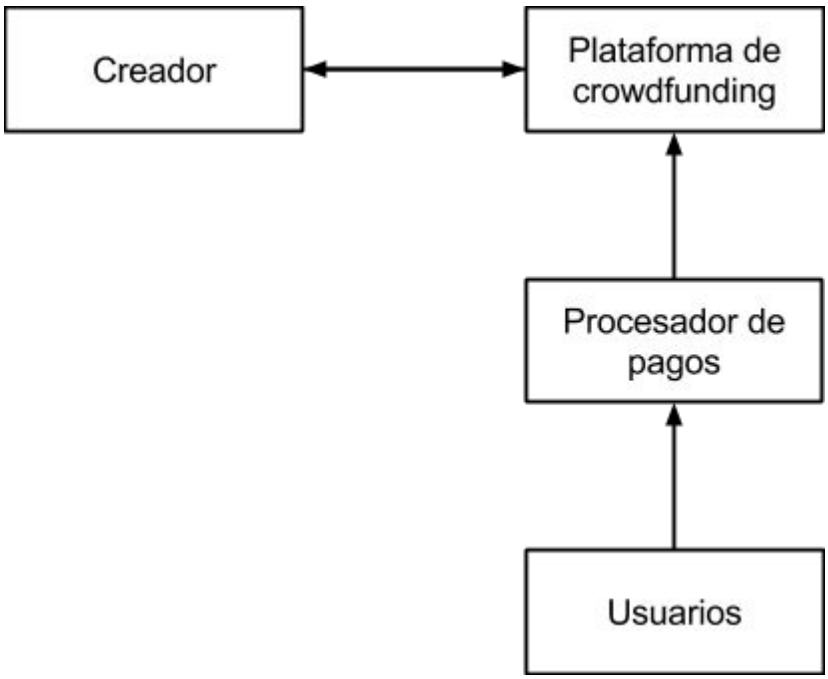


Figura 2. Esquema de funcionamiento de una plataforma de *crowdfunding* vía web

Como puede verse en la figura 2, estos sistemas están **centralizados** y requieren de la confianza tanto del creador como de los usuarios hacia las empresas encargadas de gestionar la plataforma y el procesador de pagos. También **requieren de la confianza por parte del usuario hacia el creador**, al no contar con **garantías** de la realización del proyecto una vez que se ha alcanzado la meta de financiación fijada. Tanto la plataforma como el procesador de pagos podrían tomar la decisión en cualquier momento de anular la campaña de financiación y retener indefinidamente los fondos aportados. El uso de intermediarios también implica la acumulación de comisiones sobre el dinero aportado por los usuarios.

En algunos casos, el creador del proyecto y el administrador de la página que gestiona el *crowdfunding* son la misma persona, lo cual elimina la intermediación de una tercera empresa en el proceso de recolección de fondos. En estos casos, el creador aloja en un servidor bajo su control la página con la información de la campaña de financiación y emplea una cuenta bancaria propia para guardar los fondos en lo que dura la campaña. Esta opción no está al alcance de cualquier persona interesada en financiar su propio proyecto, puesto que requiere de amplios conocimientos técnicos y un significativo esfuerzo para la correcta

gestión de los fondos. De todos modos, este sistema no soluciona el **problema de la confianza**, sólo lo traslada hacia un problema de confianza entre los usuarios y el creador.

Para **eliminar la intermediación del procesador de pagos** se puede recurrir al uso de **criptomonedas**, de modo que los usuarios envíen el dinero a través de sistemas como Bitcoin y empleen páginas como Coinfunder (12) para la gestión de sus proyectos. El uso de criptomonedas elimina tanto la necesidad de confianza con el procesador de pagos como el cobro abusivo de comisiones por las transacciones de poco valor.

Arquitecturas Distribuidas

Una de los conceptos más importantes cuando se habla de Ethereum es el de arquitectura distribuida. Propone eliminar la centralización quitando la dependencia de un servidor centralizado.

Existen tres tipos importantes de estructuras (ver figura 3):

- **Estructura Centralizada**: Toda la estructura está gestionada por **un solo nodo** y sus usuarios pertenecen a la misma comunidad. Se utiliza principalmente en **servicios web**, alojadas en un servidor centralizado por el que tienen que pasar todas las personas que quieran acceder a ella (e.g. Wikipedia, Airbnb, Github).

- **Estructura Federada**: La infraestructura está dividida en **varios nodos operativos** que funcionan como su propia estructura centralizada, fragmentado el servidor central en pequeños servidores distribuidos. Cada uno de estos tiene su **propio dueño y su comunidad.** A pesar de esto, cualquier usuario de la estructura puede acceder a los datos de otros independientemente del nodo al que pertenezcan (e.g. GNU Social, Buddycloud, Diaspora).

- **Estructura P2P** (*Peer to Peer*): Es una estructura **totalmente distribuida,** particionando los trabajos y la información entre los usuarios de la red llamados “Peers”. Cada uno de estos **tienen los mismos privilegios** en la infraestructura. Cada usuario controla su aportación a la red distribuida y generalmente **todos pertenecen a la misma comunidad** (e.g. BitTorrent, Twister, Bitcoin, Ethereum).

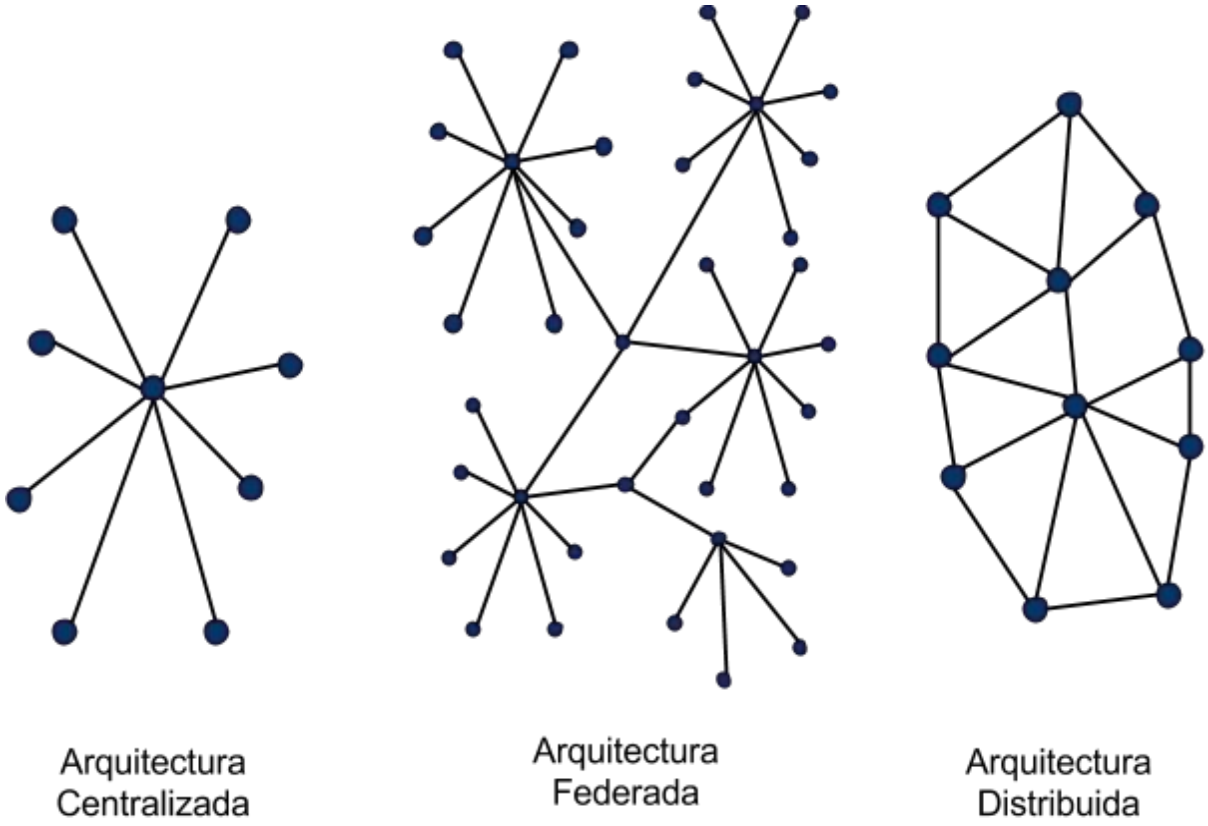


Figura 3. Representación gráfica de las arquitecturas de red (13)

Uno de los ejemplos más importantes de una arquitectura distribuida P2P es el de **Bitcoin**, en el que todos los usuarios ven todas las transacciones y tienen los mismos derechos.

Ethereum utiliza también la arquitectura P2P al igual que Bitcoin consiguiendo así que todo el software desarrollado en este lenguaje sea **100% libre y descentralizado**, ya que todos los usuarios de la red tienen acceso a todos los contratos de la cadena de bloques y al código fuente de estos**.**

Orígenes de Ethereum

Moneda Descentralizada

En la década de los 80 y los 90 empezaron a aparecer formas de pago descentralizadas como *ecash* (14) que ofrecían una **moneda con un alto nivel de privacidad**; fue entonces cuando empezó a surgir el concepto del "dinero electrónico anónimo".

Uno de los pioneros en la creación de una moneda electrónica descentralizada fue Wei Dai, que en 1998 publicó su propuesta de dinero electrónico llamada B-money (15). Wei Dai proponía la creación de dinero mediante la resolución de **puzles computacionales**. A partir de la idea de Wei Dai han ido surgiendo otras propuestas como Bit gold (16) para mejorar la implementación de una criptomoneda haciendo uso de RPOW (17), una extensión del sistema de prueba de trabajo de Hashcash (18).

No fue hasta el año 2009 que se implementó por primera vez una **moneda descentralizada,** cuando Satoshi Nakamoto publicó la primera versión de Bitcoin (19). Esta moneda tenía como objetivo crear un sistema de pagos electrónicos completamente descentralizado, empleando **pruebas criptográficas** en lugar de confianza mediante un sistema de *proof of work* o prueba de trabajo. Este mecanismo criptográfico resolvía dos problemas. Primero, ofrecía una forma efectiva de alcanzar un **consenso entre todos los nodos** de la red sobre el estado de la misma. Segundo, ofrecía la posibilidad de que **cualquier persona** podía unirse a la red de nodos, resolviendo los problemas políticos que implican el hecho de quién llevaba el control de la red de nodos.

Bitcoin

Bitcoin es una **moneda virtual** que se **basaba en un sistema de transacciones** donde existe un "estado", que consiste en el estatus de todas las transacciones de los dueños de las monedas. De esta forma, la posesión de bitcoins no implica tener una moneda virtual, sino la existencia de un listado de transacciones en la que en la última persona es el beneficiario. En la figura 4 se puede ver un ejemplo del funcionamiento de las transacciones en Bitcoin y cómo estas cambian el estado mediante la transferencia de bitcoins.

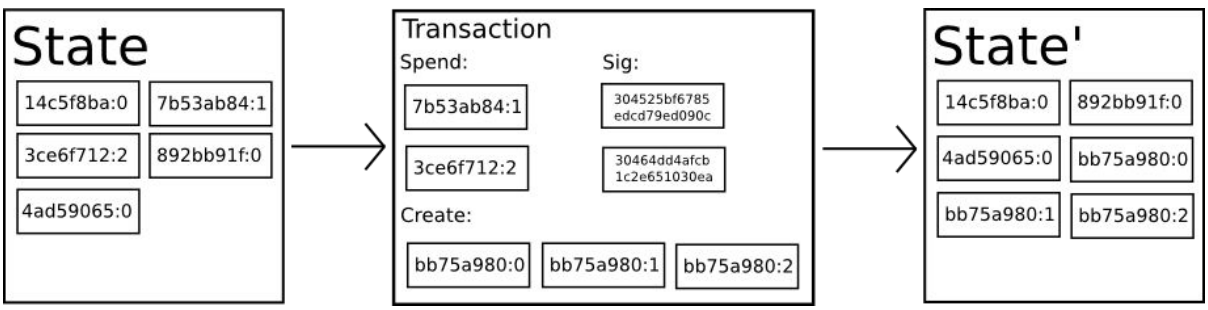


Figura 4. Funcionamiento de las transacciones en Bitcoin (20)

Para que todo esto funcione se creó un sistema conocido como *Blockchain* o cadena de bloques (21), un registro público y accesible por todos los nodos de la

red donde se almacena el historial de todas las transacciones. De esta manera se hace **prácticamente imposible falsear una transacción**, ya que si alguien intenta engañar a un nodo para hacer creer al sistema que tiene más dinero del que realmente posee, al sincronizar la información de la transacción, esta sería visible por el resto de la red y rechazada automáticamente al no poder demostrar la posesión de dichos bitcoins.

Cadena de bloques

La cadena de bloques es un tipo de **base de datos distribuida** que almacena la información en forma de bloques. Cada bloque guarda **información sobre las transacciones realizadas**, el tiempo en que el bloque fue añadido a la cadena, y hash del bloque anterior (ver figura 5). De esta forma, la validez de las transacciones futuras puede ser verificada consultando el último estado de la dirección de envío.

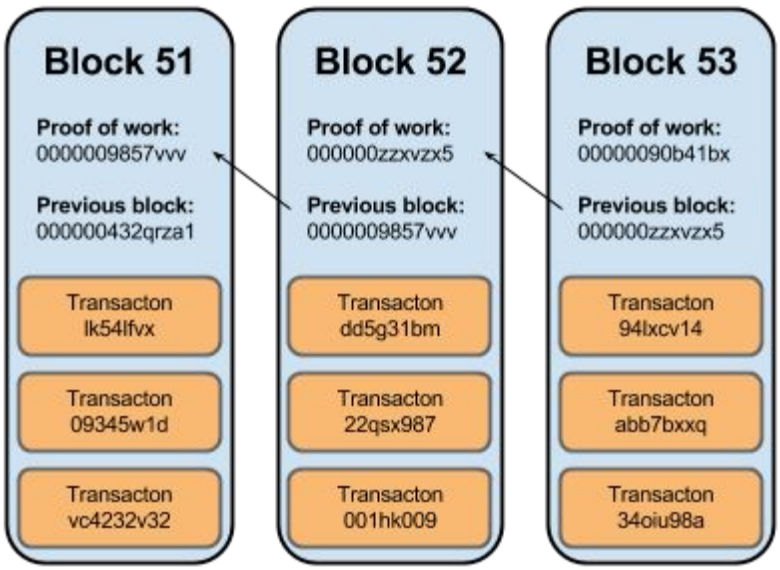


Figura 5. Representación de la cadena de bloques (22)

La inclusión del próximo bloque de la cadena se realiza mediante un sistema de **prueba de trabajo** conocido como “minar” (23). La minería permite a los nodos de la red alcanzar un consenso de forma distribuida. Los diferentes nodos de la red compiten mediante potencia de cálculo computacional por resolver un problema matemático. El nodo que resulte ganador es el encargado de formar el siguiente bloque con las transacciones pendientes de verificar y añadirlo a la cadena. Este sistema también permite la emisión de nueva moneda mediante la generación del *coinbase* (24) de cada bloque, una pequeña cantidad de moneda

que va a parar al generador del bloque junto con la comisión pagada por los usuarios cuyas transacciones han sido incluidas en él.

Altcoins

Desde la creación de Bitcoin han aparecido gran cantidad de monedas basadas en la misma tecnología de cadena de bloques.

En febrero de 2011 (25), con el aumento del valor de los bitcoins se tomó la decisión de crear una criptomoneda paralela con el objetivo de realizar ahí las pruebas sin poner en peligro la red de Bitcoin. De esta forma se creó Bitcoin Testnet, que puede ser **considerada como la primera altcoin**. Bitcoin Testnet comparte todas las características de Bitcoin puesto que su objetivo es servir como **zona de pruebas** (26).

En abril de 2011 se creó Namecoin con el objetivo de crear un sistema de DNS descentralizado y complicar así la censura de páginas web (27). Poco después, en octubre de ese mismo año se creó Litecoin, una criptomoneda **similar a Bitcoin pero con algunas características diferentes**, como un mayor número de unidades, **menos tiempo entre cada bloque** de la cadena y un algoritmo diferente para alcanzar el consenso entre los nodos de la red (28).

Desde entonces, se han ido creando nuevas criptomonedas hasta alcanzar las cientos de altcoins que hay en la actualidad (ver figura 6). La mayoría de estas monedas son copias de Bitcoin o Litecoin que han modificado alguna de sus características como el número de monedas en circulación o el tiempo que tarda en confirmarse una transacción, pero hay unas pocas que han innovado mediante la creación de nuevos algoritmos de minado y la inclusión de nuevas funcionalidades.

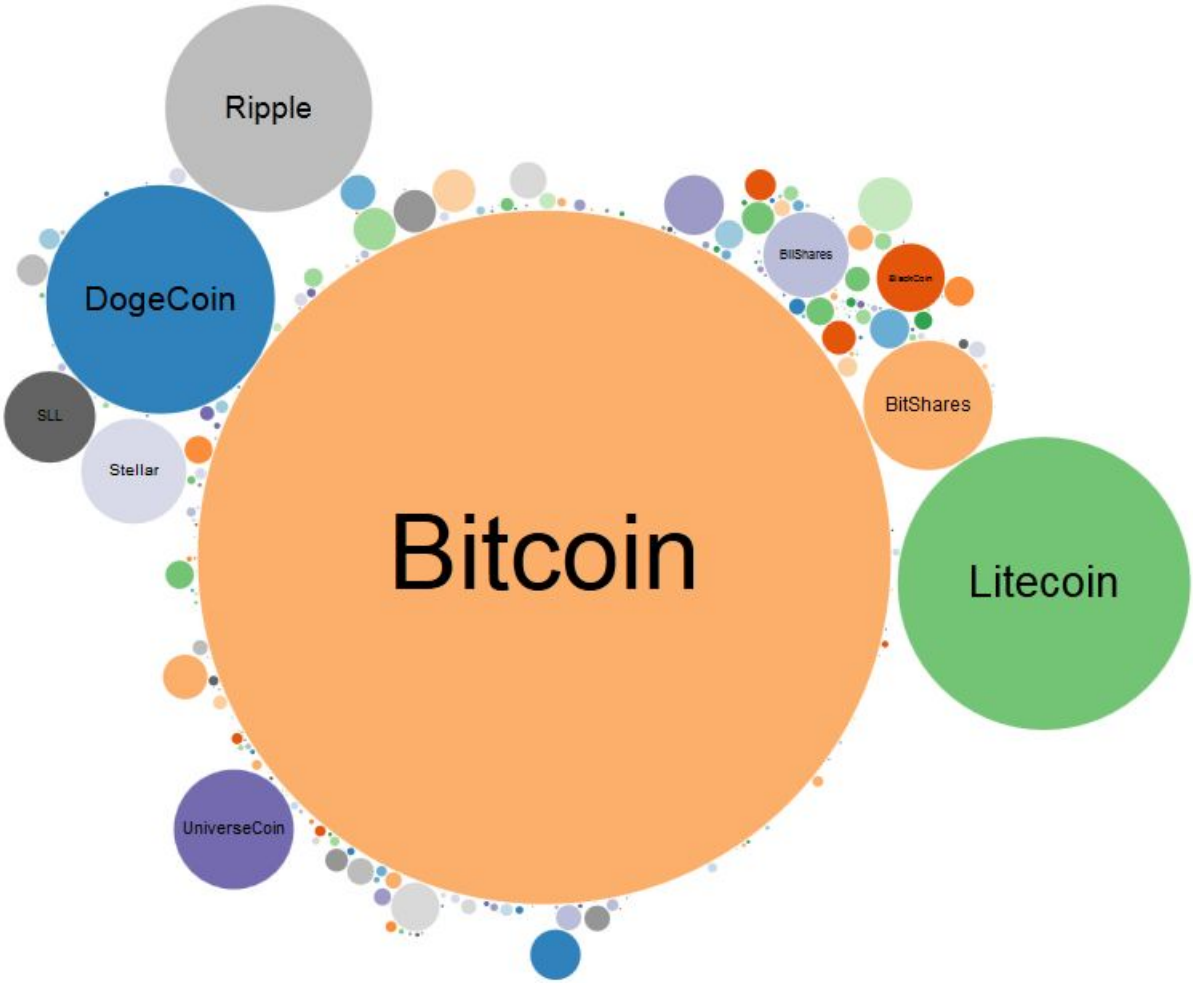


Figura 6. Criptomonedas existentes representadas por volumen de compra-venta (29)

La principal innovación que han traído las altcoins son los **diferentes métodos** de alcanzar un consenso sobre qué nodo es el siguiente en **verificar la validez** de las transacciones mediante su inclusión en el siguiente bloque de la cadena y la generación de moneda.

- *Proof of work*: Requiere la resolución de problemas matemáticos mediante fuerza bruta. Se emplean problemas difíciles de computar pero fáciles de verificar para comprobar que quien obtiene la solución ha realizado una cantidad significativa de trabajo computacional (30). Algunas monedas que emplean este sistema pero con diferentes algoritmos de minado son Bitcoin, Litecoin o Primecoin (31).

- *Proof of stake*: En lugar de resolver problemas de dudosa utilidad con el coste energético y medioambiental que ello supone, la probabilidad de ser elegido el encargado de crear el siguiente bloque de la cadena viene determinado por el número de monedas que se posee. De esta forma,

quienes más tienen generan una mayor cantidad de monedas (32). Algunos ejemplos de altcoins que usan Proof of Stake son Nxt o Peercoin. Es habitual que Proof of stake se use junto a Proof of work para crear algoritmos más seguros y con menor consumo de recursos. Faircoin (33) o Ethereum son ejemplos del uso de algoritmos híbridos.

- *Proof of resources*: Las nuevas monedas generadas se reparten en función de su contribución hacia la red. Un ejemplo de esta implementación es Safecoin (34), donde los usuarios minan monedas en función de su aportación en forma de ancho de banda o espacio en disco a la red con el objetivo de descentralizar Internet.

- *Proof of burn*: Las nuevas monedas se generan mediante la eliminación de otra moneda anterior. Para ello, los usuarios deben probar esta eliminación mediante el envío de una transacción a una dirección sin retorno (35). Una de las monedas que emplea este sistema es Counterparty (36).

Bitcoin 2.0

El concepto de cadena de bloques nació junto con Bitcoin para llevar un **registro de todas las transacciones realizadas y evitar los dobles gastos del dinero**. Esta tecnología resolvió el problema de alcanzar un consenso entre varias partes dentro de una red descentralizada. Tras Bitcoin, otros desarrolladores han ampliado esta tecnología con el objetivo de dar solución a diferentes problemas dando lugar al surgimiento del movimiento Bitcoin 2.0.

La primera moneda que surgió para ampliar las funcionalidades de la cadena de bloques fue Namecoin, que nació con el objetivo de crear un **servicio de DNS descentralizado para los dominios ".bit"**. En Namecoin, la cadena de bloques almacena la información sobre los dominios y su historial de registros. Esta cadena de bloques también puede usarse para almacenar otro tipo de información. Por ejemplo, los proyectos Onename (37) y Passcard (38) emplean la cadena de bloques de Namecoin para almacenar objetos JSON con un perfil público de sus usuarios y así crear un sistema de identificación distribuido que sustituya los sistemas de registro de la web. Inspirados por Namecoin y el concepto de emplear la cadena de bloques como base de datos distribuida, surgieron servicios como Proof of Existence (39) y BTProof (40), que funcionan a modo de **notario digital,** usan la cadena de bloques para **certificar y probar la existencia de un documento** en un momento dado. Para ello, crean un hash

del documento seleccionado y lo almacenan en la cadena de bloques mediante una transacción.

De esta forma nació el concepto de Metacoin, aplicaciones que usan la cadena de bloques para **ampliar sus funcionalidades mediante la inclusión de metadatos**. Un ejemplo de metacoins es Colored coins, que emplea la cadena de bloques de Bitcoin ya existente para permitir la emisión de participaciones o la creación de monedas alternativas. Su funcionamiento consiste en la inclusión de información adicional a un conjunto de bitcoins "coloreándolos" para que pasen a representar algún activo que quiera ser transferido. Luego esos bitcoins pueden ser transferidos a través de la cadena de bloques, transfiriendo así el activo (41). Otro ejemplo significativo que añade nuevas funcionalidades a Bitcoin es Zerocoin, un proyecto aún en desarrollo que mediante la inclusión de datos adicionales en la cadena de bloques de Bitcoin permite dotar a los usuarios de verdadero anonimato al realizar las transacciones (42).

Para usos más avanzados de la cadena de bloques, algunos desarrolladores han comenzado a crear nuevos protocolos que funcionan como **una capa por encima de Bitcoin**. El primero de estos desarrollos fue Mastercoin, que permite la emisión e intercambio de monedas con diferentes características de manera descentralizada. Mastercoin no tuvo el éxito esperado y actualmente se sigue su desarrollo bajo el nombre de Omni (43), centrándose en el concepto de *Smart Property*, que permite **representar mediante monedas un activo y transferirlo a través de la cadena de bloques** (44). Otro ejemplo más reciente es Storj, una red P2P de almacenamiento de ficheros que emplea la cadena de bloques y recompensa a sus usuarios por el uso del espacio de su disco. Los ficheros subidos a la red se dividen en trozos, se cifran, y se propagan por la red. La cadena de bloques se encarga de llevar un registro de la integridad y la disponibilidad de los ficheros y sincronizar la información entre los usuarios (45).

Muchas de estas aplicaciones resolvían problemas concretos, pero normalmente estos desarrollos **requieren de la creación de un altcoin adicional** y sus propias reglas de funcionamiento, obligando a los usuarios a hacer uso de un conjunto de altcoins y **protocolos diferentes**. Para evitar esto, en la actualidad están surgiendo diferentes proyectos que buscan crear una plataforma de desarrollo de aplicaciones descentralizadas basadas en la cadena de bloques bajo un **mismo protocolo** empleando el concepto de *Smart contract* o Contrato inteligente (46). Un contrato inteligente es un acuerdo contractual implementado mediante software. Al contrario que un contrato tradicional, en donde las partes deben recurrir al sistema legal para asegurar el cumplimiento del mismo, un contrato inteligente es capaz de **auto-ejecutarse una vez se hayan cumplido las condiciones (47)**. Por la forma en que ha sido desarrollado

Bitcoin, es posible incorporar código a una transacción mediante un lenguaje de script que dicte bajo qué condiciones debe ejecutarse la transacción (48). Proyectos como Counterparty emplean esta característica dentro de Bitcoin para extender su funcionalidad y permitir el registro de activos, emisión de dividendos o creación de contratos inteligentes (49), mientras que otros como Ethereum han decidido crear **su propia cadena de bloques** y ampliar este lenguaje de script para facilitar el desarrollo de contratos a los programadores.

Actualmente Counterparty y Ethereum son los dos principales proyectos que compiten por convertirse en la plataforma de desarrollo de aplicaciones descentralizadas conocidas como DAO (*Decentralized Autonomous Corporations*) o DApps (50) de referencia. Existen otras opciones como Maidsafe (51), Codious (52), Nxt (53) o Eris (54), pero de momento se encuentran en desarrollo y no cuentan con una comunidad de usuarios enfocada en el desarrollo de DApps como las anteriores. Counterparty emplea la cadena de Bloques de Bitcoin mientras que Ethereum decidió crear una nueva sólo para su plataforma; ambas opciones tienen sus pros y sus contras. Al emplear la cadena de bloques de Bitcoin, **Counterparty** parte con una mayor base de usuarios potenciales y mejor interacción con Bitcoin, pero al mismo tiempo arrastra algunas de sus debilidades como las **limitaciones de su lenguaje de script** o el limitado tamaño de cada bloque. Además, debe hacer uso de la cadena de bloques completa de Bitcoin limitando su escalabilidad. **Ethereum**, por su parte, aprovecha la creación de su propia cadena de bloques para **incorporar un lenguaje de script más avanzado**, mayor tamaño de bloque y un algoritmo diferente de hash para el minado que permite bloques cada 12 segundos de media (frente al tiempo de bloque Bitcoin que crece de manera logarítmica). Además, complementa su cadena de bloques con otros dos protocolos para el intercambio de ficheros (Swarm (55)) y mensajes (Whisper (56)) para así crear un entorno completo de desarrollo.

Lighthouse

Lighthouse es una plataforma de ***crowdfunding*** **descentralizada** que emplea bitcoins como sistema de pago y la tecnología de Bitcoin para la creación de contratos inteligentes en la cadena de bloques.

Lighthouse imita el funcionamiento de plataformas como Kickstarter, en donde el creador de un proyecto publica su idea en busca de financiación por parte de los de usuarios. El creador establece una cantidad de dinero necesaria para la realización de su proyecto, de modo que cuando dicha condición se cumple el dinero se desbloquea automáticamente y pasa a manos del creador para que

lleve a cabo el proyecto prometido. Hasta que la condición se cumpla, los usuarios mantendrán el control sobre su dinero pudiendo desvincularse de la financiación del proyecto en cualquier momento.

La innovación de Lighthouse frente a otras plataformas de *crowdfunding* tradicional o *crowdfunding* con criptomonedas reside en su descentralización mediante la eliminación total de intermediarios sin que ello suponga transferir la confianza de tu dinero sobre el creador durante el proceso de recolección de fondos, **evitando las comisiones de estos servicios**. En lugar de una página alojada en un servidor y una cuenta bancaria que recolecte los fondos de forma centralizada, **Lighthouse es en realidad una cartera de Bitcoin modificada**. La cartera es capaz de interpretar la información sobre los proyectos y sincronizarse con la cadena de bloques de Bitcoin, de modo que toda la **información** sobre el proyecto **se consulta en el propio ordenador del usuario** y la gestión del dinero se realiza en la cadena de bloques mediante la creación de una transacción con propiedades de contrato inteligente que monitoriza los fondos del proyecto con capacidad para desbloquearlos si cumplen las condiciones establecidas (57).

Sin embargo, el protocolo de Bitcoin fue originalmente diseñado para el envío de transacciones simples, de modo que su uso para la creación de contratos inteligentes complejos o que impliquen un gran número de datos **no es recomendable**. En el caso de Lighthouse, las condiciones posibles para ser establecidas en el contrato inteligente son muy limitadas, limitado únicamente a la cantidad de fondos recaudados en la versión actual. También se ven limitados el número de posibles donantes a un proyecto, cuyo tope está establecido en 684 personas debido al reducido tamaño de bloque de Bitcoin (58). Esta cifra queda muy lejos de las 893.000 personas que han llegado a participar en alguna de las campañas de *crowdfunding* mencionadas anteriormente.

Capítulo 3. Metodología y tecnología

Dado que Ethereum es una plataforma completamente nueva, conlleva la utilización de tecnología proporcionada por los creadores, que en muchos casos está en **una fase muy temprana del desarrollo**. A continuación se detallan los programas y plataformas utilizadas en la realización del proyecto y los métodos de trabajo del equipo encargado de llevarlo a cabo.

Metodologías

Plan de Trabajo

Debido a la naturaleza experimental de Ethereum (ya que está en desarrollo y planea tener la primera versión estable a lo largo de 2015), el plan de trabajo consistió en **investigar semanalmente** cómo evolucionaba el lenguaje de programación de Ethereum. A medida que aprendíamos a desarrollar aplicaciones, el equipo se reunía cada dos semanas para programar. También realizamos reuniones con los tutores del trabajo cada tres semanas para ir haciendo un reporte de los avances que habíamos conseguido, y para que nos aconsejaran cual era el siguiente paso a seguir. Cada vez que se conseguía algún avance, se informaba a todo el mundo mediante correo electrónico, sistema de comunicación que se empleaba también para agendar las reuniones.

Primero hicimos pequeños programas de ejemplo para **aprender cómo funcionaba internamente Ethereum.** Una vez aprendido el funcionamiento de la plataforma, pasamos a mejorar el aspecto gráfico de algunas de las aplicaciones de prueba que realizamos, ya que la idea final era que Betfunding fuera **atractiva para el usuario**. Finalmente procedimos a la implementación de la plataforma de crowdfunding según las especificaciones detalladas en esta memoria.

Enfoque de software libre

Debido a que Ethereum es una plataforma para **promover las aplicaciones distribuidas** en Internet excluyendo a los servidores centralizados, pensamos que lo más lógico es que todo el software desarrollado y este documento sean libres.

El código fuente de todas las aplicaciones mencionadas en este documento es accesible a través de la organización de GitHub:

https://github.com/EthereumUCM

La licencia elegida para nuestro software es **GPLv3,** ya que el usuario tiene la capacidad de acceder a todo el código fuente y forzando el hecho de que cualquier trabajo derivado también sea GPLv3.

Prototipado rápido

Uno de los principales problemas que encontramos al empezar este proyecto es que el **lenguaje de programación de contratos no estaba del todo desarrollado**. Para solucionarlo decidimos utilizar prototipado rápido como metodología de desarrollo. El objetivo de este sistema es poder crear **pequeños prototipos en ciclos cortos** e ir mejorándolos en cada una de las iteraciones del método. De esta manera, a medida que Ethereum evolucionaba, éramos capaces de crear pequeños programas de prueba para ver los cambios en el lenguaje de programación y poder trasladarlos al prototipo final.

Durante las fases de desarrollo de Betfunding, **Ethereum experimentó un cambio importante** en su forma de programar contratos, ya que pasó de un lenguaje llamado **Serpent (59)** basado en Python, a uno bastante más complejo y completo llamado **Solidity (60)**. Gracias a la metodología de prototipado rápido **pudimos adaptarnos a ese cambio**.

Contribuciones al proyecto

Adrián Calvo María

Los primeros meses de realización del proyecto los dediqué a explorar las funcionalidades de la plataforma de Ethereum y el lenguaje de programación Serpent.

Durante ese periodo realicé pequeños programas como un contador y su correspondiente interfaz gráfica. Esta primera aplicación era muy sencilla y tenía como único propósito probar el entorno de desarrollo para ver cómo se almacenaba la información en la cadena de bloques y cómo ésta podía ser manipulada empleando la librería web3 de Ethereum.

Poco después comencé con la realización de ETHChat, una sala de chat distribuida sobre la cadena de bloques. La primera versión de esta aplicación consistía en un solo contrato programado en Serpent y una interfaz gráfica usando Bootstrap. Con esta aplicación buscada comprobar la viabilidad de Serpent para la realización de aplicaciones más complejas que pudieran ser usadas simultáneamente por varios usuarios.

En torno al mes de diciembre, realicé una nueva versión de ETHChat que incluía dos nuevos contratos que se comunicaban entre sí para permitir el registro de nombres de usuario.

Durante los primeros meses de 2015 realizamos las dos implementaciones del juego de apuestas, yo me encargué del desarrollo de la versión en Serpent mediante dos contratos.

Finalmente realizamos Betfunding, la plataforma de crowdfunding distribuida, que estará dividida en una interfaz web y el código de la cadena de bloques en Solidity. En un primer momento me encargué de la primera especificación de las funciones y atributos del contrato. Luego repartimos el trabajo y realicé el diseño y el desarrollo de la interfaz web completa.

Una vez tuvimos una primera versión de la interfaz web y el contrato, conecté ambas partes mediante la librería web3 corrigiendo los errores y ampliando tanto la parte web como el contrato en Solidity para conseguir una aplicación funcional. Dicho trabajo incluye la realización de los algoritmos de reparto del contrato en Solidity y el testeo completo de la aplicación. También creé un juez automático que permite la verificación de los proyectos mediante información alojada en la cadena de bloques, implementado mediante un contrato en Solidity.

Todas las aportaciones aquí mencionadas pueden consultarse detalladamente desde las siguientes direcciones:

- https://github.com/AdrianClv/ethereum/commits/master/counter
- https://github.com/AdrianClv/ethereum/commits/master/chat
- https://github.com/AdrianClv/ethereum/tree/master/chat-v2
- https://github.com/EthereumUCM/serpentGamblingGame/commits/master
- https://github.com/EthereumUCM/Betfunding/commits/master

Respecto a la memoria, ha sido realizada de forma colaborativa entre los dos componentes del grupo empleando Google docs, de modo que ambos hemos

participado mediante la redacción y corrección del documento, siendo mi aportación principal los siguientes apartados:

- Resumen/Abstract
- Capítulo 2: estado del arte, crowdfunding, plataformas de crowdfunding vía web, cadena de bloques, altcoins, Bitcoin 2.0 y Lighthouse.
- Capítulo 3: Serpent, Solidity, Bootstrap y otras tecnologías.
- Capítulo 4: completo.
- Capítulo 5: introducción, especificación, requisitos funcionales y no funcionales, tipos de usuario, crowdfunding mediante apuestas, algoritmo y gráfico del ejemplo, estructura de la aplicación y ejemplo de uso.
- Capítulo 6: completo.
- Bibliografía.

Viktor Jacynycz García

Al comienzo del proyecto, la tecnología todavía estaba muy verde como para poder desarrollar programas, así que dediqué gran parte del tiempo en investigar el funcionamiento interno de la cadena de bloques.

En el primer trimestre también estuve haciendo pequeñas pruebas con pequeños fragmentos de código (programado en Serpent), para ver cómo funcionaba el sistema para subir contratos, y cómo se interactuaba con ellos una vez en la red de Ethereum.

A comienzos del 2015 se hizo pública una nueva versión del lenguaje de programación que estaba orientada a objetos, llamada Solidity, y comencé a desarrollar bastantes fragmentos de código para ver la sintaxis y la definición de tipos. Estos fragmentos de código se podían compilar en tiempo real en la web https://chriseth.github.io/cpp-ethereum/ para comprobar si está bien escrito un contrato.

Después de aprender a programar en Solidity, implementé la primera versión del juego de apuestas en este lenguaje para ver la diferencia entre éeste y su predecesor Serpent, basado en Python.

Una vez completado el juego de apuestas decidimos pasar a la implementación del prototipo inicial de Betfunding. Inicialmente yo me encargué de hacer la parte de Solidity, creando una primera versión con bastantes fallos.

Después de varias optimizaciones del código, realicé la segunda versión de Betfunding en la que creaba contratos externos para cada uno de los proyectos, pero luego vimos que esto no era viable.

Finalmente realicé una última implementación de Betfunding que conectamos más tarde con la parte de frontend que había realizado mi compañero.

Todos las aportaciones realizadas por los integrantes del proyecto se pueden consultar en https://github.com/EthereumUCM/

Como ha mencionado mi compañero con anterioridad, la memoria ha sido realizada de forma colaborativa con Google docs, aportando cada uno su parte en la redacción y corrección del documento. Principalmente mi aportación ha sido:

- Capítulo 1: completo.
- Capítulo 2: arquitecturas distribuidas, moneda descentralizada, Bitcoin y cadena de bloques.
- Capítulo 3: plan de trabajo, enfoque de software libre, prototipado rápido, Ethereum y AlethZero.
- Capítulo 4: revisión y conclusiones individuales de cada prototipo.
- Capítulo 5: crowdfunding mediante apuestas - ejemplo, diagrama de casos de uso, organización de un proyecto en el contrato, estructura de un proyecto en el contrato y mockups.
- Capítulo 6: Gran parte de la ampliación del capítulo para tener una visión más completa de las conclusiones del proyecto.
- Bibliografía.

Finalmente repasé toda la memoria, mejorando el aspecto visual de los párrafos y los títulos y poniendo en negrita los conceptos importantes para facilitar la lectura.

Tecnologías

Nuestra aplicación final está dividida en un **contrato y una interfaz web**. Para el desarrollo del contrato se han empleado lenguajes de programación, tecnologías y aplicaciones que forman parte de Ethereum. La interfaz web es independiente y se ha desarrollado mediante tecnologías web del lado del cliente como HTML, CSS, JavaScript y Bootstrap. La conexión entre ambas partes se realiza mediante el uso de una librería proporcionada por la propia plataforma de Ethereum.

Ethereum

Ethereum nace inspirado de muchos de los conceptos de Bitcoin 2.0 antes mencionados para ofrecer a cualquier usuario una herramienta para **desarrollar aplicaciones descentralizadas y seguras de forma sencilla.**

El funcionamiento interno es muy parecido al de bitcoin, con **su propia criptomoneda** llamada "ether" (61). Para la creación de aplicaciones primero se escribe un contrato mediante código y se sube a la cadena de bloques mediante una transacción. Una vez en la cadena de bloques, el contrato tiene una dirección desde la cual se puede interactuar con él.

Ethereum tiene su propio lenguaje de programación con el que desarrollar aplicaciones como plataformas de *crowdfunding*, casas de apuestas o casinos. Todas estas aplicaciones son distribuidas, lo cual implica que **cualquier persona puede crear un contrato** en Ethereum sin necesidad de un servidor centralizado.

Cuentas en Ethereum

La forma de identificarse en la red de Ethereum es mediante el uso de **cuentas**. Una cuenta es una dirección desde donde el usuario puede interactuar con el resto de la red. Una dirección en Ethereum puede **identificar a un usuario** de la red o a **un contrato** almacenado en la cadena de bloques:

- **Usuarios**: Son todos las personas que quieran interactuar con la red de Ethereum. Disponen de su **dirección** y un **balance** con el saldo disponible en su cuenta.

- **Contratos**: Un contrato en Ethereum también se identifica por su dirección y en este contiene el **código del contrato**, un **balance** con el saldo disponible y su propia **memoria donde almacenar información**. Cada vez que un contrato recibe un mensaje en forma de transacción, se ejecuta su código permitiendo la lectura y modificación de su memoria interna.

De esta manera se consigue que **usuarios y contratos se comporten de una manera indistinguible** en la red de Ethereum. Para que exista comunicación entre cuentas existen las transacciones.

Transacciones

Una transacción en Ethereum es una forma de comunicación entre dos cuentas. Esta comunicación es simplemente un paquete de datos que se envía de un sitio a otro mediante el pago de una **pequeña comisión**. El pago de estas transacciones se hace a través de la propia moneda de Ethereum llamada ***ether***, la cual se puede conseguir **minando en la red**.

Una transacción contiene:

- El recipiente del mensaje.
- Una firma identificando la cuenta que envía la transacción.
- La cantidad de ether que se envía.
- Un campo de datos opcionales (que se usarán para interactuar con los contratos).
- Un valor llamado *STARGAS* que representa el máximo de pasos computacionales que la transacción puede realizar.
- Un valor llamado *GASPRICE* que indica el precio que debe de pagar el emisor para poder realizar la transacción.

De esta manera se consigue que si un usuario quiere usar la red Ethereum para subir contratos a la cadena de bloques, tiene que conseguir ether, por lo tanto tiene que minar, lo que **contribuye al funcionamiento de toda la red.**

AlethZero

Para poder trabajar con Ethereum, es necesario **conectarse a la cadena de bloques** y poder interactuar con ella. Naturalmente, la forma de transmitir toda la información de los contratos y los usuarios es mediante una arquitectura P2P, ya que es una red que no precisa de ningún servidor. Para poder sincronizar la cadena de bloques en un ordenador es necesario un cliente que proporcionan los propios diseñadores de Ethereum llamado AlethZero (62) (ver figura 7).

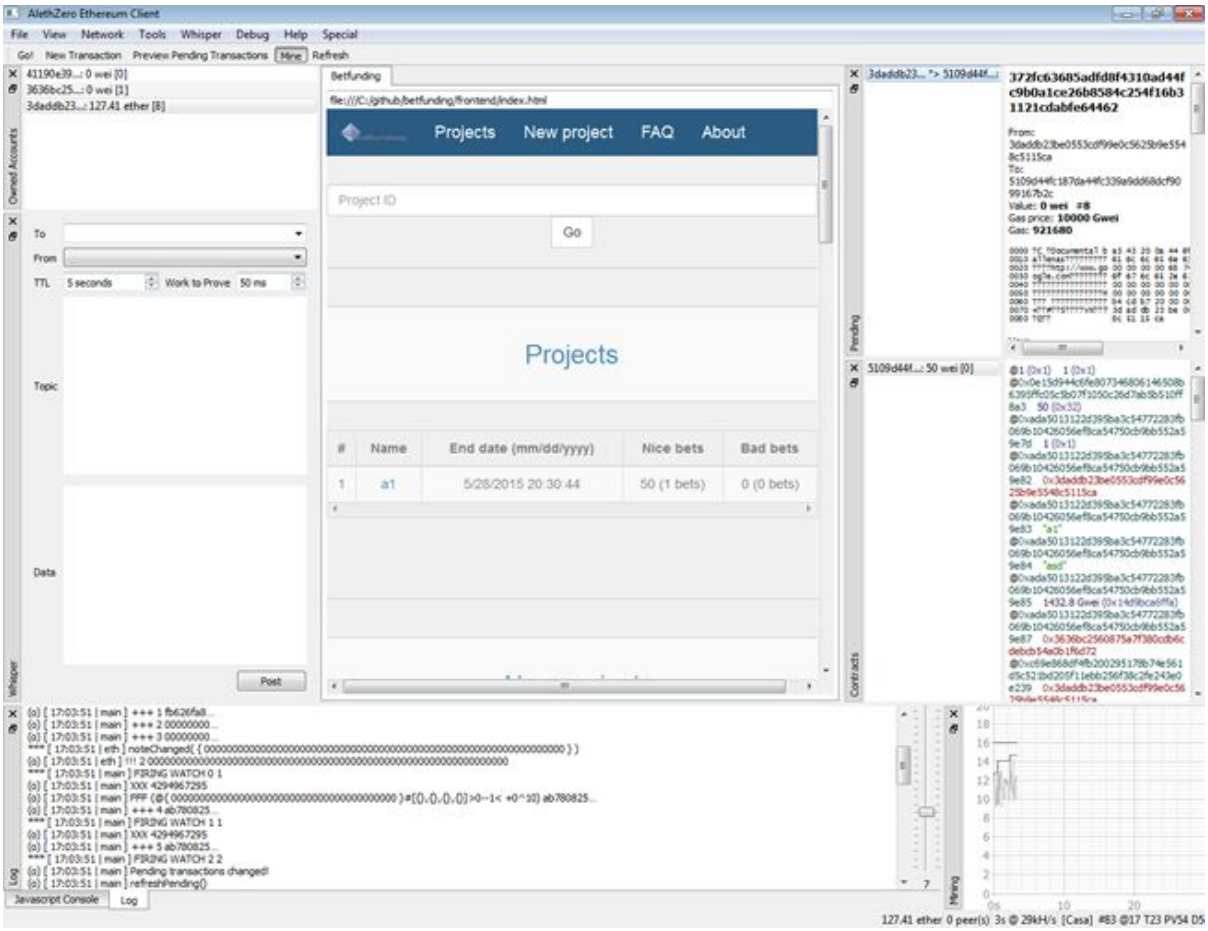


Figura 7. AlethZero v0.8.2

Se trata de un programa, un poco caótico en sus primeras versiones. En estas el programa se iniciaba con muchas ventanas pequeñas con información sobre contratos, transacciones, etc. Estas ventanas se acoplaban entre ellas sin posibilidad de hacer scroll, causando una pérdida de información para el usuario. Con **AlethZero se puede acceder a la red de Ethereum** ofreciendo la posibilidad de crear usuarios, minar, subir contratos a la cadena de bloques e interactuar con estos. También incorpora un navegador con el que podemos crear páginas con su propia librería JavaScript para comunicarse con los contratos de la cadena de bloques.

Serpent

Al igual que Bitcoin, Ethereum cuenta con con un **lenguaje de scripts** interno para la creación de transacciones EtherScript (63), de esta forma es posible crear transacciones con características especiales que puedan ser consideradas contratos inteligentes.

En el caso de Ethereum, este lenguaje de scripts es **Turing-completo** permitiendo adjuntar código más complejo a las transacciones a la hora de crear los **contratos inteligentes**. Sin embargo, estos lenguajes suelen ser de muy bajo nivel y bastante complejos de usar al contar con un gran número de códigos de operación (una porción de una instrucción de EtherScript), por ello los desarrolladores de Ethereum desarrollaron varios **lenguajes de alto nivel** para facilitar la tarea a los desarrolladores. Uno de dichos lenguajes fue **Serpent**.

Serpent fue diseñado para ser lo más parecido posible a **Python** para facilitar su aprendizaje a los desarrolladores ya familiarizados con este lenguaje. El código escrito en Serpent se compila a EtherScript para que pueda ser interpretado en la cadena de bloques. La compilación se realiza por el propio cliente de Ethereum. Una vez convertido a EtherScript, el código forma parte de la **transacción de creación del contrato** y puede ser consultado por el resto de nodos de la red.

```
init:
        contract.storage[1] = 0
        contract.storage[2] = msg.sender

code:
        if msg.data[0] == 1:
                if msg.sender == contract.storage[2]:
                        contract.storage[1] = 0
                        return(1)
                else:
                        return(0)
        else:
                contract.storage[1] += 1
                return(1)
```

Ejemplo de código escrito en Serpent

```
Code

PUSH1 0x5 PUSH1 0x11 JUMP JUMPDEST PUSH1 0xd2 DUP1 PUSH1 0x2a
PUSH1 0x0 CODECOPY PUSH1 0x0 RETURN JUMPDEST PUSH1 0x0 PUSH1 0x0
DUP2 SWAP1 PUSH1 0x0 ADD SSTORE POP CALLER PUSH1 0x1 DUP2 SWAP1
PUSH1 0x0 ADD SSTORE POP JUMPDEST JUMP STOP PUSH1 0x0
CALLDATALOAD PUSH29 0x1 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 SWAP1 DIV
DUP1 PUSH4 0x6 0x66 0x1a 0xbd EQ PUSH1 0x42 JUMPI DUP1 PUSH4 0x6d
0x4c 0xe6 0x3c EQ PUSH1 0x4e JUMPI DUP1 PUSH4 0xd8 0x26 0xf8 0x8f EQ
```

```
PUSH1 0x5e JUMPI STOP JUMPDEST PUSH1 0x48 PUSH1 0xbf JUMP JUMPDEST
PUSH1 0x0 PUSH1 0x0 RETURN JUMPDEST PUSH1 0x54 PUSH1 0xb1 JUMP
JUMPDEST DUP1 PUSH1 0x0 MSTORE PUSH1 0x20 PUSH1 0x0 RETURN
JUMPDEST PUSH1 0x64 PUSH1 0x6a JUMP JUMPDEST PUSH1 0x0 PUSH1 0x0
RETURN JUMPDEST PUSH1 0x1 SLOAD PUSH20 0xff 0xff 0xff 0xff 0xff 0xff 0xff
0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff AND CALLER PUSH20
0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff
0xff 0xff AND EQ PUSH1 0xa2 JUMPI PUSH1 0xae JUMP JUMPDEST PUSH1 0x0
PUSH1 0x0 DUP2 SWAP1 PUSH1 0x0 ADD SSTORE POP JUMPDEST JUMPDEST
JUMP JUMPDEST PUSH1 0x0 PUSH1 0x0 SLOAD SWAP1 POP PUSH1 0xbc JUMP
JUMPDEST SWAP1 JUMP JUMPDEST PUSH1 0x1 PUSH1 0x0 SWAP1 DUP2
SLOAD ADD SWAP1 DUP2 SWAP1 PUSH1 0x0 ADD SSTORE POP JUMPDEST
JUMP
```

Ejemplo de código de EtherScript

En el código de ejemplo mostrado puede verse la estructura y alguna de las características de este lenguaje. El apartado “init” **sólo se ejecuta una vez** durante la creación del contrato en la cadena de bloques, mientras que el apartado “code” **se ejecuta cada vez** que el contrato **recibe una transacción**.

La información sobre la transacción que ejecuta el contrato puede consultarse durante la ejecución del mismo; por ejemplo, la instrucción “msg.sender” permite al contrato saber desde qué dirección se ha solicitado su ejecución. Mediante “msg.data” se accede a un array de información enviado por el usuario; de esta manera puede interactuar con el contrato enviándole información que será leída durante la ejecución.

La información almacenada en el propio contrato se puede consultar mediante la instrucción “contract.storage”, que devuelve un array en donde cada posición accede a 32 bytes de información.

Durante las primeras versiones de Ethereum, Serpent **era considerado el lenguaje principal** de la plataforma aunque aún se encontraba en una fase muy temprana de su desarrollo, y no contaba con tipado de datos, estructuras o funciones. La aparición de Solidity lo ha ido desplazando progresivamente como lenguaje principal de la plataforma.

Solidity

Solidity es otro de los lenguajes de Ethereum que con el tiempo ha ido desplazando a los anteriores y **consolidándose como lenguaje oficial**. Al

contrario que el resto de lenguajes de la plataforma, Solidity cuenta con características avanzadas como estructuras de datos o funciones.

```
contract Counter {
   uint number;
   address owner;

   function Counter(){
         number = 0;
         owner = msg.sender;
   }

   function count() {
         number += 1;
   }


   function reset() {
         if(msg.sender == owner)
               number = 0;
   }

   function get() returns (uint r) {
         return number;
   }
}
```

Ejemplo de código escrito en Solidity

En apariencia, Solidity es **similar a Javascript** para minimizar la barrera de entrada a nuevos desarrolladores. Tiene los tipos de datos básicos más comunes además de otros específicos para la plataforma como “address”, que almacena direcciones de Ethereum. Al igual que el resto de lenguajes de Ethereum, cuenta con un conjunto de variables globales que permiten el acceso a información sobre la transacción.

La manipulación de los datos almacenados en el contrato ya no se realiza directamente por el programador, sino que es el compilador el encargado de asignar su posición en memoria.

Su principal ventaja respecto a los lenguajes anteriores es la posibilidad de **dividir el contrato en funciones** y emplear **estructuras de datos** o tablas hash. Un contrato programado con Solidity se puede ver como una **clase**, con sus **atributos**, **métodos** y **constructor**.

Bootstrap

Para facilitar el uso de los contratos alojados en la cadena de bloques, Ethereum cuenta con una librería llamada web3 (64) que permite conectar con ellos mediante el uso de tecnologías web. Gracias a ello, **es posible crear una interfaz gráfica** similar a la de cualquier página web con la que los usuarios ya se sientan familiarizados.

Para el desarrollo de la interfaz gráfica de nuestra aplicación, decidimos hacer uso de Bootstrap (65). Bootstrap es un **framework de software libre** para el diseño de aplicaciones web creado por Twitter. Este framework permite diseñar una interfaz gráfica de forma sencilla que se adapta a diferentes tipos de pantalla y dispositivos.

Otras tecnologías

Aparte de las tecnologías ya mencionadas, durante el desarrollo de nuestra aplicación hemos ido haciendo uso de diferentes aplicaciones y tecnologías para facilitar su desarrollo. En la siguiente tabla se muestra una lista completa:

Tipo	Nombre	Uso
Ethereum	AlethZero	Conexión con Ethereum y navegador
	Ether	Criptomoneda
Lenguajes de programación	Serpent	Programación de contratos
	Solidity	Programación de contratos
	Javascript	Programación en el lado del usuario y comunicación con la cadena de bloques
Estructura y diseño	HTML	Diseño del frontend
	CSS	Diseño del frontend
	MyBalsamiq	Creación de un mockups

Librerías	web3	Comunicación con un nodo local de Ethereum e interacción con la cadena de bloques.
	JQuery	Diseño del frontend
	Bootstrap	Diseño del frontend
IDEs y editores de texto	MIX	Escritura de código
	Notepad++	Escritura de código
	Word	Escritura de documentación
	Google Docs	Escritura de documentación
Repositorios	Github	Trabajo colaborativo y control de versiones

Capítulo 4. Experimentación con Ethereum

Durante el transcurso del trabajo hemos realizado **varias aplicaciones** antes de comenzar con el desarrollo de la plataforma de *crowdfunding*. Con estos desarrollos buscábamos **comprobar la madurez de los distintos lenguajes disponibles** y comenzar a familiarizarnos con la plataforma.

La primera aplicación realizada fue una sala de chat con el objetivo de comenzar a **familiarizarnos** con el concepto de **cadena de bloques** y cómo interactuar con ella desde la librería web3 aportada por Ethereum mediante una interfaz gráfica.

La segunda aplicación fue un juego de apuestas. Su desarrollo se realizó coincidiendo con el anuncio de Solidity como nuevo lenguaje de programación de Ethereum. Se realizaron dos implementaciones en Serpent y Solidity para analizar las **diferencias entre ambos** y su idoneidad para el desarrollo de la plataforma de *crowdfunding*.

Primer prototipo: ETHChat

Las salas de chat públicas empleadas habitualmente están alojadas en un servidor central que conectan a los usuarios entre sí y sincroniza los mensajes publicados entre ellos, un ejemplo de este funcionamiento es el protocolo IRC (66), Cryptocat (67), Terra Chat (68) o Lycos Chat (69). Esto presenta un problema, puesto que en cualquier momento la aplicación puede dejar de funcionar o **perder información** si los administradores deciden dejar de alojarla o falla el servidor. La aplicación de chat que creamos buscaba **evitar estos problemas** mediante la creación de un registro público de las conversaciones alojado en la cadena de bloques.

ETHChat es la primera aplicación que realizamos con Ethereum para comprobar su funcionamiento. El lenguaje de programación empleado fue la primera versión de Serpent, que carecía de funciones, estructuras y tipos de datos básicos. La memoria de un contrato en aquella versión de Serpent consistía en un array con un espacio de 32 bytes por cada posición.

La aplicación funciona como una **sala de chat pública** donde cualquier usuario de la red puede entrar y publicar sus mensajes. El chat carece de servidor central, los mensajes escritos se adjuntan a una **transacción propagada de forma P2P** y se almacenan en la cadena de bloques.

Los usuarios se identifican mediante la dirección desde la que mandan los mensajes a la sala de chat. Opcionalmente, pueden emplear un nombre de usuario registrando su dirección.

Implementación

La aplicación está estructurada en **dos contratos**, uno principal encargado de **almacenar un historial** de los mensajes junto con el autor de los mismos, y otro secundario encargado de **guardar la relación entre direcciones y nombres** de usuario (ver figura 8).

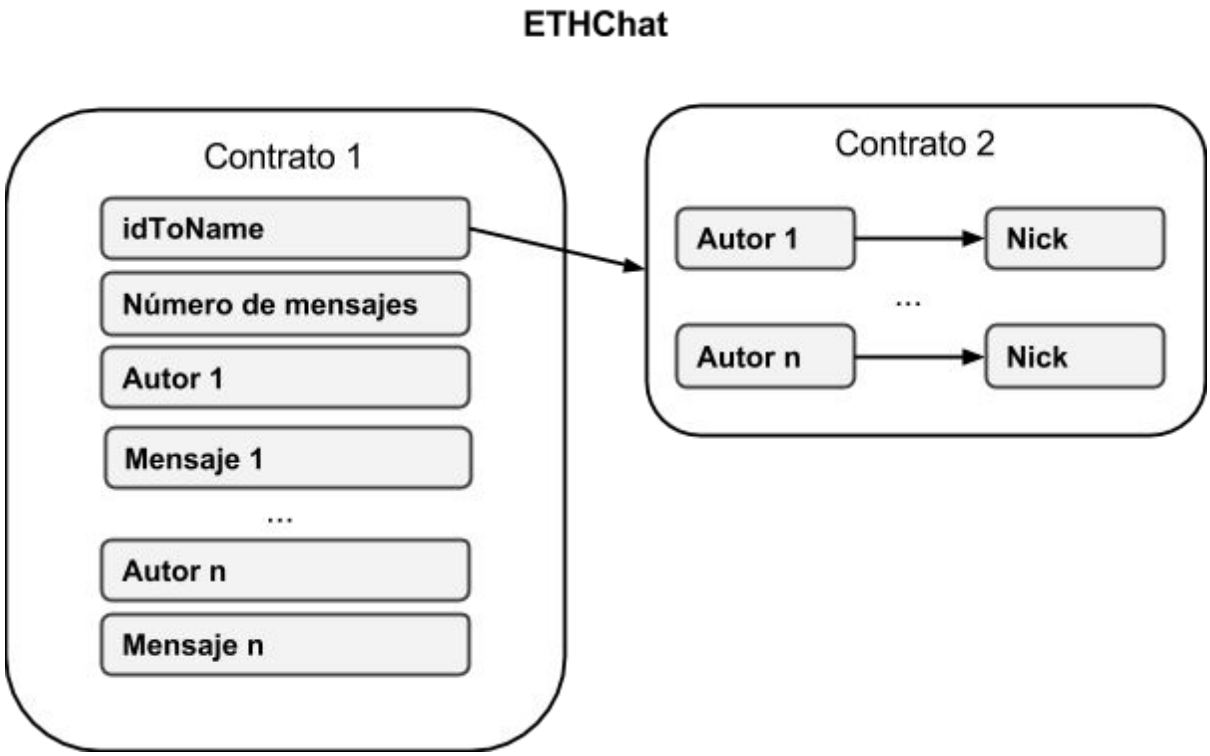


Figura 8: Estructura de los contratos en ETHChat.

El contrato principal se encarga de crear el contrato secundario y guardar una referencia en memoria para futuras consultas. Cada mensaje del chat ocupa un total de 6 posiciones del array, 1 para guardar la dirección del autor y 5 para el contenido del mensaje, limitando su longitud a 160 caracteres, configurable desde el código.

El contrato secundario únicamente se encarga de guardar el nombre de usuario registrado para cada dirección, empleando la dirección de envío de la transacción como clave en el array de memoria y el nombre enviado como datos adjuntos a la transacción como valor.

El chat dispone de una **interfaz visual** desde donde los usuarios pueden **escribir y leer los mensajes** escritos en tiempo real, limpiar la pantalla de mensajes, consultar el historial de los mensajes de la aplicación o cambiar de nombre de usuario. Los mensajes se introducen desde una casilla de texto inferior que indica al usuario el número de caracteres restantes antes de alcanzar el máximo permitido.

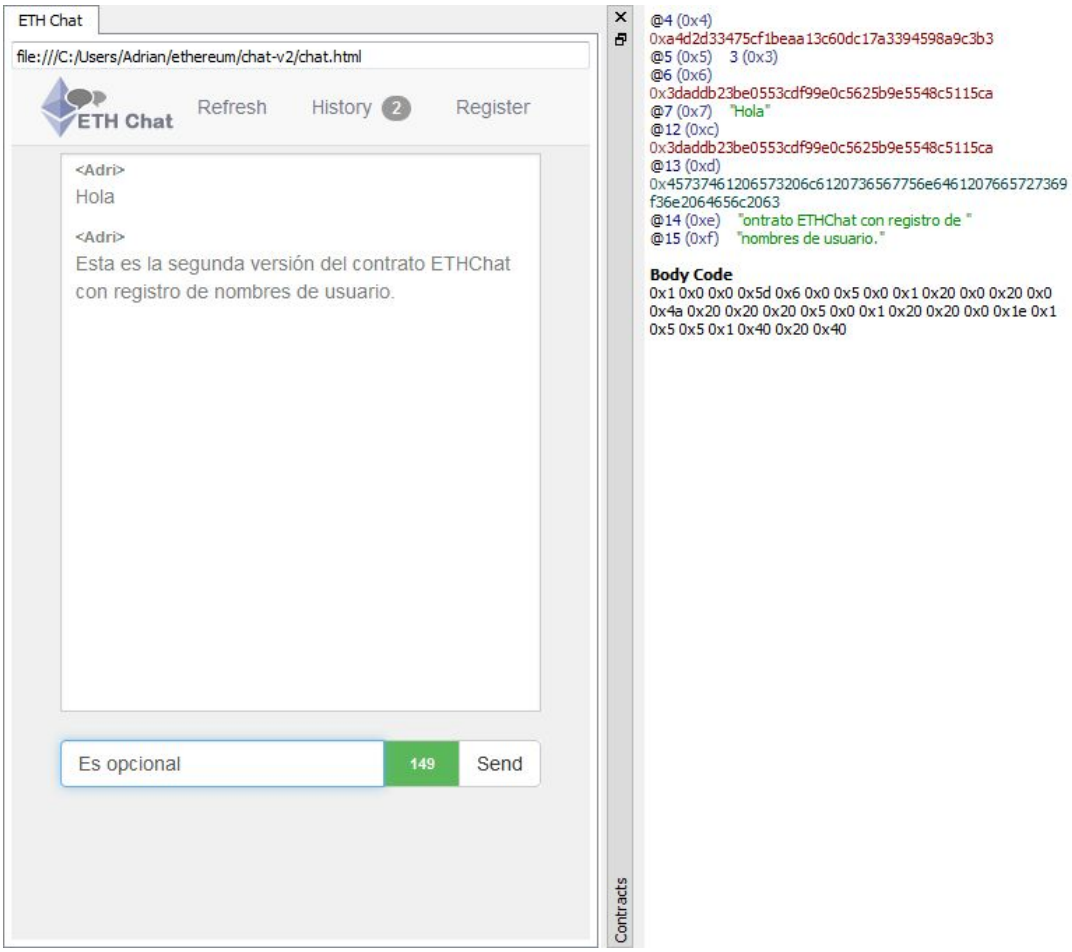


Figura 9: ETHChat (izquierda) y su visualización en la cadena de bloques (derecha)

En la mitad derecha de la figura 9 se puede ver la información almacenada en la memoria del contrato principal en la cadena de bloques. En las primeras posiciones se guarda información interna del contrato, como el número de mensajes o la referencia al contrato secundario. A partir de la sexta posición se guarda la información referente a los mensajes en bloques.

La primera posición del bloque del mensaje se reserva para la dirección del autor del mensaje (en rojo en la imagen) y las siguientes para el contenido del mensaje (en verde). Antes de ser enviado a la cadena de bloques, el mensaje se divide en bloques de 32 bytes y las cadenas que cuenten con caracteres especiales se codifican para su almacenamiento.

Conclusiones

La realización de este prototipo nos acercó mucho al concepto de la cadena de bloques y gracias a ello, fuimos **aprendiendo el funcionamiento interno de Ethereum**.

En las primeras versiones de nuestro chat tuvimos **problemas al interactuar con los usuarios** que utilizaban el servicio, así que tuvimos que averiguar cómo se comunicaba la cadena de bloques con los clientes individualmente, lo cual **nos enseñó mucho sobre la parte JavaScript del frontend** que ejecuta el usuario.

Una vez concluido el último prototipo de ETHChat, ya teníamos conocimientos sobre las **transacciones**, los **contratos** y el **entorno de programación** proporcionado por los desarrolladores de Ethereum, lo cual nos dió pie a **poder experimentar con algún prototipo más** antes de empezar con la plataforma final de *crowdfunding*.

Segundo prototipo: Juego de Azar

Conforme se ha ido facilitando la transmisión de dinero a través de Internet, los casinos online y juegos de apuestas se han ido popularizando. El uso de **criptomonedas** permite **reducir las comisiones** en las transacciones y aumentar el anonimato de los jugadores. Páginas como Satoshi Dice (70), Satoshibet (71) y FortuneJack (72) han popularizado las **apuestas mediante Bitcoin**, convirtiéndose en uno de los **servicios más usados** con esta tecnología. Estas páginas gestionan las carteras de Bitcoin de sus clientes desde un servidor central, por lo que los usuarios corren el riesgo de perder el dinero si la página cierra.

Nuestro objetivo era desarrollar una aplicación que permitiera **gestionar las apuestas desde la propia cadena de bloques**, sin necesidad de un servidor central.

La aplicación elegida fue un sistema de apuestas similar a una lotería. El sistema permite a cualquier usuario iniciar una lotería indicando el tiempo disponible para realizar las apuestas. Durante ese tiempo, los usuarios podrán apostar por el número que deseen.

Cuando finaliza el tiempo, el contrato espera la orden para realizar el sorteo y escoger un número ganador de forma aleatoria mediante el timestamp del bloque. Los usuarios que apostaron por el número ganador recibirán su parte correspondiente del bote acumulado en función de la cantidad apostada. En caso de no haber ganador se le devolverá a cada usuario su apuesta.

Transición de Serpent a Solidity

El desarrollo de esta aplicación coincidió con el anuncio de Solidity como nuevo lenguaje de programación para Ethereum. La aplicación **ha sido desarrollada simultáneamente en Serpent y Solidity** para comprobar el grado de **estabilidad** y **usabilidad** de ambos lenguajes de cara a crear la aplicación final de crowdfunding. A continuación se hace un repaso sobre las principales diferencias encontradas al realizar ambas implementaciones.

Estructura

En el caso de Serpent, la implementación se realizó mediante **dos contratos**. Uno encargado de almacenar **una lista de loterías** y otro **específico** para cada lotería encargado de guardar el dinero y el registro de apuestas de los usuarios (ver figuras 10, 11 y 12). El contrato principal es el encargado de crear las loterías y recibir todas las peticiones de los usuarios, que serán redireccionadas al contrato correspondiente.

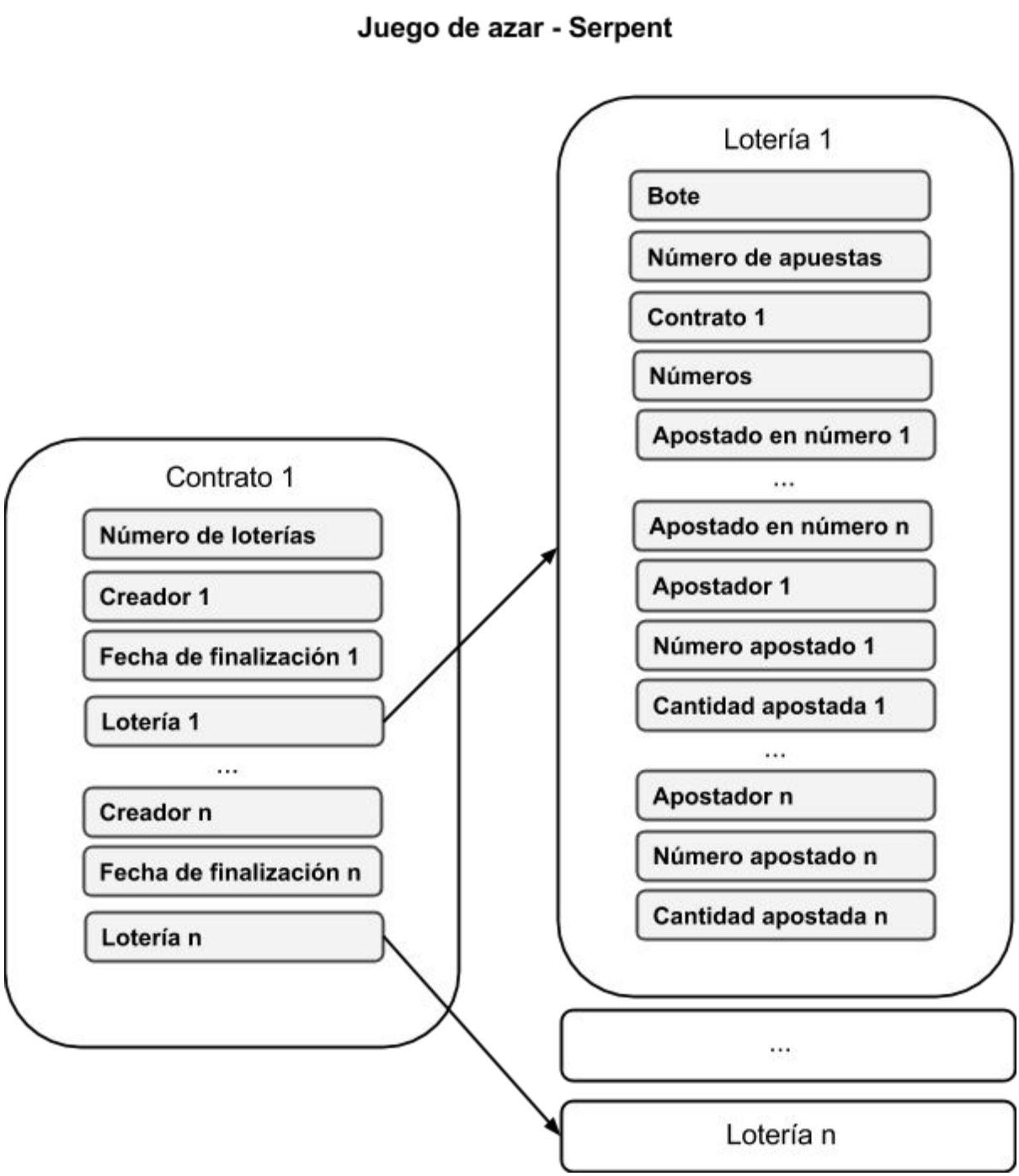


Figura 10: Estructura de los contratos de Juego de azar en Serpent.

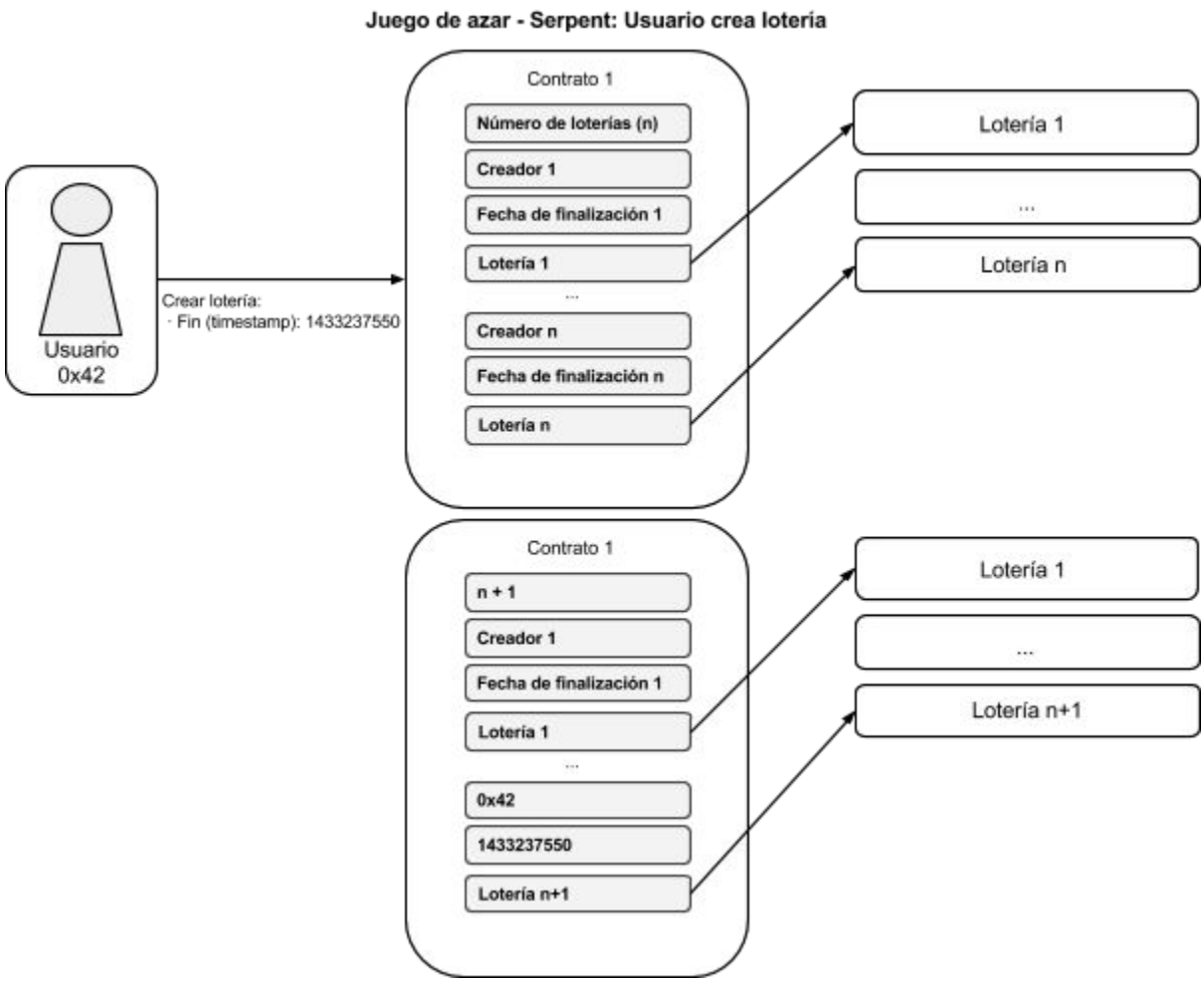


Figura 11: Estructura de los contratos de Juego de azar en Serpent.

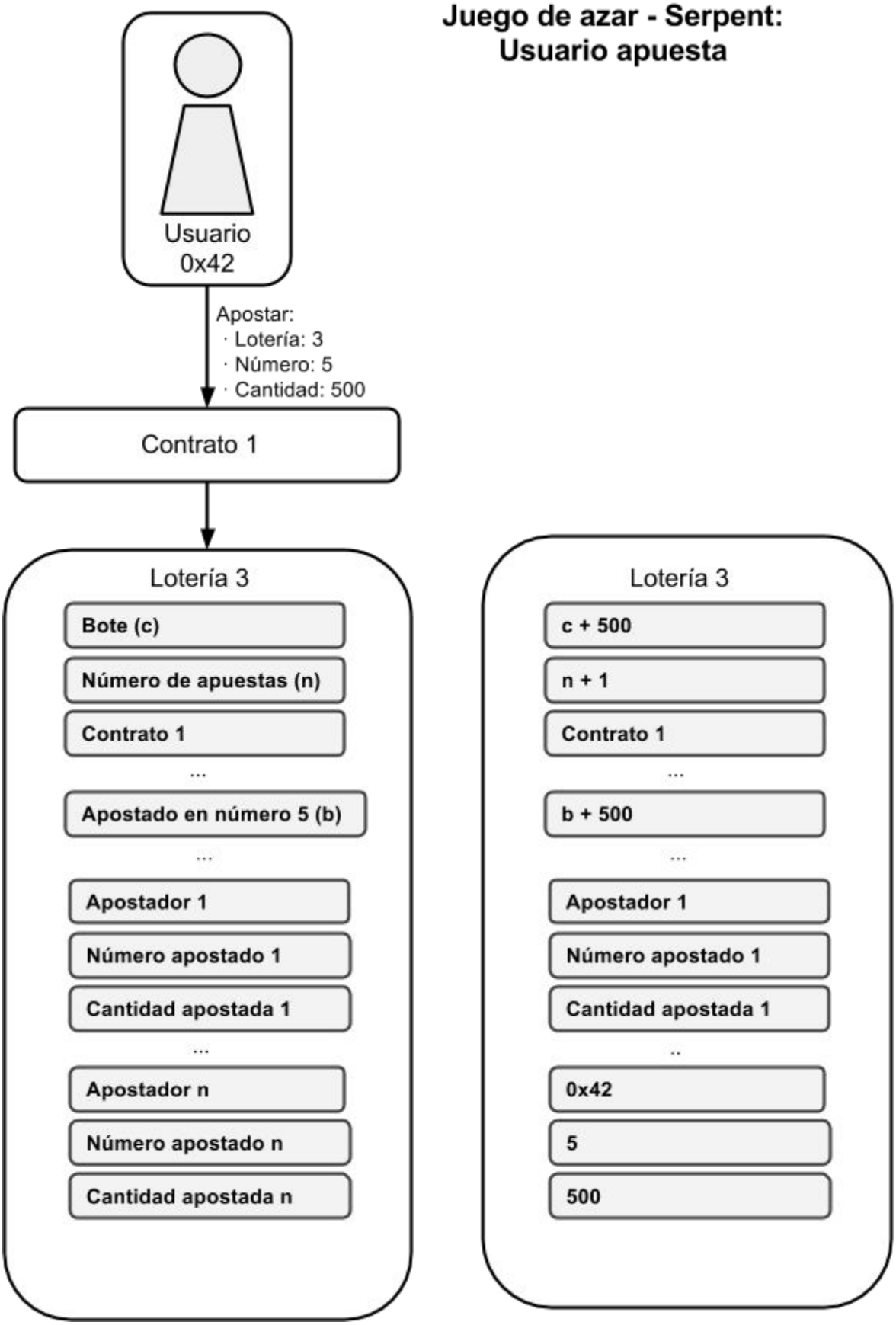


Figura 12: Estructura de los contratos de Juego de azar en Serpent.

Al no disponer de más memoria aparte de un array de datos de 32 bytes, la información sobre las apuestas se **guardan en bloques de tres posiciones** del array (apostador, número apostado y cantidad apostada) y se reservan las primeras posiciones del array para información sobre el contrato.

En Solidity, hay solo un contrato que gestiona todas las loterías mediante estructuras almacenadas en una lista. Las apuestas tienen asignado un **id** y cuentan con sus propias listas donde se almacena la **información sobre la apuesta** (ver figura 13).

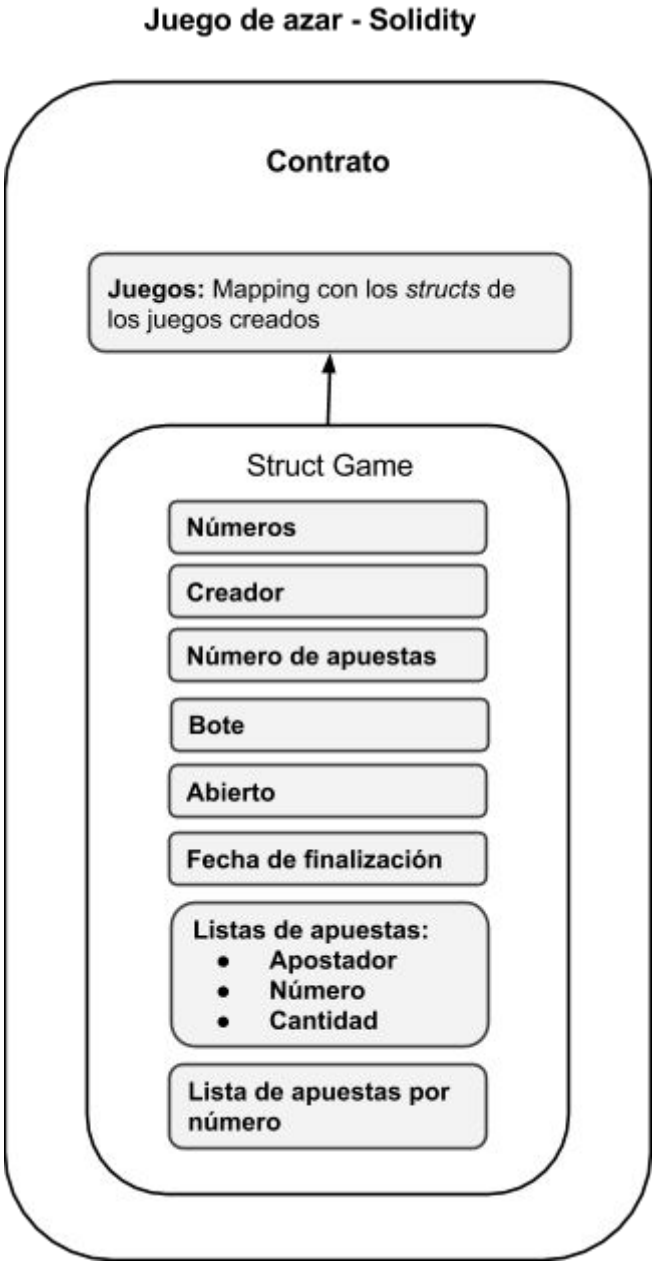


Figura 13: Estructura de la aplicación Juego de azar en Solidity.

La gestión de la memoria ya no recae en el programador y pasa a ser gestionado por el compilador.

Funciones

Al carecer de funciones, en Serpent se toma el primer elemento del array de datos adjunto a la transacción para indicar la funcionalidad del código que se quiere ejecutar.

```
var contrato = '0xa82e47dda3c6cf9e4d5e685a2a5d6b722c992d00';
var datos = [1, dato2, dato3]

web3.eth.transact({to: contrato, data: datos, gas: 5000});
```

Ejemplo de envío de transacción desde JavaScript.

```
if msg.data[0] == 0:
        # Código de la “función” 0

elif msg.data[0] == 1:
        # Código de la “función” 1

elif msg.data[0] == 2:
        # Código de la “función” 2
```

Ejemplo de código de Serpent simulando el uso de funciones.

Tomando como ejemplo los códigos mostrados arriba de JavaScript y Serpent, el código que se ejecutará en el contrato es el código de la “función” 1.

En Solidity, las funciones permiten a los usuarios **llamar a la parte de código que necesiten** de forma más intuitiva y mantener una mayor **limpieza en el código** por parte del desarrollador.

```
var abi = [
        { name: "funcion1", type: "function", inputs: [{ name: "x", type: "uint" }],
        outputs: [] },
        { name: "funcion2", type: "function", inputs: [], outputs: [{ name: "r",
        type: "uint" }] }
];
var direccion = '0xa82e47dda3c6cf9e4d5e685a2a5d6b722c992d00';
var Contrato = web3.eth.contractFromAbi(abi);
var contratoInstancia = new Contrato(direccion);

contratoInstancia.funcion1(123);
```

Ejemplo de envío de transacción desde JavaScript.

```
contract SimpleStorage {
        function funcion1(uint x) {
                // Código de la función 1
        }

        function funcion2() returns (uint r) {
                // Código de la función 2
        }
}
```

Ejemplo de código de Solidity usando funciones.

En los ejemplos de arriba se muestra cómo se envía información y su posterior tratamiento mediante funciones en un contrato programado con Solidity. Para ello, desde JavaScript se indica el ABI, su dirección, y se crea una instancia del contrato desde la que ejecutar las funciones. En el código del ejemplo se ejecutará la función “funcion1” con el parámetro “123”.

Conclusiones

Este pequeño programa fue muy significativo para el desarrollo de este proyecto por dos factores importantes:

En primer lugar **aprendimos a manejar dinero** en forma de criptomoneda **dentro de la plataforma de desarrollo** Ethereum. Esto es una parte primordial del programa final ya que se trata de una aplicación de financiación de proyectos y el manejo de dinero **es la parte más importante** de la implementación de esta.

En segundo lugar nos adaptamos al nuevo lenguaje de programación de contratos Solidity, el cual era **mucho más intuitivo** que su antecesor Serpent. Gracias a este nuevo lenguaje pudimos desarrollar la plataforma de *crowdfunding* de una manera mucho más **limpia**, **intuitiva** y **fácil de entender**.

Con la realización de este prototipo de juegos de azar **estábamos preparados para pasar a la implementación final** de nuestra aplicación Betfunding, ya que teníamos conocimientos de:

- El **funcionamiento interno** de las transacciones en la cadena de bloques.
- El **manejo de dinero dentro de un contrato** y cómo transferirlo a uno o varios usuarios.
- El uso de **programación orientada a objetos** para desarrollar los contratos gracias al lenguaje de programación Solidity.

Capítulo 5. Betfunding: plataforma de crowdfunding distribuida

Betfunding es la plataforma de crowdfunding distribuida de la que trata este proyecto. El dinero empleado es la criptomoneda Ether y su gestión se realiza de forma **distribuida** usando un contrato en la cadena de bloques para evitar la necesidad de intermediarios.

Cualquier usuario puede **publicar un proyecto** y la financiación funciona mediante **apuestas**. Se crean dos depósitos, el depósito de los usuarios y el de los creadores. Las personas que quieran que el proyecto **salga adelante** envían sus aportaciones al depósito de los usuarios (**apuesta negativa**), mientras que las personas que quieran **realizar** ellos mismos un proyecto lo harán en el de los creadores (**apuesta positiva**). Mediante la aportación de dinero en el depósito de los creadores, el **creador se compromete** a la realización del proyecto y ofrece ese dinero como **garantía**, de modo que si no cumple su palabra pierde el dinero. Sin embargo a nivel la aplicación, los creadores y los especuladores son indistinguibles, ya que ambos apuestan positivamente, pero **el especulador no tienen porqué contribuir a la realización del proyecto.**

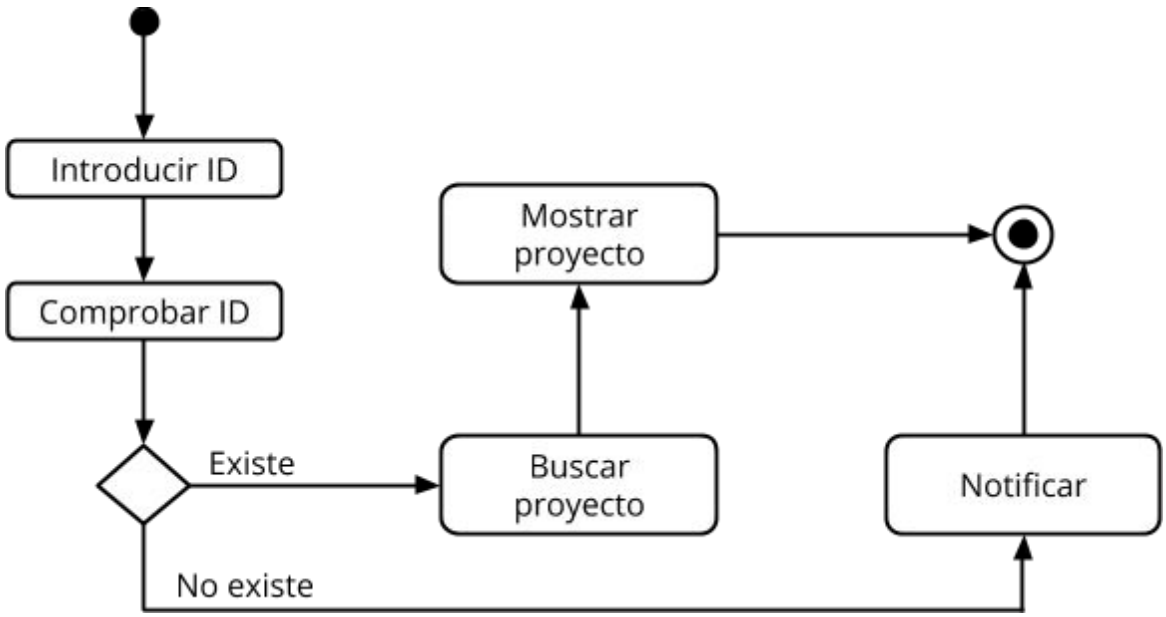
La verificación de que el proyecto ha sido correctamente realizado siguiendo las indicaciones de la persona que lo publicó la realiza un **juez**. El juez es una **dirección de Ethereum** nombrada por la persona que publica el proyecto. Un juez puede ser una **persona de confianza** que realice la **verificación** de forma manual u otro contrato.

Si el proyecto se **verifica dentro del plazo** establecido para su realización, el dinero del depósito de los usuarios se **reparte entre los creadores**. El reparto entre los creadores se realiza **ponderando por el orden de llegada y la fianza aportada**. Si llega fecha de realización y no se ha verificado un proyecto, entonces la fianza aportada por los creadores se reparte entre quienes apoyaron económicamente la realización del proyecto de forma ponderada por la cantidad aportada.

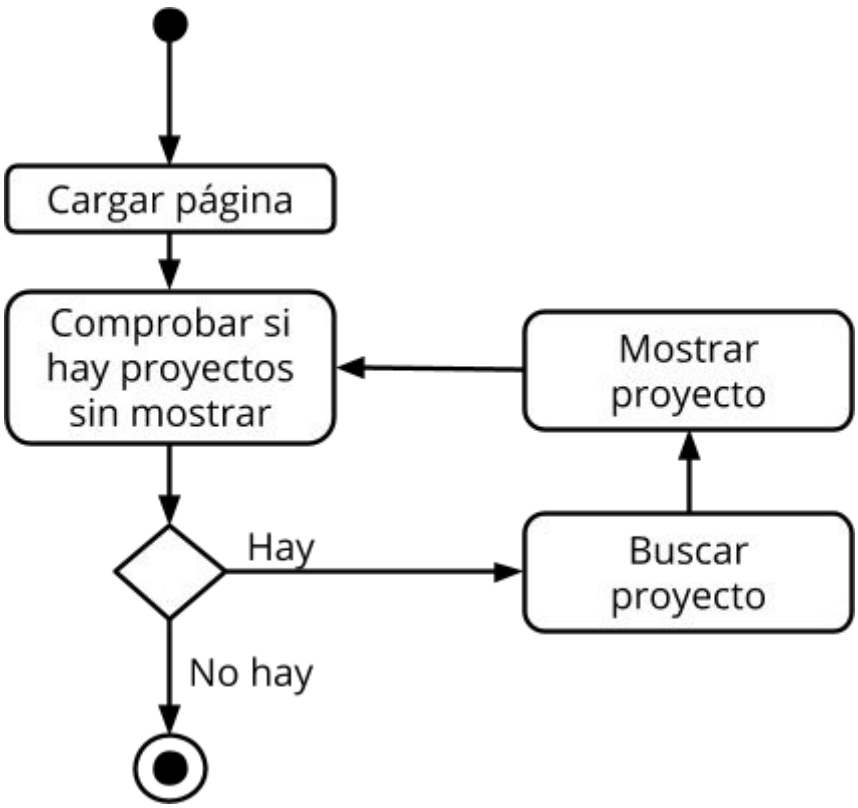
Especificación

Requisitos funcionales

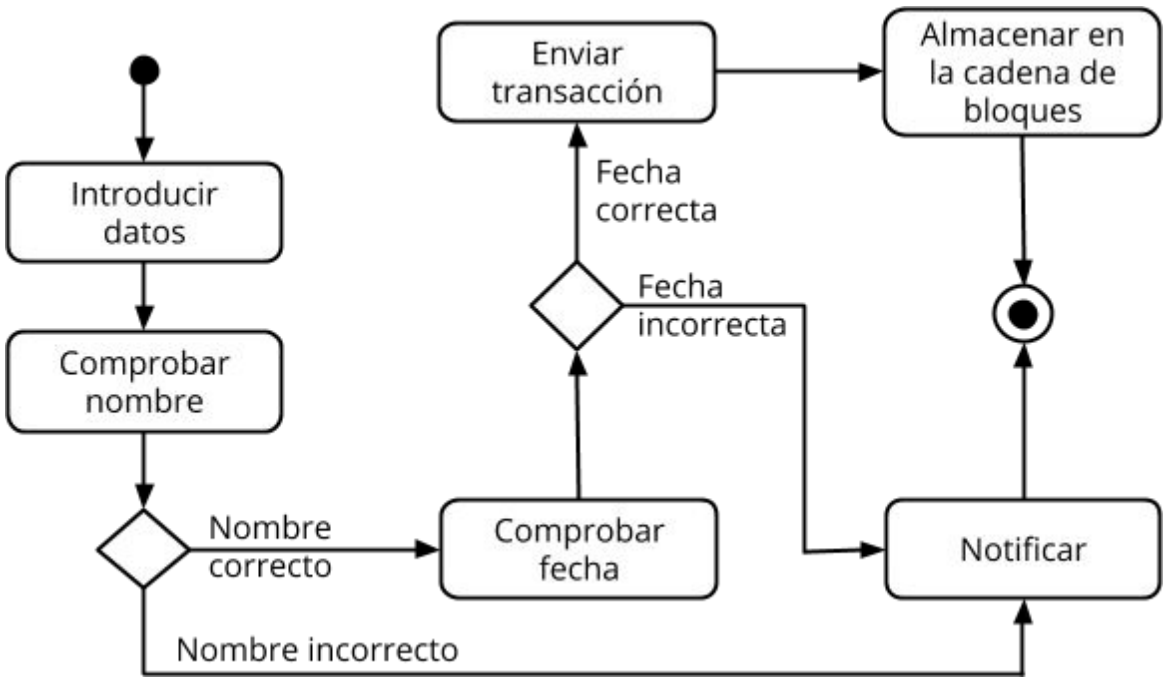
RF-1	**Buscar proyecto**
Descripción	Conocido el ID del proyecto que se quiere consultar, el usuario puede acceder a la información y el estado del proyecto.
Entrada	ID del proyecto
Salida	ID, nombre, creador, fecha de finalización, apuestas, juez, descripción, estado.
Necesita	Conexión con nodo local de Ethereum.
Precondición	Existe un proyecto con dicho ID.
Poscondición	Información del proyecto mostrada por pantalla.
Excepciones	Si no existe un proyecto con el ID seleccionado, se informa al usuario.



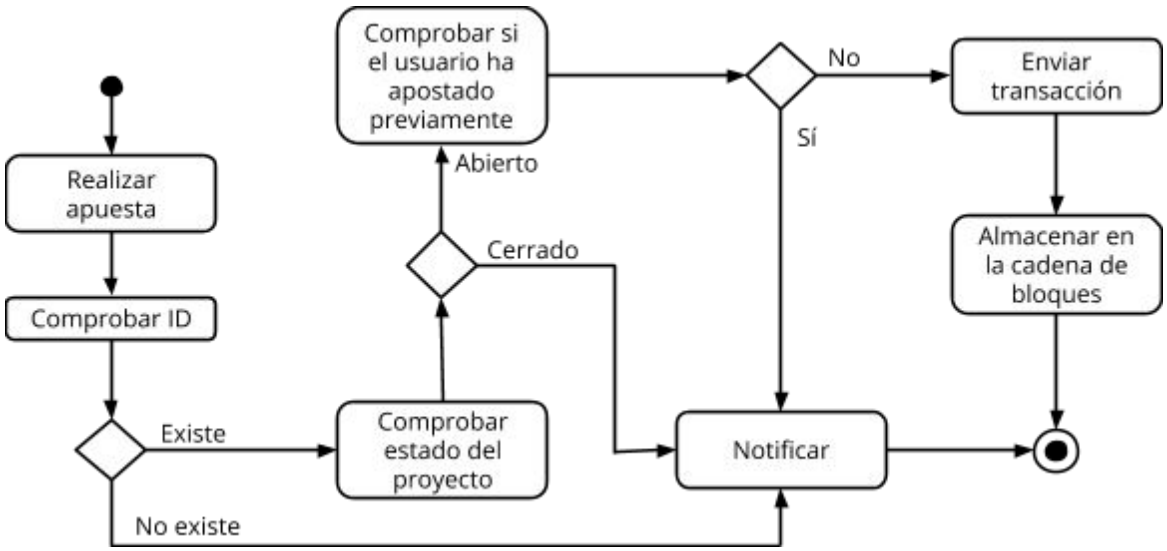
RF-2	**Ver proyectos**
Descripción	Muestra la lista de los proyectos creados en la plataforma y permite consultar su información.
Entrada	-
Salida	ID, nombre, fecha de finalización y apuestas de todos los proyectos.
Necesita	Conexión con nodo local de Ethereum.
Precondición	-
Poscondición	Muestra información breve sobre todos los proyectos de la plataforma
Excepciones	-



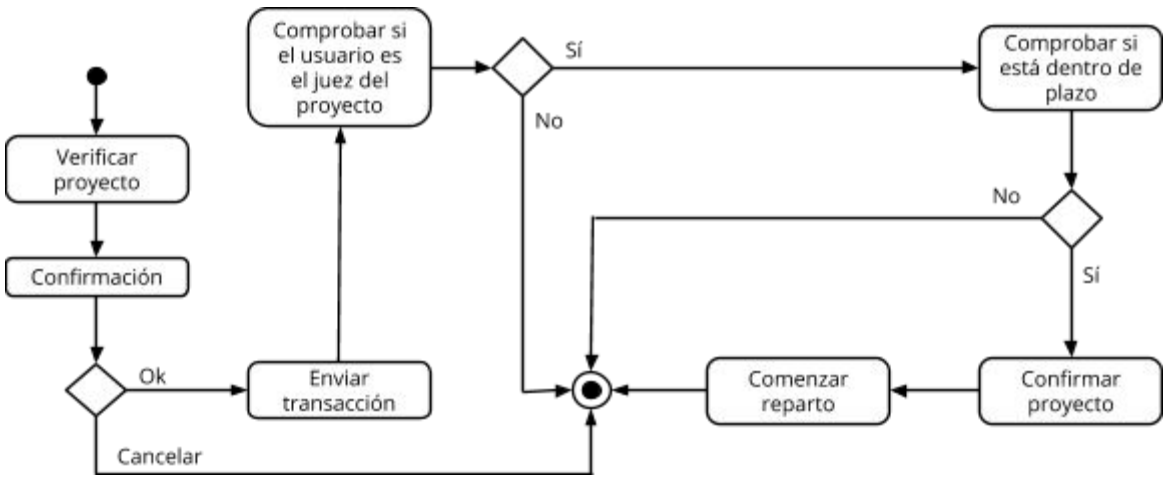
RF-3	**Proponer proyecto**
Descripción	Crea un nuevo proyecto y lo sube a la cadena de bloques.
Entrada	Nombre, fecha de finalización, juez, descripción.
Salida	-
Necesita	Conexión con nodo local de Ethereum. Conexión con otros nodos de la red. Ether.
Precondición	La fecha de finalización es posterior a la actual y el nombre del proyecto es menor de 32 caracteres.
Poscondición	Proyecto añadido a la cadena de bloques.
Excepciones	Si la fecha de finalización es incorrecta o el nombre del proyecto excede el límite, se notifica al usuario.



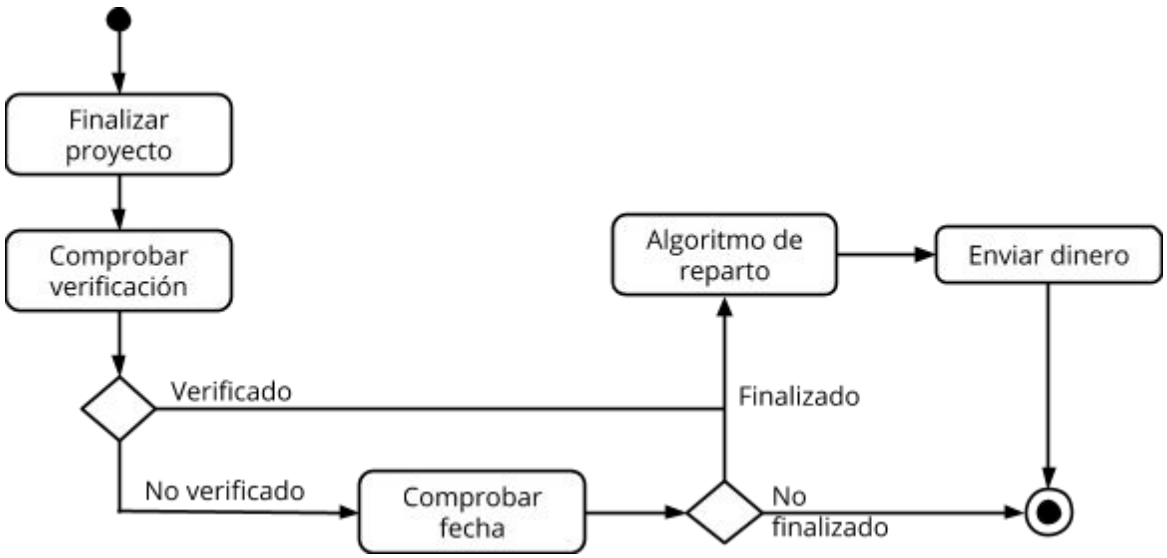
RF-4	**Apostar**
Descripción	El usuario apuesta la cantidad de ether indicada a un proyecto de su elección indicando el sentido de la apuesta.
Entrada	ID proyecto, sentido de la apuesta, cantidad.
Salida	-
Necesita	Conexión con nodo local de Ethereum. Conexión con otros nodos de la red. Ether.
Precondición	Existe un proyecto con ese ID y el proyecto está abierto (no se ha sobrepasado la fecha de finalización ni ha sido verificado aún)
Poscondición	La apuesta queda registrada en la cadena de bloques y el dinero apostado pasa al contrato.
Excepciones	Si no existe un proyecto con ese ID o el proyecto está cerrado, se notifica al usuario.



RF-5	**Verificar proyecto**
Descripción	El usuario verifica que un proyecto ha cumplido los requisitos establecidos dentro del plazo.
Entrada	-
Salida	-
Necesita	Conexión con nodo local de Ethereum. Conexión con otros nodos de la red. Ether.
Precondición	El usuario que ejecuta la acción es el juez del proyecto y no se ha sobrepasado la fecha de finalización.
Poscondición	Se verifica el proyecto y comienza el reparto del dinero.
Excepciones	Si el usuario que ejecuta la acción no es el juez del proyecto o se ha pasado el plazo de realización del proyecto, no se verifica el proyecto.



RF-6	**Finalizar proyecto**
Descripción	El usuario avisa al proyecto de que es el momento de iniciar el reparto del dinero.
Entrada	-
Salida	-
Necesita	Conexión con nodo local de Ethereum. Conexión con otros nodos de la red. Ether.
Precondición	El proyecto debe estar verificado o haber sobrepasado la fecha de finalización.
Poscondición	Se reparto del dinero.
Excepciones	Si no se ha sobrepasado la fecha de finalización y el proyecto no ha sido verificado, no se reparte el dinero.



Requisitos no funcionales

Rendimiento: El rendimiento de la aplicación está **limitado por el funcionamiento de la plataforma** sobre la que está desarrollado. El tiempo medio de bloque en Ethereum es de aproximadamente 12 segundos (73). Dichas limitaciones quedan fuera del alcance de nuestra aplicación, que siempre trabaja muy por debajo de dichos límites.

Coste: Por el tipo de tecnologías empleadas, el código del contrato mediante el que se sincroniza la información requiere de gas para su ejecución, de modo que es muy importante **reducir estos costes al mínimo**. En el caso de nuestra aplicación, se han intentado reducir estos costes **trasladando** parte del tratamiento de los **datos de entrada a la interfaz**.

Extensibilidad: La funcionalidad de la aplicación puede ser ampliada conforme avance el grado de desarrollo de la plataforma. En **futuras versiones** se pueden incluir **nuevos modos de verificación** mediante la extensión del contrato seleccionado como juez cuando se haya desarrollado la funcionalidad que permite consultar información **fuera de la cadena de bloques** por parte de los desarrolladores de Ethereum.

Almacenamiento: El almacenamiento en la cadena de bloques **es muy costoso,** por ello, **la cantidad de datos** sincronizados mediante la cadena de bloques ha quedado **reducido al mínimo** evitando la incorporación de contenido multimedia. Dichos contenido se espera que se incorporen en futuras versiones cuando termine el desarrollo del protocolo de sincronización de ficheros estilo Bittorrent para Ethereum (Swarm).

Usabilidad: La aplicación ha sido desarrollada pensando en la **sencillez** de facilidad de uso. El usuario puede ver por pantalla todas las opciones de las que dispone de forma visual. Para aquellas funcionalidades que puedan crear algún tipo de confusión, se ha creado una **sección de ayuda** que da respuesta a las preguntas más frequentes.

Tipos de usuario

Al ser una plataforma dedicada a la financiación de proyectos mediante crowdfunding, es fácil identificar a dos tipos de usuarios diferentes: **creadores** y **financiadores**. Estos dos tipos de usuarios son comunes a los de cualquier otro sistema de crowdfunding, sin embargo, debido a la particularidad de la plataforma y el método de financiación mediante apuestas, en nuestra aplicación surgen otros dos tipos de usuarios: **jueces** y **especuladores**.

- **Financiadores**: Son usuarios que **quieren ver realizado un proyecto** pero no cuentan con las habilidades necesarias para llevarlo a cabo por su cuenta. Pueden **sugerir proyectos y financiarlos** mediante apuestas negativas.

  Su papel dentro de la plataforma es el de **aportar dinero** a un bote común del proyecto que quiera ver realizado con el objetivo de incentivar su desarrollo por parte de los creadores.

- **Creadores**: Son los encargados de llevar a cabo la **creación de los proyectos** propuestos por el resto de usuarios. En el momento de comprometerse a la realización de un proyecto deben realizar una **apuesta positiva a modo de garantía** para optar al bote de recompensa por su creación.

  Es posible que en un mismo proyecto haya **varios creadores trabajando al mismo tiempo**, ya sea de forma independiente o colaborando entre sí. Para ello, cada uno de los creadores deberá realizar una apuesta positiva por su cuenta al momento de comenzar a trabajar en el proyecto y el sistema **repartirá el bote** por su creación en función de la **cantidad aportada** como garantía en la apuesta y el **orden** en que se sumaron a la realización del proyecto.

- **Jueces**: En un principio pensamos intentar conectar el contrato con algún **RSS o fuente externa** a la cadena de bloques para poder verificar los proyectos. Sin embargo, los desarrolladores de Ethereum, a pesar de que dijeron que implementarían una forma para comunicar la cadena de bloques con fuentes externas, **decidieron no implementar** esta funcionalidad, al menos de momento. Por lo que tuvimos que pensar una **forma alternativa** para poder verificar un proyecto de manera imparcial para que ninguna de las dos partes saliera beneficiada.

A la conclusión a la que llegamos es que al **crear un proyecto se asigna un juez** del que se tienen que fiar los posibles inversores del proyecto. De ahí la importancia de especificar bien el modo de verificación de cada uno de los proyectos en la descripción de estos. Un juez puede ser tanto un **humano** que cuente con la confianza de ambas partes y realice la verificación de forma manual como **un contrato inteligente** que funcione de forma autónoma y verifique el proyecto basándose en unas reglas preestablecidas.

- **Especuladores**: En Betfunding la única regulación existente es la del propio código y permite que cualquier usuario puede ser creador y financiador de un proyecto sin aportar más información que la dirección desde donde manda las transacciones. Esto, junto al sistema de apuestas empleado para la financiación de los proyectos, abre la puerta a la aparición de especuladores, **usuarios que no tienen interés en los proyectos** y cuya única finalidad es la de **ganar dinero.**

  Un especulador **aporta dinero a uno u otro bote** de apuestas si ve que la cantidad de dinero aportada por el creador como fianza está **descompensada** respecto al bote de los usuarios. De esta forma, está pensando en el **beneficio económico** que puede sacar con ello y no en financiar un proyecto o formar parte del equipo de creadores. El papel de los especuladores en la plataforma ayuda a m**antener un equilibrio en las apuestas** y **aumentar los incentivos** económicos para creadores y financiadores consiguiendo con ello aumentar su participación.

Crowdfunding mediante apuestas

Normalmente, en una aplicación de crowdfunding los usuarios aportan dinero a una cuenta común, que se encarga de desbloquear los fondos una vez alcanzado el objetivo.

En nuestra aplicación queríamos que fueran los usuarios quienes propusieran los proyectos y **pudiera haber más de un creador** al mismo tiempo realizando el proyecto. Inspirados por sistemas descentralizados de predicción de mercados como Augur (74), decidimos emplear un sistema de **apuestas** para la **financiación de los proyectos**. De esta forma, ningún creador puede "secuestrar" la realización de un proyecto y todos se ven recompensados proporcionalmente.

En el caso de Betfunding se crean **dos cuentan independientes**, en una de ellas los usuarios **aportan dinero** como recompensa para los creadores. En la otra, los creadores **aportan una fianza** cuando comienzan a desarrollar el proyecto.

- Si el proyecto no se realiza correctamente dentro del plazo establecido, los **creadores pierden la fianza** y los **usuarios recuperan su dinero**. De esta forma los usuarios compensan el tiempo perdido repartiéndose la fianza depositada por los creadores y pueden volver a proponer el mismo proyecto con más recursos para atraer a nuevos creadores.

- Si el proyecto se realiza con éxito, los usuarios quedan satisfechos por ver realizado el proyecto que querían apoyar y los **creadores reciben el dinero** aportado por los usuarios como recompensa y recuperan la fianza.

Este sistema abre la puerta a que entren **especuladores** haciéndose pasar por creadores con la única intención de **hacer dinero** y quedarse con una buena parte de la recompensa depositada para los creadores. Para evitar este escenario, se hace uso de diferentes **algoritmos de reparto** para cada uno de los botes que custodian los fondos.

- La fianza depositada por los creadores se reparte entre los usuarios de forma **ponderada en función de la cantidad aportada**. De esta forma, los usuarios que estuvieran dispuestos a perder una mayor cantidad de dinero para la realización de un proyecto, recibirían una mayor compensación si éste no sale adelante.

- La recompensa depositada por los usuarios se reparte entre los creadores de forma ponderada en función de la fianza depositada y el **orden de llegada** a la realización del proyecto. De esta forma todos los creadores que comenzaron a trabajar **antes** en la realización del proyecto **recibirían una mayor parte** de la recompensa aportada por los usuarios a menos que los nuevos creadores aporten una fianza significativamente mayor.

Con estas modificaciones, se logra un **equilibrio entre ambos botes** de forma que se garantice una recompensa lo suficientemente interesante para quienes comenzaron a desarrollar el proyecto como para que continúen con él. Los nuevos creadores o especuladores haciéndose pasar por creadores que se unan posteriormente deberán sopesar si les merece la pena una menor recompensa por el mismo trabajo. En este escenario, **la mejor opción** para quienes entren después en el lado de los creadores **es colaborar** con quienes ya había antes en la realización del proyecto, ya sea mediante **trabajo o la provisión de recursos.**

Puede darse el caso de que quienes no están participando en el desarrollo del proyecto aporten una fianza lo suficientemente grande como para que a quienes sí lo están haciendo no les merezca la pena continuar por la recompensa. En este caso, el equilibrio en los botes volvería a su lugar con la entrada de nuevos especuladores en el lado de los usuarios atraídos por el tamaño de la fianza. Si esto no se produjera, los creadores de verdad podrán simplemente dejar de trabajar en el proyecto y apostar al bote contrario con la certeza de que el proyecto no saldrá adelante y recibir beneficios. Al mismo tiempo, pueden abrir un nuevo proyecto con las mismas características y continuar ahí su desarrollo.

Ejemplo

Alice es una apasionada de los juegos para móvil, pero no sabe desarrollar este tipo de aplicaciones. Lo que Alice quiere es una aplicación revolucionaria que llamaremos Xenon y además dispone de una pequeña cantidad de dinero. Para que alguien desarrolle esta aplicación, Alice apuesta en Betfunding 1000 wei (125 $) que **nadie va a desarrollar** esa aplicación antes de una fecha límite.

Varias personas interesadas en esta aplicación también apuestan a que **no se va a realizar** Xenon.

A Bob le gusta mucho la programación, y explorando en Betfunding descubre que puede realizar Xenon, así que apuesta 160 wei a que si se hará a tiempo, y se pone a programar.

Eve es una especuladora que está buscando ganarse algo de dinero. Como ve que solo una persona ha apostado a que se realizaría, y piensa que el proyecto va a fracasar, también apuesta a que no se completará antes de la fecha límite. Llegados a este punto hay una cantidad de 2000 wei en la parte que dice que no se podrá hacer Xenon.

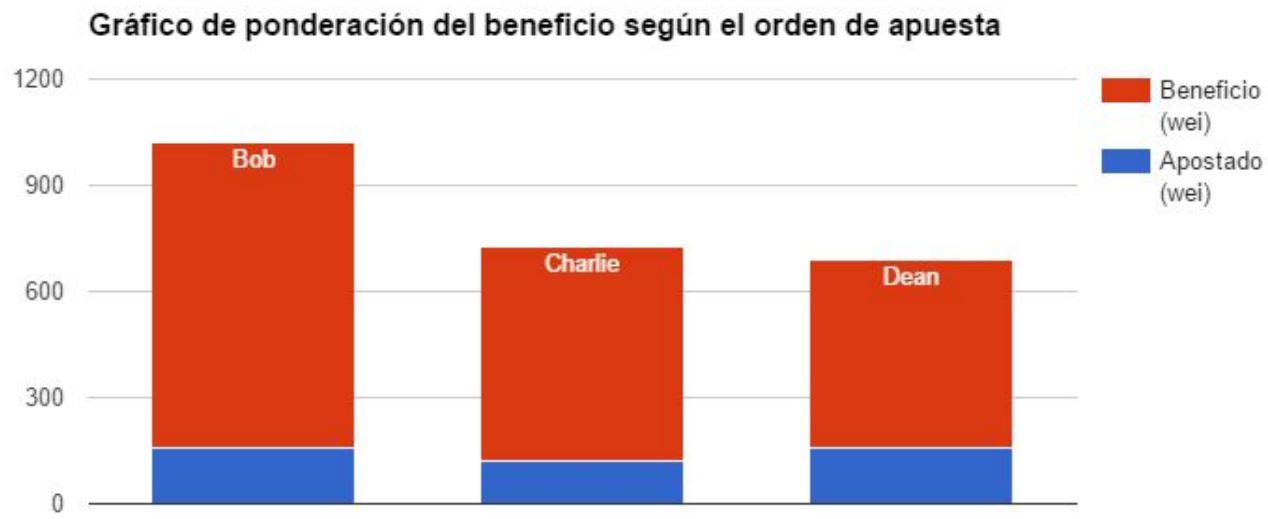
Chalie y Dean ven una oportunidad de ganar algo de dinero y apuestan 120 wei y 160 wei respectivamente. Ambos se ponen en contacto con Bob para reunir fuerzas y poder desarrollar la aplicación a tiempo.

Unidos los tres, consiguen verificar el proyecto antes del *deadline* y gracias a ello reciben una cantidad de dinero proporcional a la que ellos apostaron, ponderada según el orden en el que lo hicieron, para **beneficiar a los que apuestan antes** (ver la tabla siguiente).

Tabla de apuestas positivas del ejemplo teniendo en cuenta una cantidad de apuestas negativas de 2000 wei:

ID	POSICIÓN	Apostado (wei)	Apostado ($)	P-Orden	P-Cantidad	P-Media	Beneficio (wei)	Total recibido (wei)
Bob	1	160	20	0,5	0,363	0,431	862	1022
Charlie	2	120	15	0,333	0,272	0,302	604	724
Dean	3	160	20	0,166	0,363	0,264	528	688
Total	3	440	55	0,999	0,998	0,997	1994	2434

El siguiente gráfico ilustra que la cantidad obtenida al completar el proyecto:



Beneficio de las tres personas que apostaron a que si se realizaba

Casos de uso

A continuación, en la figura 14 se muestran los diferentes casos de uso de forma gráfica para cada uno de los usuarios de la aplicación.

En la representación del usuario se ha diferenciado entre usuario normal, financiador, creador, especulador y juez en función del rol que toma el usuario en su interacción con la plataforma, siendo el usuario normal aquel que todavía no ha interactuado con el proyecto y cuya acción puede ser realizada por cualquiera de los otros tipos de usuario.

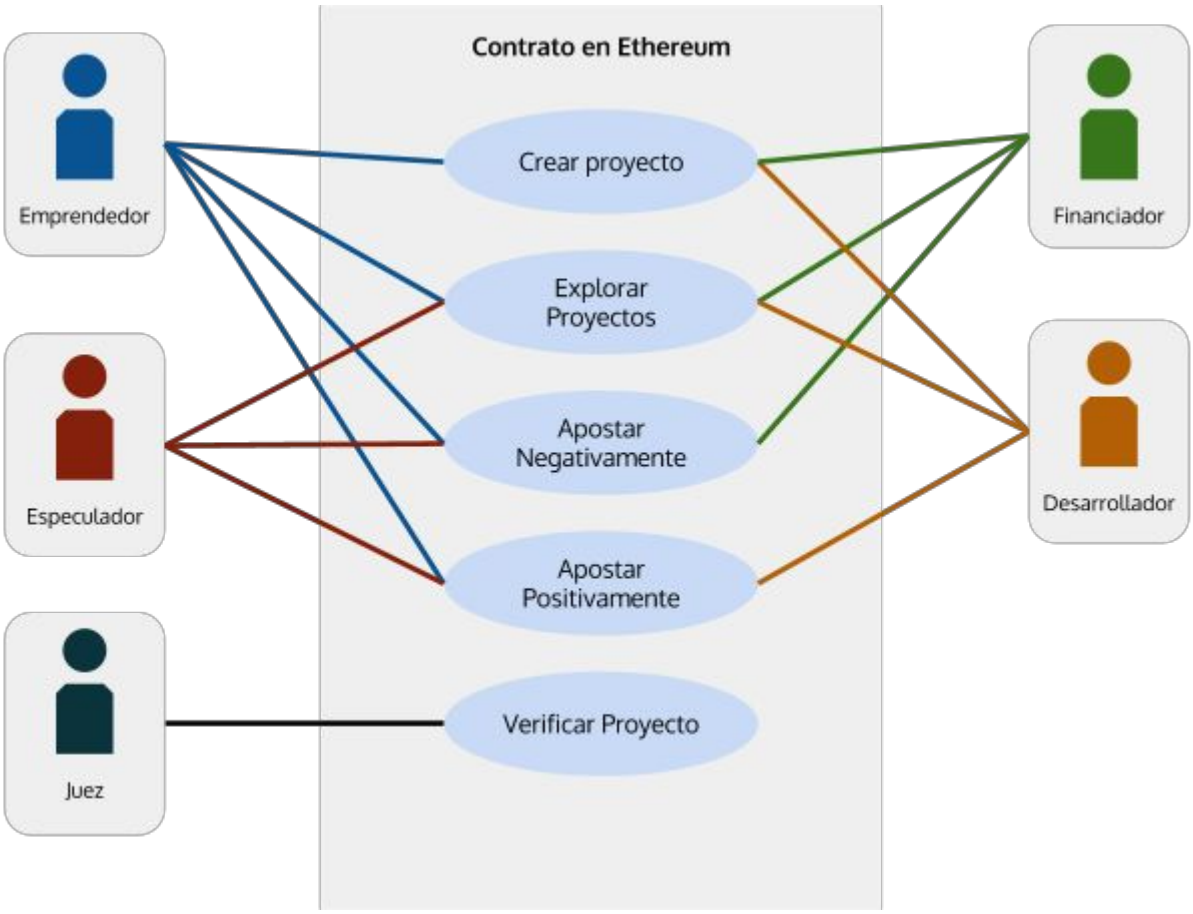


Figura 14. Diagrama con los casos de uso de la aplicación

Implementación

Estructura de la aplicación

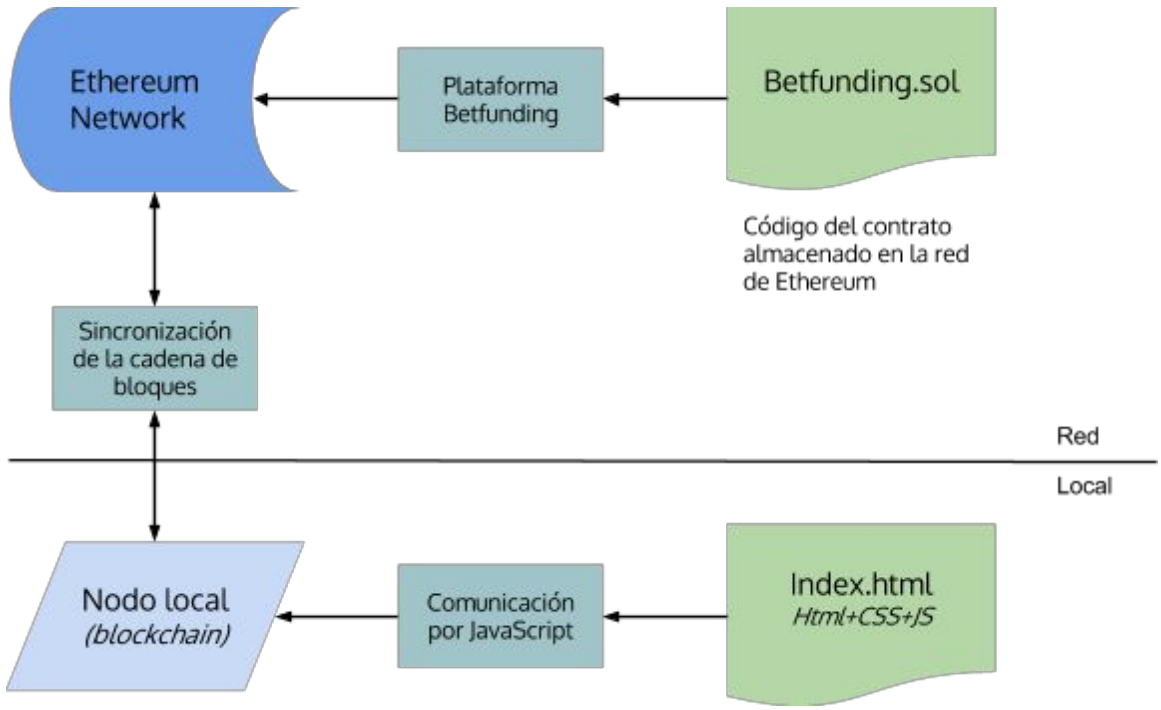


Figura 15. Estructura de de la aplicación Betfunding

Como puede verse en la figura 15, la aplicación está dividida en **dos partes**, el **contrato** (Betfunding.sol) que será el encargado de dar **funcionalidad** a la aplicación, y la **interfaz** (index.html) que permitirá a los usuarios **interactuar** con el contrato de una forma visual e intuitiva.

Para la interacción entre ambas partes es necesario contar con un nodo local de Ethereum que se sincronice con el resto de nodos de la red.

Contrato

Haciendo un símil con la estructura de una aplicación web tradicional, el contrato podría ser el *backend* de la aplicación que se ejecute en el lado del servidor. En el caso de Betfunding, al tratarse de una aplicación descentralizada **no existen servidores centrales**, en su lugar la sincronización de los datos se realiza de forma **P2P entre los usuarios.**

El código del contrato está escrito en el lenguaje de programación Solidity, que contiene la información sobre los proyectos creados, apuestas y las funciones

que indican cómo se comportará la aplicación. También tiene capacidad para **recibir y enviar dinero**.

El contrato se envía a la cadena de bloques mediante una **transacción**, donde una vez subido, pasa a funcionar de forma **autónoma** conforme a las reglas ya establecidas en el código. En este punto el contrato es autónomo y **ni siquiera el desarrollador puede modificar su comportamiento** o acceder al dinero almacenado.

Interfaz

Es **similar a la de cualquier aplicación web**, emplea tecnologías del lado del cliente ampliamente conocidas como HTML, CSS y Javascript. Esta interfaz es interpretada mediante el cliente de Ethereum que cuenta con un navegador web incorporado.

La interfaz se comunica con un **nodo local de Ethereum** mediante la librería web3 que trae incorporada el cliente de Ethereum. Para ello se crea una **instancia del contrato** con el ABI que indique el formato y la forma de llamar a las funciones.

Red Ethereum

La red Ethereum representa a los diferentes **nodos conectados entre sí,** cada uno de ellos con su copia de la cadena de bloques.

Los nodos se conectan de forma P2P sin una parte centralizada. **Los datos** resultantes de las transacciones realizadas **se sincronizan** mediante el proceso de **minado** explicado más arriba.

Nodo local

El nodo local es el nodo **alojado en el ordenador del usuario** y es su representación en la red de Ethereum. Desde él el usuario puede acceder a sus cuentas y comunicarse con el resto de la red.

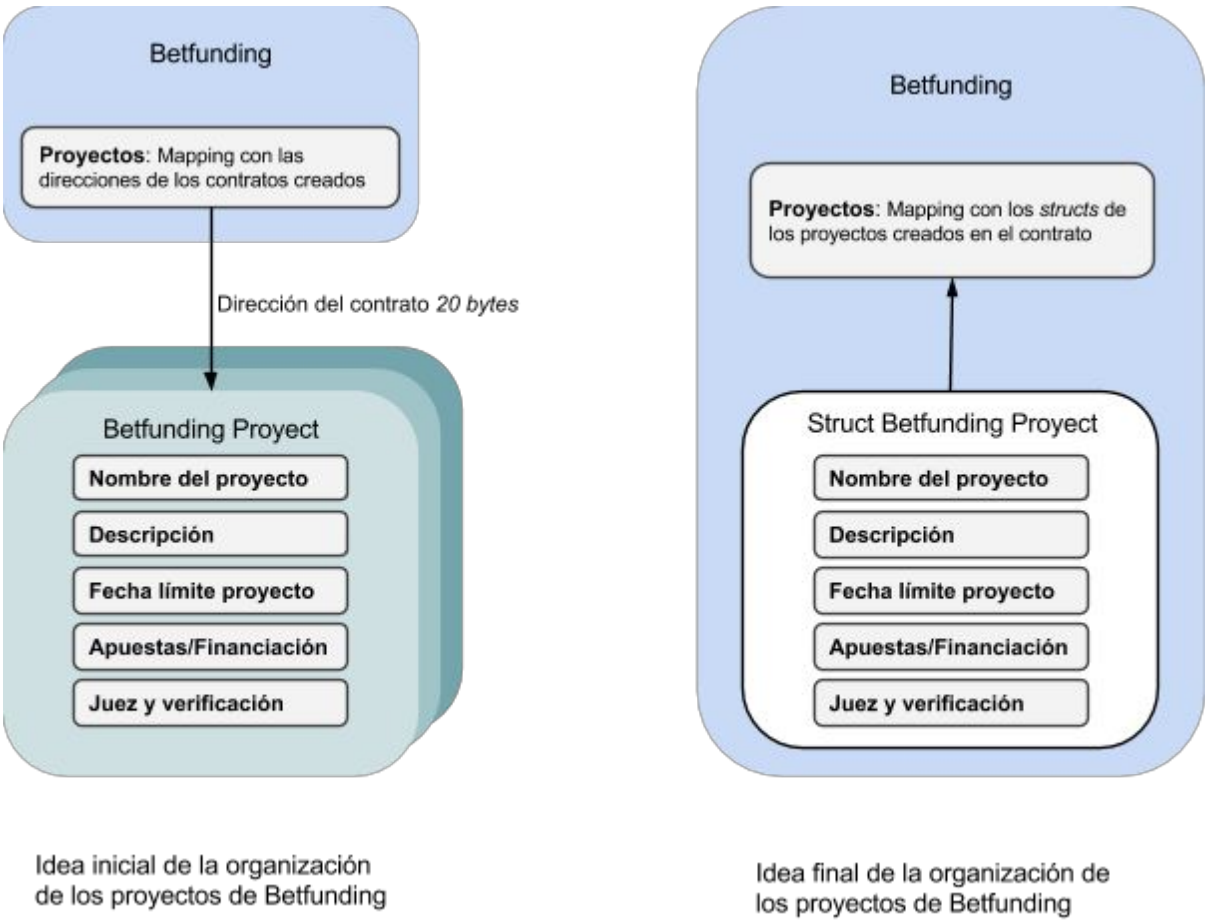
Cada nodo que forma parte de la red aloja una copia de la cadena de bloques con el registro de todas las transacciones, lo cual **incluye el código de todos los contratos y su estado**. El nodo alojado en el ordenador del usuario tendrá una

copia del contrato con el que se comunicará a través de la interfaz. La sincronización de los datos resultantes de la interacción entre el usuario y su nodo serán sincronizados a través de la red Ethereum.

Organización de proyectos en el contrato

La idea original que nos planteamos en el primer prototipo de Betfunding era la de emplear dos **tipos de contratos**. Un **contrato principal** que llevaría un registro de los proyectos creados mediante una lista y un segundo tipo de contrato para **cada proyecto** con información y un registro de las apuestas. Sin embargo, la versión que empleamos de Solidity **no permite la creación y uso de un contrato desde otro.**

La solución fue crear **un solo contrato** que guardase un registro de todos los proyectos y la información de cada uno de ellos mediante una **estructura de datos** como se muestra en la figura 16.



Idea inicial de la organización de los proyectos de Betfunding

Idea final de la organización de los proyectos de Betfunding

Figura 16. Evolución de la organización de los contratos en Betfunding

Estructura de un proyecto en el contrato

La estructura de datos principal dentro del contrato es la de los proyectos de Betfunding. Al crear un proyecto, la aplicación pide al usuario cierta información para poder almacenar el proyecto dentro del contrato en la cadena de bloques:

- **Creador del proyecto** *(address)*: Dirección de la persona que crea el proyecto. Esta información se obtiene automáticamente detectando la dirección desde donde se manda la petición.

- **Nombre de proyecto** *(String32)*: Es una cadena de texto que se almacena para saber como se llama cada uno de los proyectos. Esta cadena es la que se mostrará en el frontend a la hora de explorar los proyectos. Tiene un límite de 32 caracteres.

- **Descripción del proyecto** *(String32)*: Una de las limitaciones a la hora de almacenar la información de los proyectos en la cadena de bloques (al menos en la versión actual de Ethereum), es guardar en esta grandes cadenas de texto. Al ver que la limitación estaba en 32 bytes hasta el momento, pensamos que lo más lógico era escribir la descripción del proyecto en un *pastebin* o tecnología similar, y almacenar la URL de la página en la parte de descripción del proyecto en Betfunding.

- **Fecha límite** *(uint)*: Un número entero que indica la fecha máxima para poder completar el proyecto, el cual determinará si se ha completado a tiempo. Si el proyecto se verifica antes de la fecha límite, todas las personas que apostaron que sí se completaría reciben el dinero, de lo contrario el dinero de las apuestas se reparte entre las que apostaron a que no. Este número se calcula mediante el tiempo que tiene asociado cada bloque en Ethereum. Cuando se crea uno nuevo en la cadena de bloques, tiene asociado un valor que indica la fecha, valor que utiliza el contrato para ver si un proyecto es verificable o no.

- **Juez del proyecto** *(address)*: Esta dirección es la de la persona o contrato encargado de verificar un proyecto. En el momento de la creación del proyecto en el contrato, es necesario indicar la dirección del juez, ya que sin esta, el proyecto no podrá ser verificado y todas las personas que apostaron a que se realizaría perderían su dinero.

- **Apuestas del proyecto**: A la hora de apostar en un proyecto tenemos dos partes importantes:

    - **BadGamblers**: Son las personas que quieren aportar dinero para la realización de un proyecto, es decir, quienes apuestan a que un proyecto no se va a realizar. La información de estas apuestas se guarda mediante dos tablas que relacionan orden de llegada con la dirección del apostante y el apostante con la cantidad de dinero apostada.

    - **NiceGamblers**: Son las personas que apuestan a que un proyecto se va a completar dentro de la fecha límite y trabajan para la realización de este. El funcionamiento de las apuestas a análogo al de *BadGamblers.*

Mockups

La idea inicial era adoptar una imagen parecida a una **web estándar**, con un menú principal, los últimos proyectos, una sección sobre nosotros, etc. A través de la herramienta MyBalsamiq (75) creamos una **idea inicial** del aspecto que tendría nuestra aplicación y lo tomamos como referencia a la hora de crear el frontend (ver figura 17).

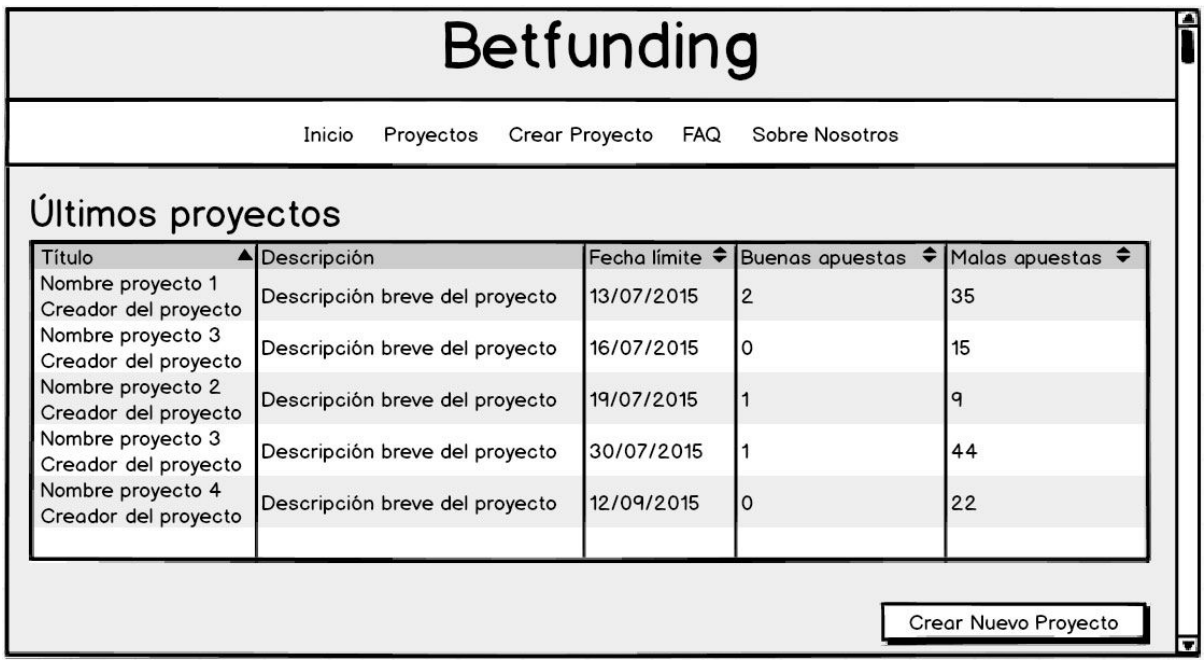


Figura 17. Mockup de Betfunding creado con MyBalsamiq

Pero una vez iba avanzado el diseño, decidimos hacer que la estructura de la aplicación tuviera una sola página (estilo “OnePage”), ya que **facilitaba la ejecución de la aplicación** en diferentes terminales, y era mucho más fácil de trasladar.

El frontend de la aplicación se encuentra en **un archivo html** que tiene que tener el usuario localmente, así que para ejecutar la página en un ordenador primero hay que descargar todas las páginas de la aplicación, por eso decidimos **simplificar todo el código** en un solo html (ver figura 18).

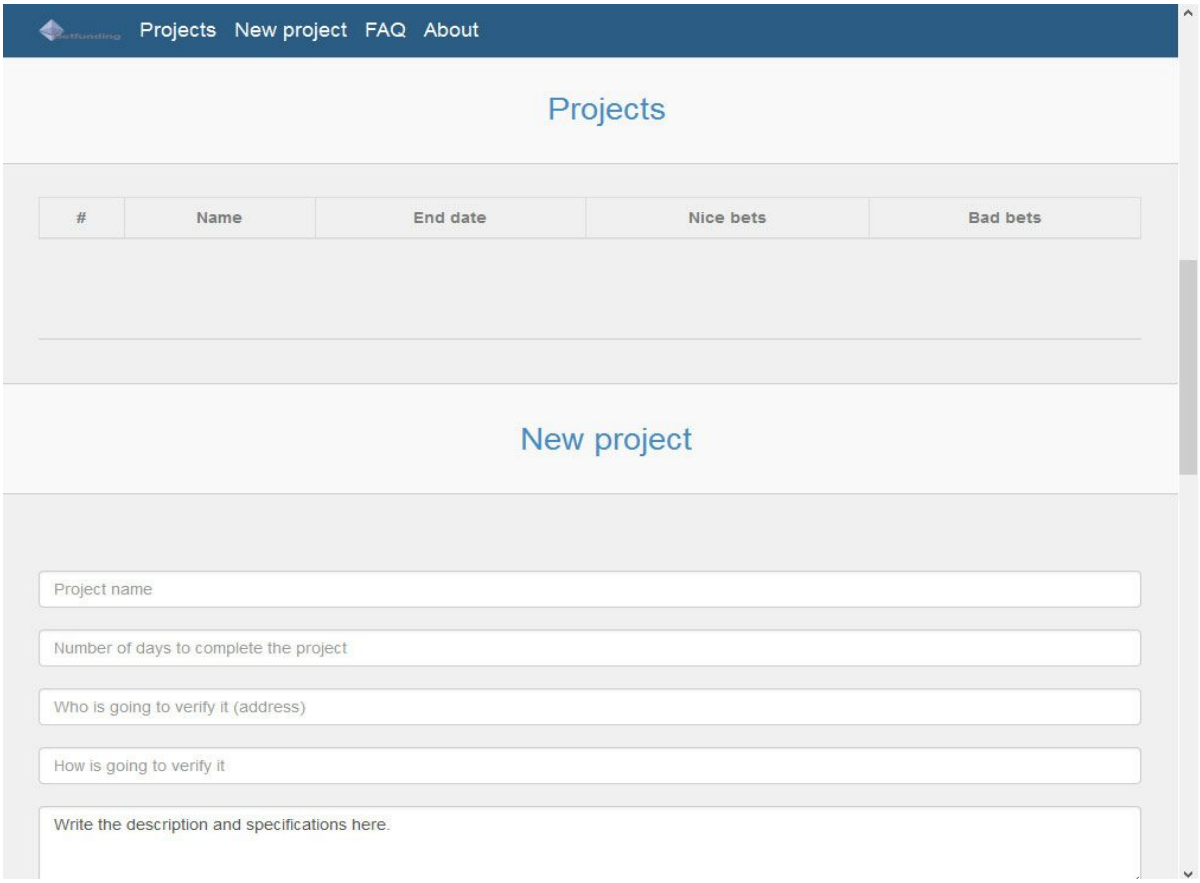


Figura 18: Diseño final de la aplicación Betfunding

Debido a que Betfunding tiene un funcionamiento diferente a otras plataformas de *crowdfunding*, decidimos añadir una **sección de preguntas frecuentes** (FAQ) en el diseño de la aplicación en el que explicamos, no solo el funcionamiento de la plataforma, sino cómo crear un proyecto, explorar todos los proyectos y el significado de las apuestas en cada uno de ellos (ver figura 19).

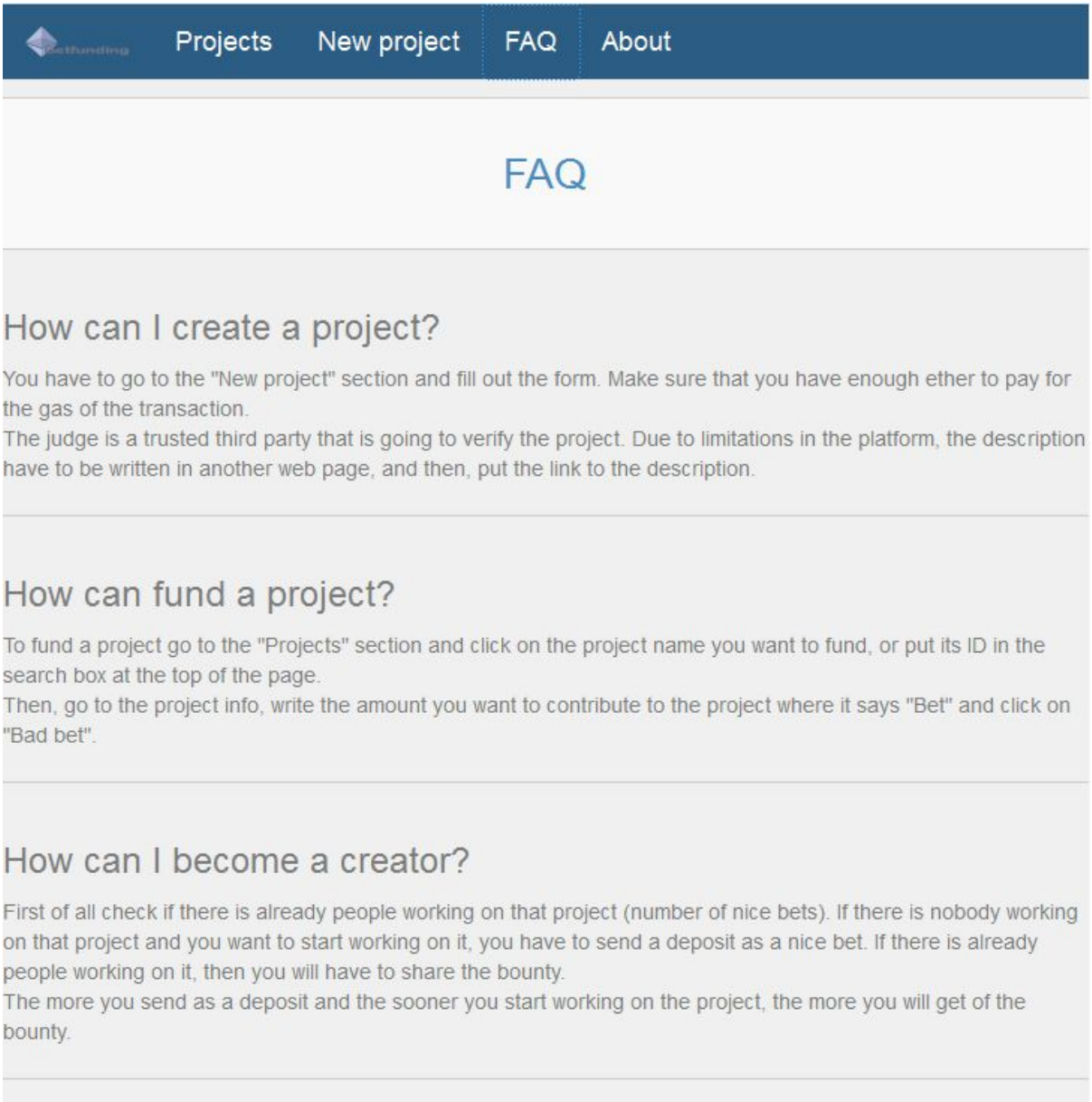


Figura 19. Sección FAQ de Betfunding

Ejemplo de uso

Creación de la plataforma

Para crear subir la aplicación de Betfunding a la cadena de bloques de Ethereum, es necesario enviar una transacción y adjuntar el código del contrato (betfunding.sol) como se ve en la figura 20.

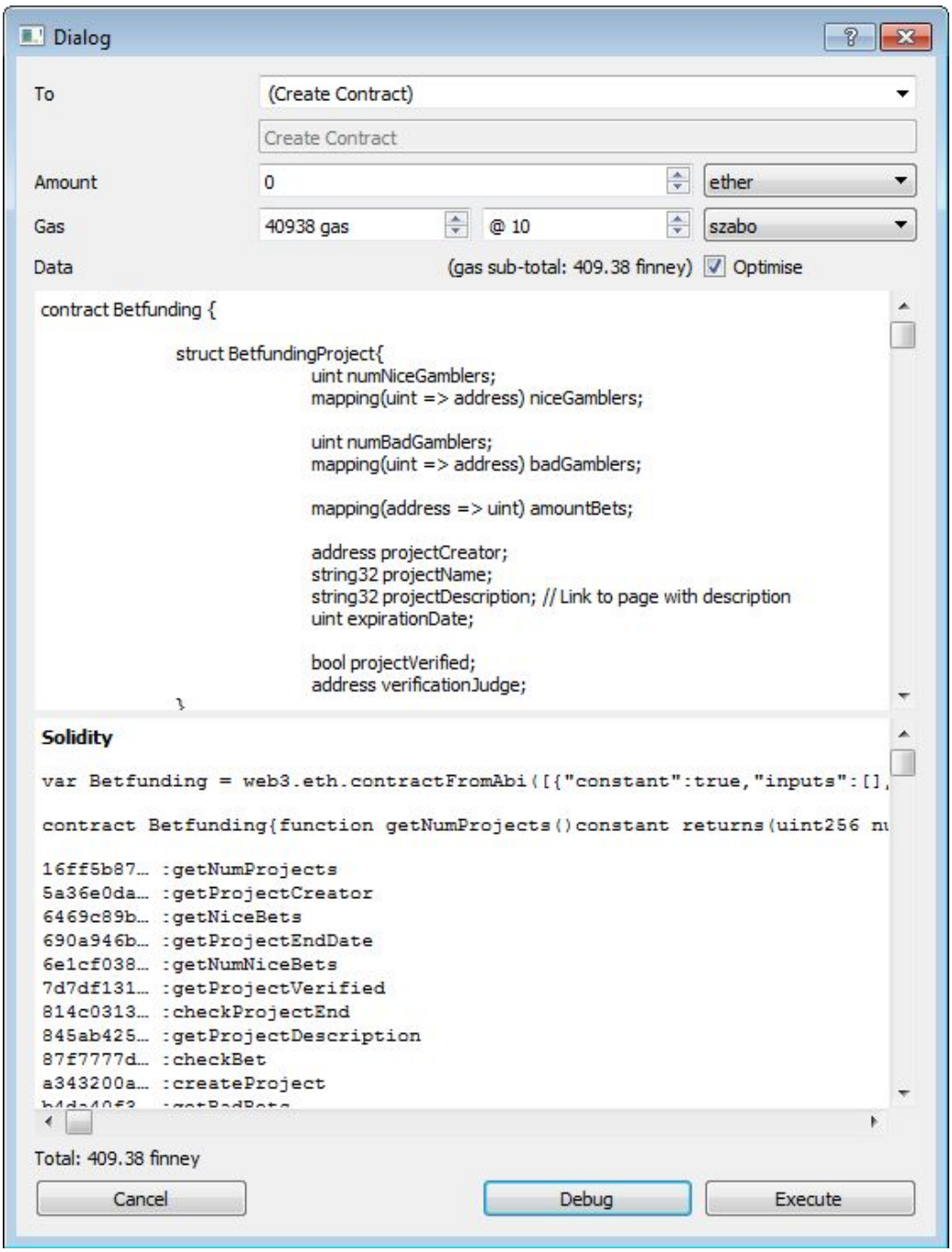


Figura 20. Creación de Betfunding

Una vez enviada la transacción con el código, la transacción quedará a la espera de ser confirmada por la red mediante el proceso de minado. Durante este tiempo, la transacción aparecerá como pendiente de confirmación (ver figura 21).

Figura 21. Transacción de creación de Betfunding pendiente

Cuando la transacción se haya minado, el código del contrato adjunto pasará a formar parte de la cadena de bloques. En la figura 22 puede verse el aspecto de nuestra aplicación tras haber sido alojada en la cadena de bloques.

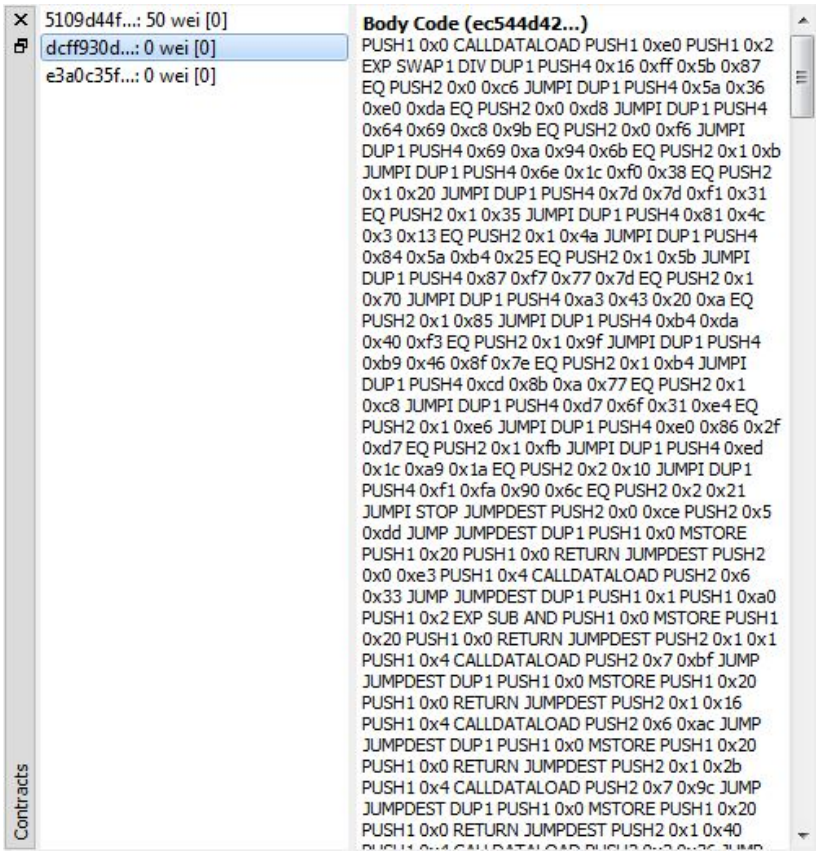


Figura 22. Información del contrato en la cadena de bloques

Una vez el contrato está alojado en la cadena de bloques, puede cargarse desde una interfaz gráfica para facilitar la comunicación con los usuarios. Para ello, basta con introducir la ruta del fichero (index.html) en el navegador del cliente de Ethereum. Antes hay que asegurarse de que la dirección del contrato está correctamente referenciada desde la interfaz (scripts.js) para crear una instancia del contrato de forma local que permita la comunicación. El aspecto que tendrá entonces la aplicación puede verse en la figura 23.

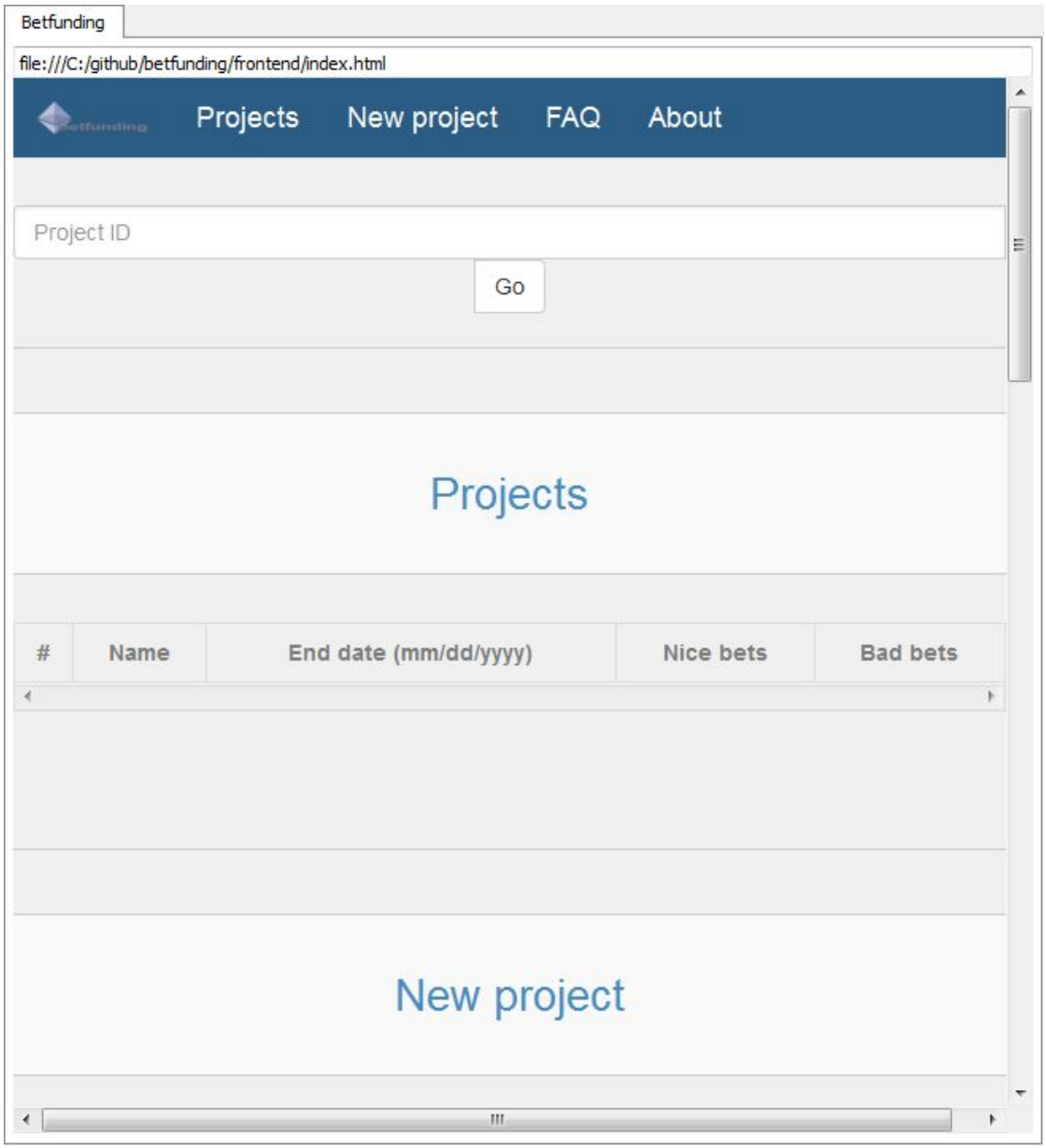


Figura 23. Betfunding

En este momento, todos los usuarios de la red de Ethereum que conozcan la dirección del contrato o tengan los ficheros de la interfaz, podrán hacer uso de la aplicación.

Interacción con la plataforma

Para la creación de un proyecto, el usuario tiene que rellenar un formulario con la información y pulsar un botón. Esto generará una transacción hacia el contrato con la información del formulario adjunta. La información necesaria es: Nombre del proyecto, número de días para su realización, dirección del juez, y URL donde se puede ver la descripción del proyecto y el método de verificación que seguirá el juez.

El cliente de Ethereum generará una ventana de confirmación informando al usuario del coste de dicha acción como se ve en la figura 24.

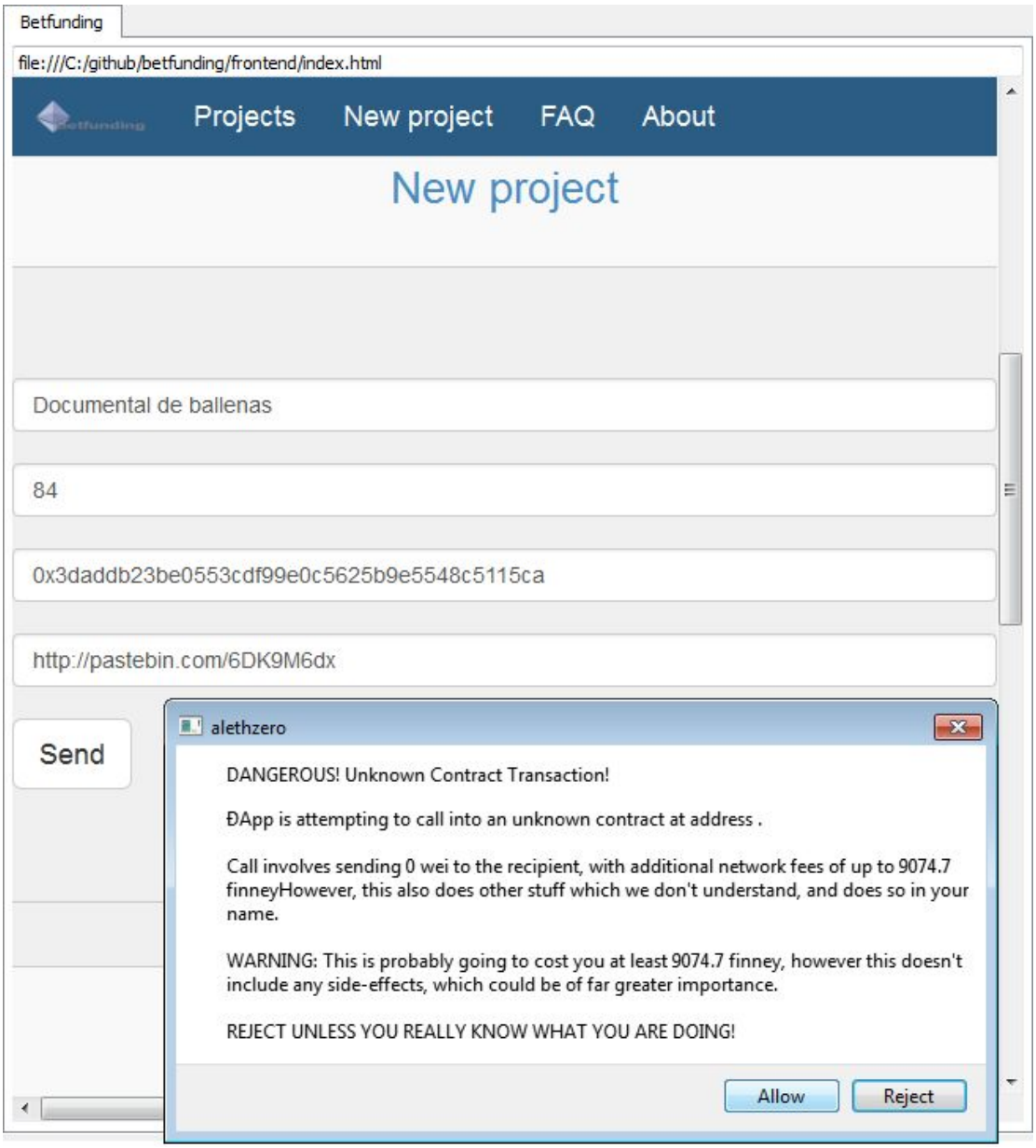


Figura 24. Creación de un proyecto en Betfunding

Como toda acción que requiera del envío de una transacción, es necesario esperar a que se confirme por la red mediante el proceso de minado.

Una vez confirmada la transacción, la información del proyecto quedará guardada en el contrato. En la figura 25 puede verse como el proyecto ha sido añadido a la lista de proyectos en la interfaz gráfica (izquierda) y cómo la memoria interna del contrato ha sido modificada en la cadena de bloques (derecha).

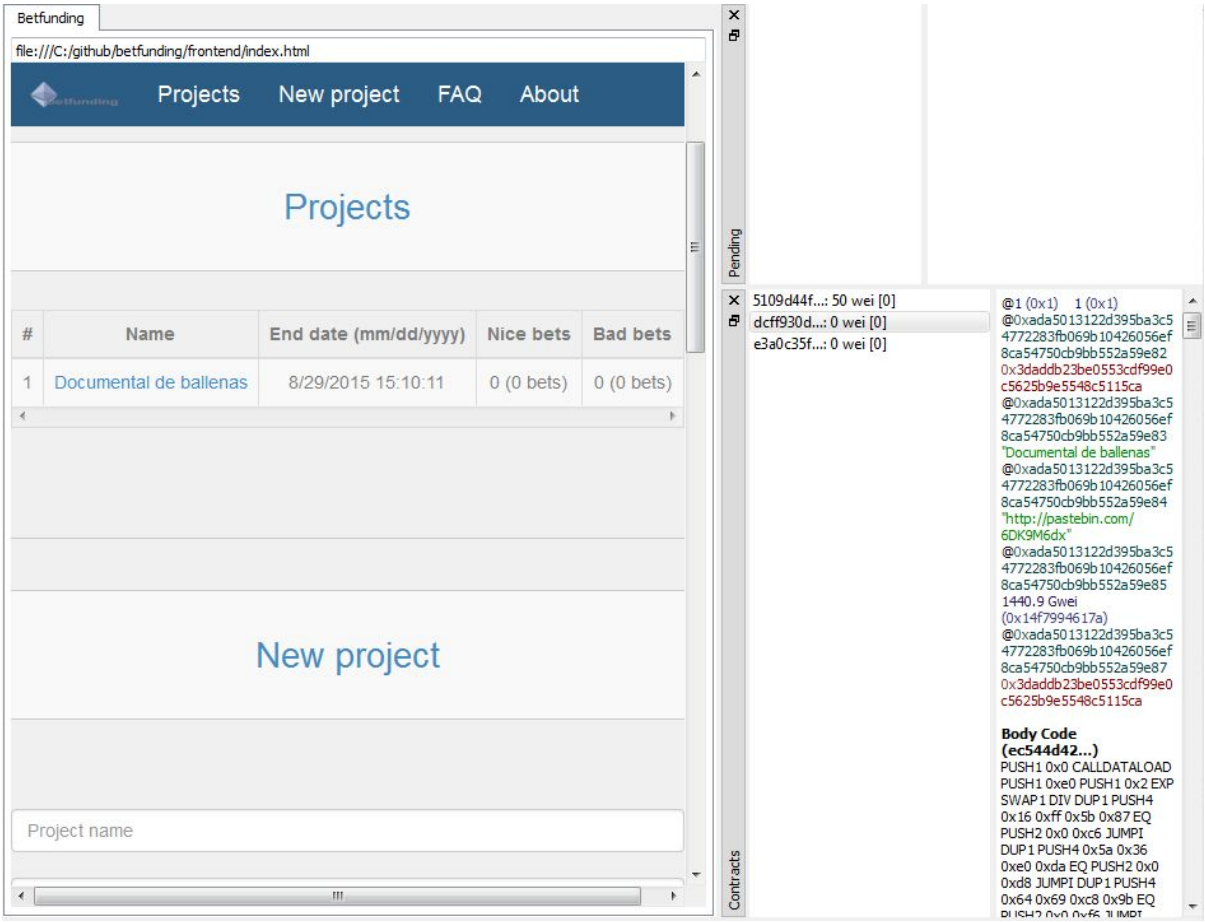


Figura 25. Nuevo proyecto en Betfunding

Para acceder a la información completa de un proyecto basta con hacer click sobre el nombre o introducir su ID en el cuadro de búsqueda. Se desplegará una caja con información sobre el proyecto, estado y apuestas (ver figura 26). También aparecerá un formulario que permitirá hacer una apuesta para financiar el proyecto.

Al haber escogido como juez nuestra propia dirección, nos aparecerá un botón que indica que somos juez del proyecto para que podamos verificarlo cuando haya sido realizado.

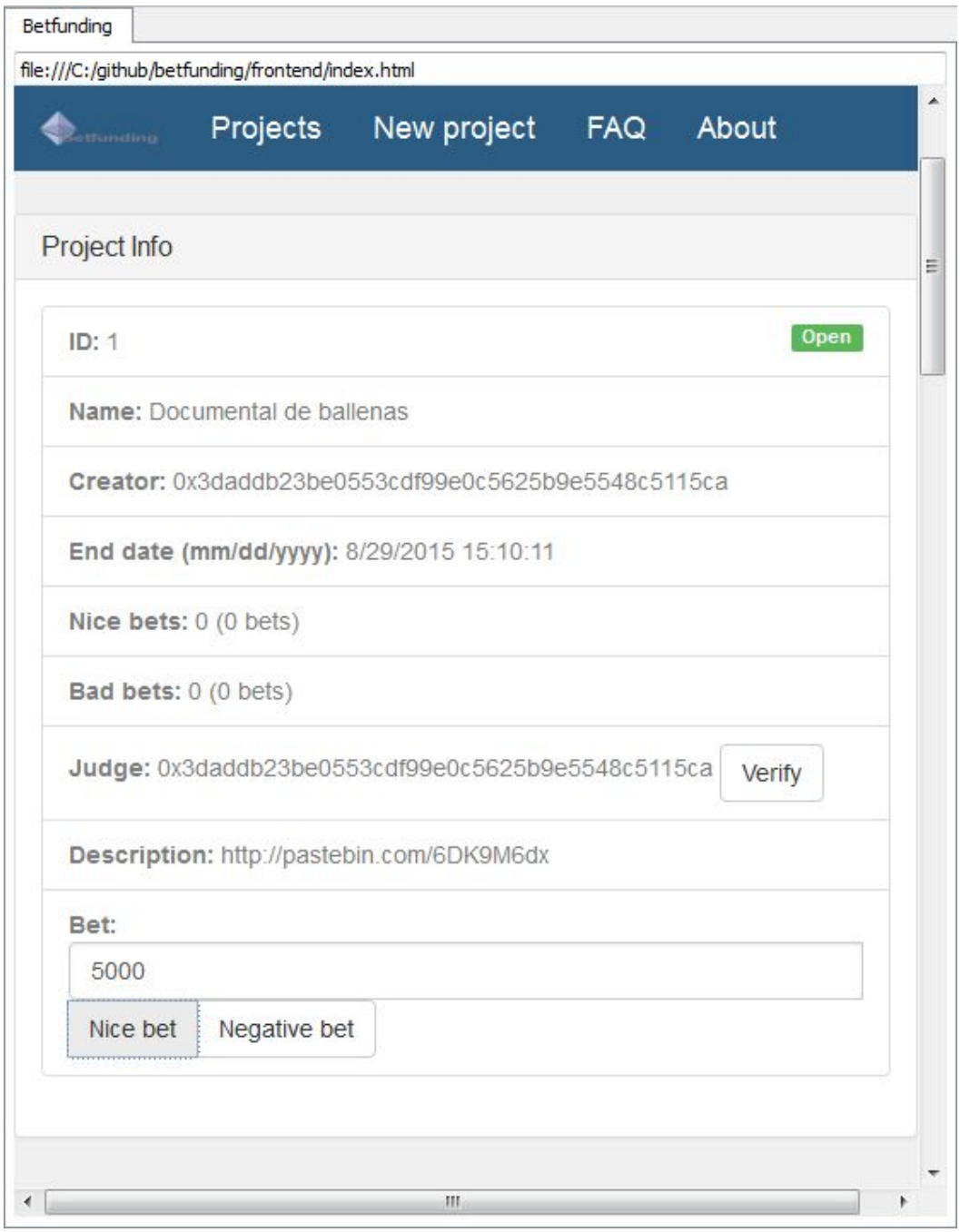


Figura 26. Información de un proyecto en Betfunding

Cuando se realiza una apuesta, se envía una transacción por valor de la cantidad indicada en el formulario. En la figura 27 puede verse la información del contrato actualizada tras recibir una transacción como “Nice bet” por valor de 5000 wei. En la parte de la derecha se puede ver como el contrato con dirección

“dcff930d...” ahora tiene a su derecha el valor “5000 wei”, que indica la cantidad de dinero que tiene el contrato.

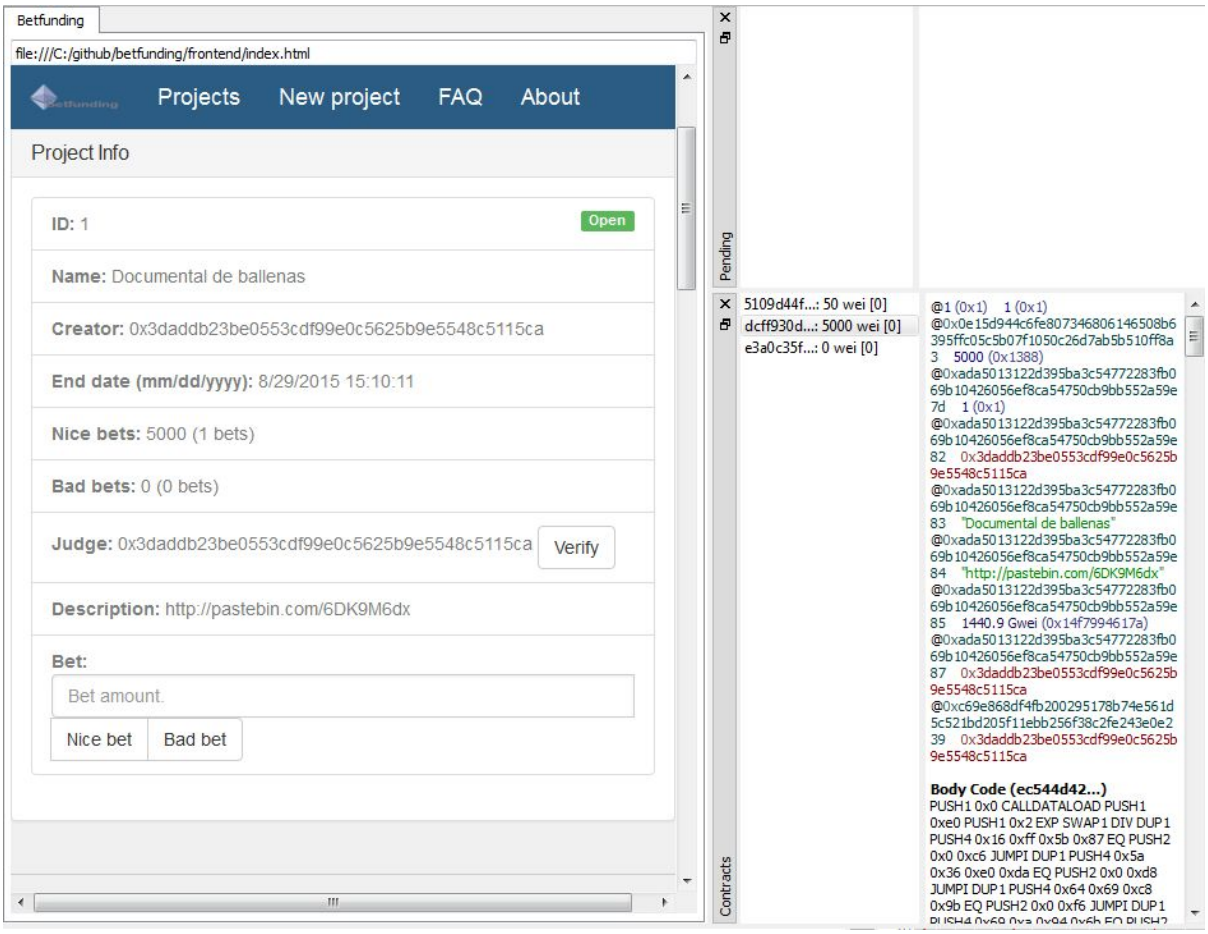


Figura 27. Información de un proyecto tras realizar una apuesta

Cuando la realización del proyecto ha sido verificada por el juez o se ha acabado el tiempo de plazo para su realización, el estado del proyecto cambia a cerrado (ver figura 28). Cuando se cierra un proyecto ya no se permiten nuevas apuestas y aparece un nuevo botón “Ping” para que cualquier usuario pueda activar la función de reparto de los botes en función del resultado del proyecto.

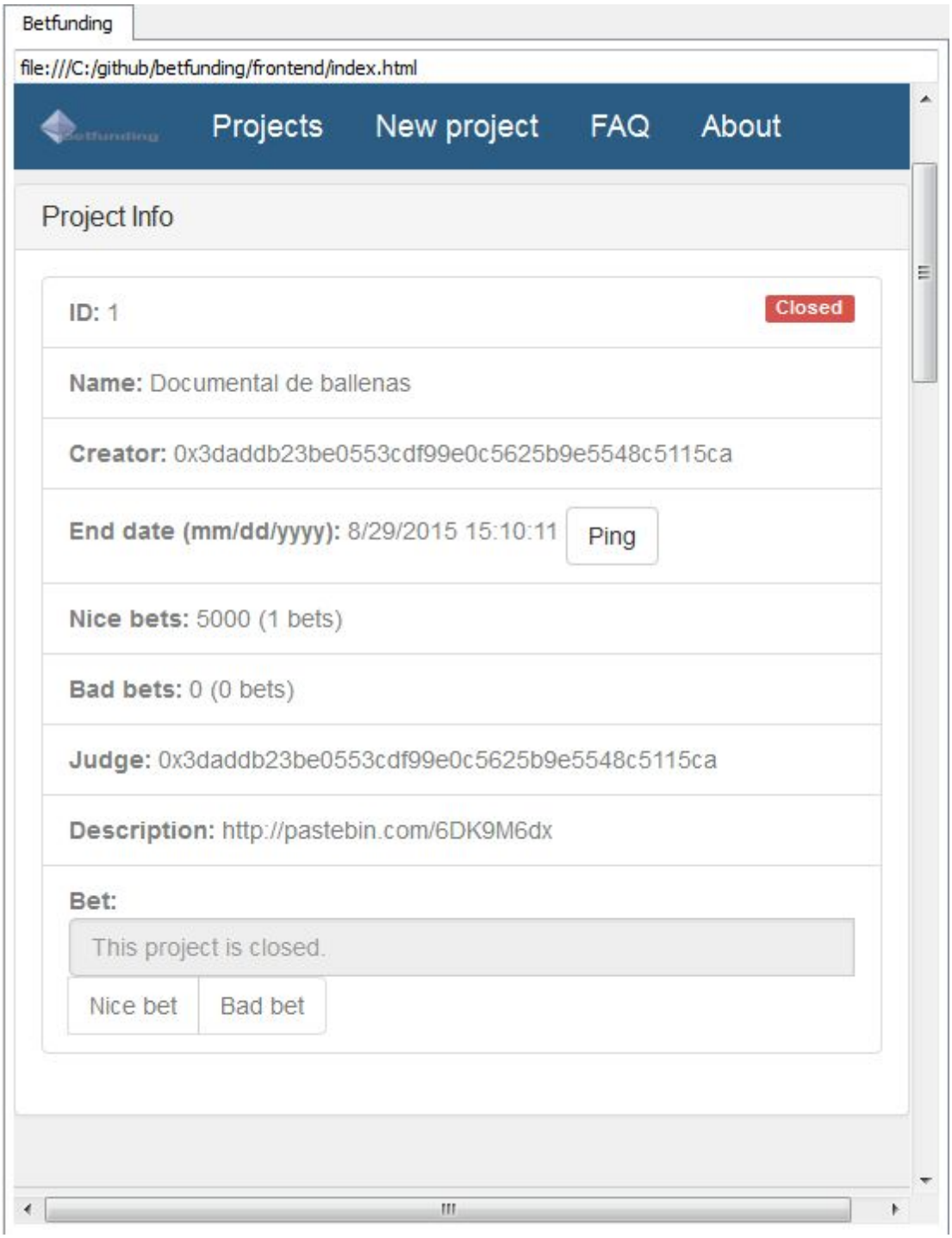


Figura 28. Contrato cerrado en Betfunding

Capítulo 6. Conclusiones y trabajo futuro

El primer objetivo de este Proyecto era **investigar, experimentar y aprender** sobre los **contratos inteligentes,** y la tecnología de la cadena de bloques. Este objetivo lo hemos **conseguido mediante la implementación de varias aplicaciones funcionales**: Un chat público en el que cualquier persona puede participar, un juego de azar donde se puede apostar dinero a través de un contrato inteligente, y una plataforma de crowdfunding como aplicación principal de este proyecto.

Gracias a esta plataforma final que hemos creado estamos intentando **promover la creación de proyectos** de una forma novedosa y atractiva. Esta aplicación es **totalmente funcional** y cualquier persona puede ponerla en funcionamiento, interactuar con ella y ganar dinero para financiar su proyecto.

También hemos comprobado que la viabilidad de esta plataforma de desarrollo nos da una **gran diversidad de posibles trabajos futuros**, que explicaremos más adelante, una vez evolucione a **versiones más estables**. A día de hoy esta tecnología todavía está en progreso ya que faltan muchas funcionalidades que prometieron los desarrolladores que aún están pendientes de implementar. También falta fomentar la utilización de esta plataforma ya que son los usuarios los que consiguen que Ethereum funcione.

En un principio comenzamos desarrollando **una sala de chat distribuida** sobre la cadena de bloques con el lenguaje de programación Serpent, donde los usuarios pueden comunicarse de forma P2P. Continuamos con un **juego de azar** similar a una lotería que guardaba el dinero de **forma segura sin terceras partes** de confianza mediante un contrato inteligente con los lenguajes de programación Serpent y Solidity. Después de eso desarrollamos Betfunding, una **plataforma de crowdfunding distribuida** que usa apuestas en lugar de donaciones para promover la producción artística y el trabajo creativo.

Para todos esos proyectos empleamos **Ethereum como plataforma de desarrollo** y *ether* como criptomoneda.

Viabilidad de una plataforma de *crowdfunding* mediante apuestas

A pesar de que uno de los objetivos de este proyecto no era poner a prueba la **viabilidad de una plataforma de *crowdfunding* usando apuestas,** llegamos a la conclusión de que **no es factible para todos los casos.**

Esto se debe a varios factores importantes. Primero, esta forma de financiación requiere la **confianza por parte del usuario** y dado que el funcionamiento interno de esta plataforma de *crowdfunding* en concreto es bastante complejo, no se espera que tenga una buena aceptación por parte de la comunidad de internet acostumbrada a usar este tipo de servicios.

Además dado que hay proyectos que requieren una **alta inversión**, el creador o creadores de los proyectos **no sabrán qué cantidad de dinero van a recibir** hasta la finalización del proyecto, ya que **existe la posibilidad de que los especuladores reciban una parte importante** de ese dinero sin haber contribuido a la creación del proyecto.

Sin embargo, Betfunding es una plataforma excelente para promover la creación de **pequeños proyectos** en los que haya un **gran número de personas interesadas** que puedan contribuir al mismo y que **requieran una inversión media-baja.** Un ejemplo muy actual es la creación de juegos "Indie" para dispositivos móviles y tablets, en los cuales se requiere un pequeño esfuerzo en desarrollo y tienen una gran aceptación en la comunidad de internet.

Experiencia con Ethereum

Cuando comenzamos a desarrollar el proyecto, la fecha esperada para la publicación de la versión 1.0 del cliente de Ethereum era octubre del 2014. Desde entonces, **ha habido sucesivos retrasos en su desarrollo** y a día de hoy todavía no hay disponible una versión estable**.**

Aun así, ha habido muchas mejoras desde el comienzo y hemos podido trabajar con la cadena de bloques desde diferentes puntos de vista usando Serpent y Solidity.

Desarrollar contratos de Ethereum actualmente está lleno de dificultades debido al estado en desarrollo en que se encuentra la plataforma. Además, **falta documentación** sobre muchas características y hay **inconsistencia** entre

diferentes versiones. Para poder solucionar los problemas de documentación tuvimos que recurrir al ensayo y error y a la investigación en los foros de los desarrolladores. Pese a ello, el balance ha sido positivo y pensamos que **tiene mucho potencial,** ya que las posibles aplicaciones abarcan desde la creación de **casinos**, otras **plataformas de financiación**, **criptomonedas personalizadas**, **contratación de servicios**, **intercambio de dinero**, **verificación electrónica de cualquier tipo, firmas electrónicas**, etc.

Viabilidad de la tecnología desde el punto de vista del desarrollador

Aunque Ethereum aún es una prueba de concepto, sus principales características ya **funcionan y permiten crear contratos** en la cadena de bloques. Los desarrolladores de la plataforma están desarrollando varias tecnologías adicionales como Swarm (protocolo de almacenaje descentralizado) o Whisper (protocolo de mensajería P2P) que complementarán las ya existentes.

Aparte de Bitcoin, **Ethereum es el proyecto más prometedor** basado en la tecnología de la cadena de bloques. Varias compañías como IBM o Samsung han mostrado interés en la plataforma para aplicar la tecnología al Internet de las cosas.

Trabajo futuro

Al haber implementado la aplicación sobre una plataforma en desarrollo, una gran parte del trabajo futuro pasa por ir adaptando y ampliando parte del trabajo realizado con las **nuevas funcionalidades** que se vayan incorporando.

Una de las tareas pendientes es mejorar el sistema de **verificación de los proyectos**. La solución ideal es leer una fuente RSS externa de confianza desde un contrato cuando sea posible. Otra posibilidad es integrar un **protocolo externo** como SchellingCoin junto a sistema de reputación distribuido para verificar los proyectos.

Con el desarrollo de Swarm por parte de los desarrolladores de Ethereum se podrá **almacenar la descripción de un proyecto dentro de la plataforma** sin depender de servicios externos. También abre la posibilidad de permitir la incorporación de vídeos, imágenes y audio a la descripción de los proyectos.

Usando Whisper se podría mejorar la **comunicación entre los creadores**, fomentando el trabajo colaborativo entre ellos. También permitirá a los usuarios añadir comentarios en los proyectos.

Actualmente nuestra aplicación sólo usa Ether como moneda, pero en un futuro podría hacer uso de algún tipo de *exchange* descentralizado como Coinffeine para que **cada usuario use la moneda de su preferencia**. Incluso podría ser posible aceptar monedas estilo Colored coins para permitir el uso de activos.

Chapter 6. Conclusions and future work

The first objective of this project was to **investigate**, **experiment and learn** about **smart contracts**, and blockchain technology. We have achieved this goal by **implementing several functional applications**: A public chat in which anyone can participate, a gambling game where you can bet money through an intelligent contract and a crowdfunding platform as the main application of this project.

Thanks to this crowdfunding platform we are trying to **promote the creation of projects** in a new and attractive way. This application is **fully functional** and anyone can make it work, interact with it and earn money to fund his project.

We have also found that the viability of this development platform gives us a **great variety of possible future work**, explained later, once it evolves to more **stable versions**. Today, this technology is still in progress and there are many features that are not implemented yet. This network also needs to be promoted to have a huge community because the use of this platform makes Ethereum work.

We started developing a **distributed chat room** on the blockchain with Serpent programming language, where users can send messages in a P2P way. We continue with a **gambling game** similar to a lottery that uses smart contracts to store the money in a **secure way without trusted third parties**, using Serpent and Solidity programming languages. After that, we developed a **distributed crowdfunding platform** that uses bets instead of donations to support artistic production and creative work.

For all those projects we used **Ethereum as the development platform** and ether as cryptocurrency.

Feasibility of a crowdfunding platform using bets

Besides one of the aims of this project was not to test the **feasibility of a crowdfunding platform using bets** instead of bids, we have concluded that **it is not feasible for all cases.**

This is due to several important factors. First, this form of financing requires **confidence by the user** and because the internal work of this particular

crowdfunding platform is quite complex, is not expected to be well accepted by the Internet community used to using this type of services.

Moreover since there are projects that require **high investment,** the creator or creators of the projects **will not know how much money they will receive** until the end of the project, since **there is the possibility that speculators receive a significant portion of that money** without helping to the completion of the project.

However, Betfunding is an excellent platform to promote the creation of **small projects** where there is a **large number of stakeholders** that can contribute to it and which require **medium-low investment**. A very recent example is the creation of the "indie" games for mobile and tablet devices, in which a small development effort is required and are widely accepted in the Internet community.

Experience with Ethereum

When we started developing the project, the expected date for the release of the version 1.0 of the Ethereum client was on October 2014. Since then, there have been **many delays in its development** and nowadays is still a **proof of concept**.

Nevertheless, there have been a lot of improvements since the beginning and we have been able to work with the blockchain from different perspectives using Serpent and Solidity.

Developing Ethereum contracts is full of difficulties due to the in-development nature of the platform. Also, there is a **lack of documentation** for many features and **inconsistency** between different versions. In order to solve this problem we had to use a trial and error process and searching in the developer's forums. Despite that, the balance has been positive and we think that **it have a lot of potential**, range from the creation of **casinos**, other **financing platforms**, **custom cryptocurrencies**, **hiring services, money exchange,** any type of **electronic verification**, **electronic signatures**, etc.

Viability of the technology from the point of view of the developer

Even though Ethereum is still a proof of concept, its main features already **work and allow creating contracts** on the blockchain. The Ethereum developers are developing some additional technologies such as Swarm (decentralized storage protocol) or Whisper (P2P messaging protocol) that complements the ones that already has.

Apart from Bitcoin, **Ethereum is the most promising project** based on blockchain technology. Some companies like IBM or Samsung have shown interest on the platform to use its technology for the Internet of things.

Future work

Due to the fact that the application is implemented on a platform under development, a large part of the future work is going to be to adapt and extend part of the work already done with the **new features** as they are being added.

One of the pending tasks is to improve the **verification system** of the projects. The ideal solution is reading an external trusted RSS feed from a contract whenever possible. Another possibility is to integrate an **external protocol** like SchellingCoin with a distributed reputation system to verify the projects.

With the development of Swarm by the Ethereum developers it will be possible to **store the description of a project in the platform** without depending on external services. It also opens the possibility of allowing the inclusion of videos, images and audio to the description of the projects.

Using Whisper the **communication among creators** could be improved, encouraging collaborative work between them. It will also allow users to add comments to projects.

Currently, our application only uses Ether as currency, but in the future it could use some kind of decentralized exchange like Coinffeine so that **each user uses the currency of his preference**. It would even be possible to accept Colored coin-like currencies to allow the use of assets.

Bibliografía

1. DR. GAVIN WOOD. *ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER.* , 2014.

2. HEMER, J. *A Snapshot on Crowdfunding.* Working papers firms and region, 2011.

3. SCHWIENBACHER, A. and LARRALDE, B. *Crowdfunding of Small Entrepreneurial Ventures.* Rochester, NY: Social Science Research Network, 2010 ISBN ID16-99183.

4. JAMES, T. Far from the Maddening Crowd: Does the Jobs Act Provide Meaningful Redress to Small Investors for Securities Fraud in Connection with Crowdfunding Operations*. Boston College Law Review*, 2013, vol. 54, no. 4.

5. *Google Trends - Web Search Interest - Worldwide, 2004 - Present.* Available from:https://www.google.es/trends/explore.

6. *Stretch Goals - Roberts Space Industries.* Available from:https://robertsspaceindustries.com/funding-goals.

7. *Pebble Time - Awesome Smartwatch, no Compromises.* Available from:https://www.kickstarter.com/projects/597507018/pebble-time-awesome-smartwatch-no-compromises.

8. *Ethereum.* Available from:https://www.ethereum.org/ether#sale.

9. ERANTI, V. and LINDMAN, J. Crowdsourcing & Crowdfunding a Presidential Election.

10. *Financiación - Podemos.* Available from:http://podemos.info/financiacion/.

11. *Qué Es Kickstarter — Kickstarter.* Available from:https://www.kickstarter.com/hello?ref=footer.

12. *Bitcoin Crowdfunding - CoinFunder.* Available from:http://coinfunder.com/.

13. BARAN, P. On Distributed Communications Networks, 1964, vol. 12, no. 1. pp. 1-9 ISSN 0096-1965.

14. *DigiCash - an Introduction to Ecash.* [viewed May 28, 2015]. Available from:https://web.archive.org/web/19971009044558/http://digicash.com/publish/ecash_intro/ecash_intro.html.

15. W. DAI. *B-Money Proposal.* [viewed May 28, 2015]. Available from:http://www.weidai.com/bmoney.txt.

16. N. SZABO. *Unenumerated: Bit Gold.* December 27, 2005 [viewed May 28, 2015]. Available from:http://unenumerated.blogspot.com.es/2005/12/bit-gold.html.

17. H. FINNEY. *RPOW - Reusable Proofs of Work.* August 16, 2004 [viewed May 28, 2015]. Available from:http://cryptome.org/rpow.htm.

18. A. BACK. *[ANNOUNCE] Hash Cash Postage Implementation.* March 28, 1997 [viewed May 28, 2015]. Available from:http://www.hashcash.org/papers/announce.txt.

19. S. NAKAMOTO. *Bitcoin Open Source Implementation of P2P Currency - P2P Foundation.* February 11, 2009 [viewed May 28, 2015]. Available from:http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source.

20. *Ethereum White Paper.* [viewed May 15, 2015]. Available from:https://github.com/ethereum/wiki/wiki/White-Paper.

21. A. ANTONOPOULOS. *Mastering Bitcoin.* April, 2014 [viewed June 3, 2015]. Available from:http://chimera.labs.oreilly.com/books/1234000001802/ch02.html#_bitcoin_mining.

22. Y. BRIKMAN. *Bitcoin by Analogy.* April 24, 2014 [viewed June 2, 2015]. Available from:http://www.ybrikman.com/writing/2014/04/24/bitcoin-by-analogy/.

23. *Mining - Bitcoin Wiki.* [viewed June 3, 2015]. Available from:https://en.wikipedia.org/wiki/Bitcoin#Mining.

24. *Coinbase - Bitcoin Wiki.* [viewed June 3, 2015]. Available from:https://en.bitcoin.it/wiki/Coinbase.

25. *Block 0 - TEST Bitcoin Block Explorer.* [viewed May 28, 2015]. Available from:https://blockexplorer.com/testnet/block/000000000933ea01ad0ee984209779baaec3ced90fa3f408719526f8d77f4943.

26. *Testnet - Bitcoin Wiki.* [viewed May 28, 2015]. Available from:https://en.bitcoin.it/wiki/Testnet.

27. [announce] Namecoin - a distributed naming system based on Bitcoin. April 18, 2011 [viewed May 28, 2015]. Available from: https://bitcointalk.org/index.php?topic=6017.0.

28. [ANN] Litecoin - a lite version of Bitcoin. Launched!. October 9, 2011 Available from: https://bitcointalk.org/index.php?topic=47417.0.

29. C. EISENBERG. *Graphical Comparison of all Cryptocurrencies.* [viewed May 28, 2015]. Available from:https://www.cryptocoincharts.info/coins/graphicalComparison.

30. PÉREZ SOLÀ, C. and HERRERA JOANCOMARTÍ, J. *Bitcoins Y El Problema De Los Generales Bizantinos.* Universidad de Alicante, September 1, 2014 Available from: http://rua.ua.es/dspace/handle/10045/40444.

31. KING, S. *Primecoin: Cryptocurrency with Prime Number Proof-of-Work.* , 2013.

32. KING, S. and NADAL, S. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.* , 2012.

33. *FairCoin.* [viewed May 28, 2015]. Available from:https://fair-coin.org/.

34. LAMBERT, N., MA, Q. and IRVINE, D. *Safecoin: The Decentralised Network Token.* , 2015.

35. *Proof of Burn - Bitcoin Wiki.* [viewed May 29, 2015]. Available from:https://en.bitcoin.it/wiki/Proof_of_burn.

36. *Why Proof-of-Burn | Counterparty.* [viewed May 28, 2015]. Available from:http://counterparty.io/news/why-proof-of-burn/.

37. *Onename.* [viewed May 28, 2015]. Available from:https://onename.com/.

38. *Passcard.* [viewed May 28, 2015]. Available from:https://passcard.info/.

39. *Proof of Existence.* [viewed May 28, 2015]. Available from:http://www.proofofexistence.com/about.

40. *BTProof - Trusted Timestamping on the Bitcoin Blockchain.* [viewed May 28, 2015]. Available from:https://www.btproof.com/.

41. ASSIA, Y., BUTERIN, V., ROSENFELD, M. and LEV, R. *Colored Coins whitepaper .*

42. MIERS, I., GARMAN, C., GREEN, M. and RUBIN, A. *Zerocoin: Anonymous Distributed E-Cash from Bitcoin.*

43. P. RIZZO. *Mastercoin Seeks Second Start with Omni Reboot.* Coinbase: January 21, 2015 [viewed May 28, 2015]. Available from:http://www.coindesk.com/mastercoin-new-beginning-omni/.

44. WILLETT, J.R., et al. *The Master Protocol / Mastercoin Complete Specification.* Available from: https://github.com/OmniLayer/spec.

45. WILKINSON, S. *Storj A Peer-to-Peer Cloud Storage Network.* , December 15, 2014.

46. SZABO, N. *Formalizing and Securing Relationships on Public Networks.* , 1997 Available from: http://ojphi.org/ojs/index.php/fm/article/view/548/469.

47. N. SZABO. *The Idea of Smart Contracts.* , 1997 [viewed May 28, 2015]. Available from:http://szabo.best.vwh.net/smart_contracts_idea.html.

48. *Transaction - Bitcoin Wiki.* [viewed May 25, 2015]. Available from:https://en.bitcoin.it/wiki/Transaction.

49. *About Counterparty | Counterparty.* [viewed May 28, 2015]. Available from:http://counterparty.io/docs/about_counterparty/.

50. JOHNSTON, D., et al. *The General Theory of Decentralized Applications, Dapps.* , April 17, 2014 Available from:

https://github.com/DavidJohnstonCEO/DecentralizedApplications.

51. *MaidSafe.Net Announces Project SAFE to the Community.* , 2014 [viewed May 29, 2015]. Available from:https://github.com/maidsafe/Whitepapers/blob/master/Project-Safe.md.

52. THOMAS, S. and SCHWARTZ, E. *Smart Oracles: A Simple, Powerful Approach to Smart Contracts.* , 2014 Available from: https://github.com/codius/codius/wiki/Smart-Oracles:-A-Simple,-Powerful-Approach-to-Smart-Contracts.

53. *Upcoming Features | Nxt.Org.* [viewed May 29, 2015]. Available from:http://nxt.org/about/upcoming-features/.

54. *Eris Industries | Eris:Db.* [viewed May 29, 2015]. Available from:https://erisindustries.com/products/erisdb/.

55. *Swarm - Ethereum Wiki.* [viewed May 29, 2015]. Available from:https://github.com/ethereum/cpp-ethereum/wiki/Swarm.

56. *Whisper - Ethereum Wiki.* [viewed May 29, 2015]. Available from:https://github.com/ethereum/wiki/wiki/Whisper.

57. M. HEARN. *Contracts - Bitcoin Wiki.* Available from:https://en.bitcoin.it/wiki/Contracts#Example_3:_Assurance_contracts.

58. M. HEARN. *Lighthouse FAQ.* Available from:https://www.vinumeris.com/lighthouse/faq#max-pledges.

59. *Serpent - Ethereum Wiki.* [viewed May 28, 2015]. Available from:https://github.com/ethereum/wiki/wiki/Serpent.

60. *Solidity Features - Ethereum Wiki.* [viewed May 28, 2015]. Available from:https://github.com/ethereum/wiki/wiki/Solidity-Features.

61. *Ethereum - Ether.* [viewed May 28, 2015]. Available from:https://www.ethereum.org/ether.

62. *Using AlethZero - Ethereum Wiki.* [viewed May 28, 2015]. Available from:https://github.com/ethereum/cpp-ethereum/wiki/Using-AlethZero.

63. V. BUTERIN. *Introducing Ethereum Script 2.0 - Ethereum Blog.* February 3, 2014 Available from:https://blog.ethereum.org/2014/02/03/introducing-ethereum-script-2-0/.

64. *JavaScript API - Ethereum Wiki.* [viewed May 28, 2015]. Available from:https://github.com/ethereum/wiki/wiki/JavaScript-API.

65. *About - Bootstrap.* [viewed May 15, 2015]. Available from:http://getbootstrap.com/about/.

66. J. OIKARINEN and D. REED. *Internet Relay Chat Protocol.* , 1993 [viewed 2015].

Available from:https://tools.ietf.org/html/rfc1459.

67. *Cryptocat.* [viewed 2015]. Available from:https://crypto.cat/.

68. *Terra Chat.* [viewed 2015]. Available from:http://www.terra.es/chat/.

69. *Lycos Chat.* [viewed 2015]. Available from:http://chat.lycos.es/.

70. *Satoshi Dice - Bitcoin Wiki.* [viewed June 2, 2015]. Available from:https://en.bitcoin.it/wiki/Satoshi_Dice.

71. *Bitcoin Casino - Satoshibet.* [viewed 2015]. Available from:https://satoshibet.com/.

72. *FortuneJack.* [viewed 2015]. Available from:https://fortunejack.com/.

73. V. BUTERIN. *Toward a 12-Second Block Time - Ethereum Blog.* July 11, 2014 [viewed May 25, 2015]. Available from:https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/.

74. PETERSON, J. and KRUG, J. *Augur: A Decentralized, Open-Source Platform for Prediction Markets.* Available from: http://www.augur.link/augur.pdf.

75. *myBalsamiq - Balsamiq.* [viewed May 25, 2015]. Available from:https://balsamiq.com/products/mockups/mybalsamiq/.