Projecte Final de Grau
Grau en Enginyeria Informàtica
Universitat Oberta de Catalunya (UOC)

Disseny i implementació d'un Smart Contract de micromecenatge sobre Ethereum.

Autor: Pere Bertran Sole **<pbertran1@uoc.edu>**

Supervisió: Cristina Pérez Solà <cperezsola@uoc.edu>

02 de Gener de 2017

Llicència

Del present document

D’obres de tercers

En el document s’inclouen extractes de codi font i d’obres de tercers, cadascun dels quals està degudament identificat i subjecte a la llicència escollida pels seus autors.

Agraïments

A Cristina Pérez Solà, supervisora del projecte, per les detallades correccions i suport realitzats en el desenvolupament del projecte.

També vull agrair als desenvolupadors que contribueixen en el projecte Ethereum i la seva dedicació a documentar i compartir els coneixements i avenços que van realitzant en el projecte.

Resum.

Aquest projecte té com a objectiu avaluar Ethereum, una plataforma descentralitzada per al desplegament de contractes intel·ligent en la Blockchain. L'enfocament d'aquest treball consisteix a iniciar una xarxa privada Ethereum de prova (Private-Net), en la que desenvoluparem i desplegarem una aplicació descentralitzada , un contracte intel·ligent de micromecenatge. Els passos cap a la construcció de l'entorn de desenvolupament, així com l'aplicació desenvolupada del contracte es detallarà en aquest informe. Les experiències adquirides, així com la estimació sobre el potencial del projecte Ethereum es presenten en la avaluació del treball.

Abstract.

This project aims to evaluate Ethereum, a decentralized platform for the deployment of Smarts Contracts on the Blockchain. Approach of this work is to start a private network Ethereum test (Private-Net), in this network we develop and deploy an application decentralized: crowdfunding Smart Contract. The steps towards building the development environment and developed the application contract will be detailed in this report. Gathered experiences, as well as estimate the potential of the project Ethereum are presented in the job evaluation.

Índex

Llista de taules

Llista de figures

Llista de fragments de codi.

1-Introducció.

Un Smart Contract és un contracte capaç d'executar-se i fer-se complir per si sol de forma autònoma i automàtica, sense la necessitat que intervinguin mediadors o mitjancers. Els Smarts Contracts realment són programes informàtics de tal forma que els requeriments del contracte són sentències u ordres inclosos en el seu codi.

Els Smart Contracts tenen validesa sense dependre de cap autoritat donat que és un codi visible per a tothom, el qual no es pot canviar ja que utilitza la tecnologia *Blockchain* que li dona aquest caràcter descentralitzat, immutable i transparent.

Ethereum és una plataforma web que permet als desenvolupadors crear i publicar aplicacions distribuïdes sota tot un seguit de normes fixades en el codi de la pròpia aplicació on els contractes intel·ligents tenen un paper essencial.

La línia de treball del projecte serà analitzar els Smart Contracts e implementar-ne un utilitzant la plataforma de codi obert Ethereum. Per tal de poder implementar el Smart Contract serà necessari:

1. Per tal de documentar i explicar una mica millor el context dels Smart Contracts es realitzarà una introducció de conceptes com el funcionament descentralitzat de la xarxa, *Blockchain* i els *Blocs*.

2. Estudiar que és un Smart Contract, com funcionen, quins tipus hi ha i perquè s'utilitzen habitualment.

3. Descriure la plataforma Ethereum, com s'usa i quins són els passos a seguir per implementar un Smart Contract.

4. Estudiar el llenguatge solidity, necessari per implementar el Smart Contract en Ethereum.

5. Implementar un petit Smart Contract que simuli una acció de micromecenatge sobre la plataforma Ethereum.

6. Realitzar diferents simulacions del comportament del Smart Contract, per tal d'obtenir dades sobre el funcionament del les condicions definides en el Smart Contract.

1.1-Objectius i resultats esperats del treball.

Els principals objectius del projecte són el següents:

1. Conèixer i analitzar el funcionament dels Smart Contracts i els seus usos més habituals.

2. Conèixer i analitzar el funcionament de la plataforma de codi obert Ethereum.

3. Implementar un Smart Contract sobre Ethereum que poder realitzar una acció de micromecenatge.

4. Estudiar el comportament del Smart Contract.

1.2-Estudi de la situació actual.

Un contracte intel·ligent o Smart Contract és un programa informàtic que s'executa de forma autònoma i automàtica els termes del contracte definits en el codi del mateix.

Una propietat important dels contractes intel·ligents és que poden rebre i processar informació en forma de input necessària per l'execució del contracte sense la intervenció humana en el procés. També és important remarcar que les parts involucrades en un contracte intel·ligent poden ser tan persones com màquines, el que obre la porta al internet de les coses [21].

El concepte de contracte intel·ligent el va definir en 1997 el criptògraf Nick Szabo quan va començar a parlar de les propietats intel·ligents en el seu "The idea of Smart Contracts"[10]. En aquell moment no es va poder portar a la pràctica perquè no es disposava de la infraestructura tecnològica necessària. Per poder executar-los feia falta l'existència de les transaccions programables en un sistema financer que les reconegui, això és el que aporta la tecnologia de cadena de blocs (Blockchain) en el protocol de Bitcoin o més especialitzat en Ethereum.

La tecnologia blockchain permet l'existència d'actius digitals com diners, accions, registres i que es puguin controlar mitjançant el codi informàtic d'un contracte intel·ligent. La cadena de blocs aporta la seguretat informàtica suficient per permetre que no es pugui modificar un cop introduït el codi dintre de la cadena de blocs. Aquest punt també es veu assegurat pel caràcter distribuït de la blockchain on el consens informàtic necessari per modificar-la es converteix en confiança matemàtica que no requereix intermediaris humans o servidors centralitzats que poden ser objecte d'atacs informàtics.

Els diners encara que siguin en format digital estan sotmesos a una estricte regulació centralitzada normalment per bancs o entitats governamentals. Blockchain ha permès la creació de nous sistemes de pagament i monedes com Bitcoin o Ether que operen al marge d'aquesta regulació centralitzada.

1.3 Implementacions existents de Smart Contracts.

Els Smart Contracts o contractes intel·ligents no es limiten solament a automatitzar les funcionalitats dels contractes tradicionals, sinó que obren les portes a nous usos on es redueixen despeses i tracten d'oferir noves garanties del compliment del contracte. Alguns d'aquests nous usos que ja s'estan utilitzant actualment poden ser:

DRM

La gestió actual dels drets digitals DRM *"Digital rights management"* [1] que té com a finalitat limitar i controlar l'accés a un material o dispositius digitals, normalment de contingut audiovisual: musica, pel·lícules, etc.

Aquest sistema es pot considerar un Contracte intel·ligent entre el venedor i el comprador on es limita els dispositius on el comprador podrà accedir al material multimèdia producte de la transacció.

El que fa el software DRM és vincular l'arxiu multimèdia a un usuari, un hardware i sofware determinat. Així, si s'intenta accedir al contingut de l'arxiu multimèdia des d'un dispositiu no vinculat a l'arxiu el DRM impedeix el seu accés, per incompliment del contracte.

D'aquesta forma es manifesta una de les premisses dels Contractes intel·ligents, el DRM s'executa sense necessitar que intervinguin tercers per verificar que es compleixen les condicions del contracte. DRM directament fa una identificació del usuari i del hardware i permet o denega l'accés al contingut de l'arxiu audiovisual.

També es demostra una debilitat del DRM ja que s'ha aconseguit desenvolupar software que trenca aquest sistema i dona lliure accés al contingut audiovisual. En aquest cas el resultat no és que es incompleixen les condicions del Contracte intel·ligent sinó que s'ha trencat la seguretat del sistema DRM.

La tecnologia DRM està sent utilitzada per nombroses companyes de contingut digital com Sony, Microsoft, BBC, etc tot i que hi ha excepcions com Apple que ha renunciat l'ús de DRM en les cançons que ofereix en el seu portal iTunes.

DAO

Les Organitzacions Autònomes Descentralitzades DAO pot ser un altre exemple d'aplicació dels Contractes intel·ligents. Una DAO es pot comparar a una societat digital, però sense cap tipus d'entitat legal adscrita. L'entitat solament existeix com a codi informàtic sobre la cadena de blocs de Ethereum, està controlada pels creadors de la mateixa sense necessitat d'una direcció centralitzada.

Les DAO tenen una sèrie de regles formalitzades i automatitzades a través del software del seu contracte inicial que col·loca el poder de decisió en els membres participants de la DAO els quals poden presentar propostes a la comunitat i si són aprovades s'executaran de forma automàtica un cop afegides en el codi de l'entitat.

Les DAO actualment existents es financen usant el crowdfunding o finançament col·lectiu, són exemples la DAO Hub de Slock.it que ha recaptat 33,5 milions de dòlars, la DAO de Elio Motors que de moment ha recaptat 26 milions de dòlars, o la DAO de l'empresa de vídeo jocs Star Citizen que ha recaptat 18 milions de dòlars[11].

Productes financers

També s'estan començant a utilitzar contractes intel·ligents en productes o derivats financers [2] com ara:

- Herències. Es poden utilitzar per automatitzar l'assignació d'actius financers d'un compte bancari a un altre un cop confirmada la defunció d'una persona.

- Dipòsits de garantia. Els contactes intel·ligents es poden configurar com un dipòsit de garantia que realitzen el seguiment d'un intercanvi entre dues parts. Així el comprador d'un servei transferiria el pagament del mateix al compte del contracte. El contracte supervisaria els serveis (aplicacions d'una empresa de missatgeria per exemple) i un cop complert, el contracte alliberaria automàticament el pagament al venedor. Aquest sistema s'utilitza ja en la majoria de compres realitzades a través d'internet.

Intercanvi d'arxius en P2P

Una aplicació més dels Contractes intel·ligents podria ser l'intercanvi d'arxius en P2P [4] on el software sense necessitat que intervinguin tercers fa que quan un usuari decideix obtenir un contingut de la xarxa P2P com pot ser Kazan, Bot Torrent, eMule o Ares, passa a convertir-se en un node més de la xarxa i a compartir informació amb la resta de la xarxa P2P.

1.4-Planificació temporal.

El projecte està pensat per realitzar-se durant un semestre (de Setembre a Febrer). Segons els puts assenyalats en la introducció, el projecte es pot dividir en dues parts: anàlisi e implementació.

En la fase de anàlisi es realitzarà la recerca d'informació necessària sobre els Smart Contracts i la plataforma Ethereum i el seu llenguatge de programació solidity. Aquesta primera fase quedarà contemporitzada en les PAC1 (definició del pla de treball) i PAC2 (anàlisi dels Smart Contracts i de la plataforma Ethereum)

En la fase de implantació es realitzarà el disseny, les proves i verificacions del funcionament del Smart Contract en la plataforma Ethereum. Aquesta segona fase quedarà contemporitzada en les PAC3 (Implementació del Smart Contract) .

Per últim, la redacció i elaboració de la memòria incloent les conclusions està prevista que es realitzi durant l'execució del projecte, però temporalment estarà encabida en la PAC4 (memòria i producte resultant).

A la següent figura es mostra el diagrama de Gantt on es pot veure tota la planificació.

Id	Mo de tare	Nombre de tarea	Duración	Comienzo	Fin
1		**PAC1- Pla de treball**	**15 días**	**mié 21/09/**	**mar 11/10/**
2		Selecció proposta projecte	3 días	mié 21/09/16	vie 23/09/16
3		Recerca de documentació	1 día	lun 26/09/16	lun 26/09/16
4		Preparació esborrany	4 días	mar 27/09/1	vie 30/09/16
5		redacció Pla de treball	4 días	mié 05/10/1(	lun 10/10/16
6		**PAC2- Analisi Smart Contract i**	**20 días**	**mié 12/10/16**	**mar 08/11/16**
7		Recerca informació sobre Smart Contacts	8 días	mié 12/10/16	vie 21/10/16
8		Recerca informació sobre Etherum	7 días	lun 24/10/16	mar 01/11/16
9		Recerca informació Solidity	5 días	mié 02/11/16	mar 08/11/16
10		**PAC3- implementació Smart Contract**	**19 días**	**mié 09/11/16**	**lun 05/12/16**
11		implementació Smart Contacts	19 días	mié 09/11/16	lun 05/12/16
12		**PAC4- Memoria i producte resultant**	**21 días**	**mar 06/12/16**	**mar 03/01/17**
13		Prova funcionalitat Smart Contract	5 días	mar 06/12/16	lun 12/12/16
14		Revisió document definitiu	7 días	mar 13/12/16	mié 21/12/16
15		Conclusions	4 días	jue 22/12/16	mar 27/12/1
16		Preparació de la presentació	4 días	jue 29/12/16	mar 03/01/17

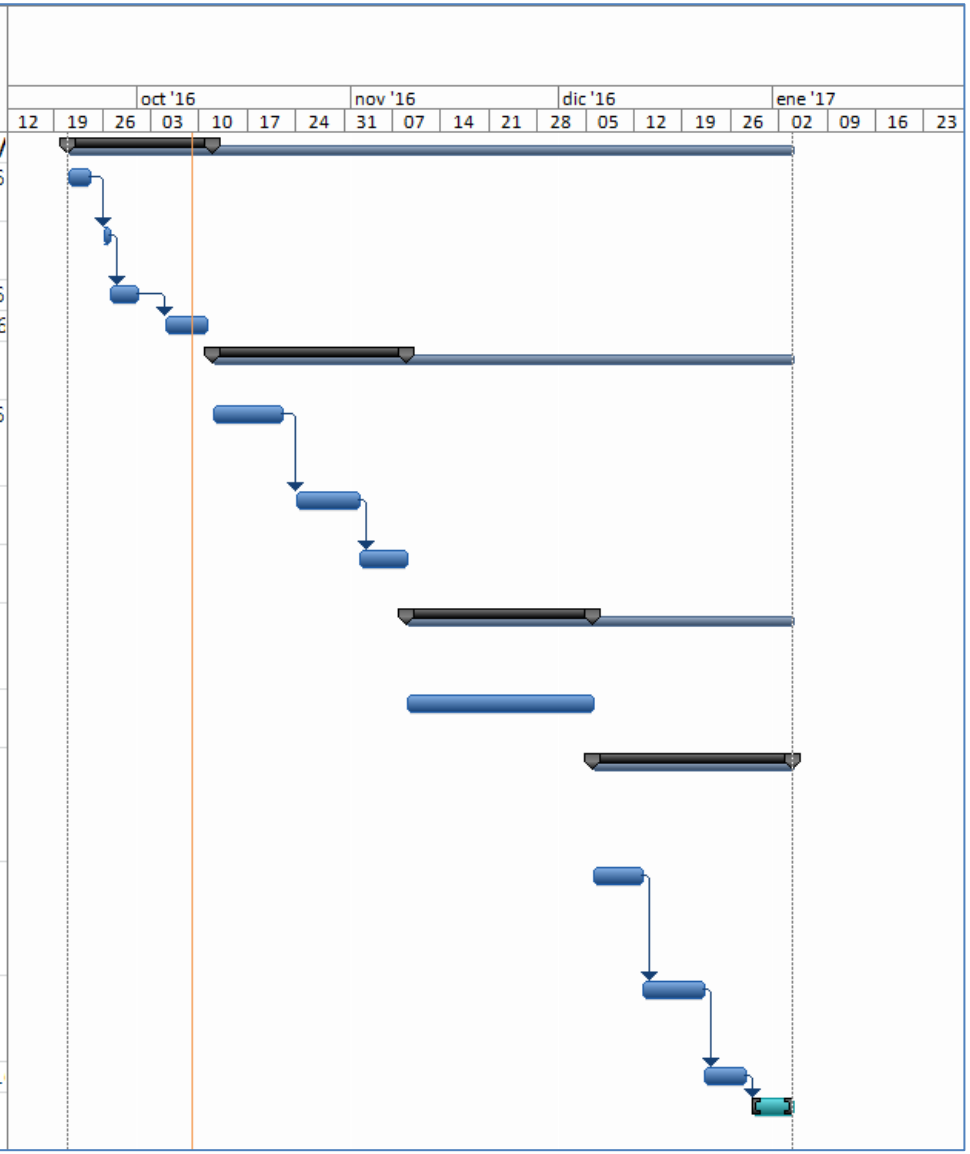

Figura 1 : Pla Temporal

1.5-Productes a lliurar.

Durant la realització del projecte i seguint el Pla docent de l'assignatura s'haurà d'entregar els documents corresponents a les proves d'avaluació continua, concretament en cada PAC s'ha d'entregar:

Data	PAC	Contingut
11/10/2016	PAC1	-Pla de treball. -Iniciar la taula de contingut de la memòria del projecte. -Introducció al projecte amb la definició dels termes principals. -Descriure els objectius del projecte.
08/11/2016	PAC2	-Descripció de que és un Smart Contract, com funcionen, quins tipus hi ha i perquè s'utilitzen habitualment. -Descriure la plataforma Ethereum, com s'usa i quins són els passos a seguir per implementar un Smart Contract.
05/12/2016	PAC3	-Implementació d'un Smart Contract. -Verificar la funcionalitat del Smart Contract. -Documentar els resultats del projecte.
03/01/2017	PAC4	-Memòria final del projecte.
11/01/2017	Presentació final	-Presentació virtual.

Taula 1 : Llista de productes a lliurar en el projecte.

1.6-Requeriments del projecte.

Per tal de poder implementar el contracte intel·ligent serà necessari instal·lar el moneder "wallet Mist" de Ethereum. Es pot descarregar un executable amb la instal·lació d'aquest moneder des de la mateixa pàgina web de Ethereum. Un cop instal·lat el moneder també serà necessari descarregar-se una rèplica de la cadena de blocs de Ethereum. En el nostre cas per simplificar utilitzarem la cadena de blocs de prova de Ethereum la "Private-net".

2-Bitcoin.

Com s'ha comentat en el capítol anterior els contractes intel·ligents són programes informàtics amb un seguit de funcions (clàusules d'un contracte) que s'apliquen segons si es produeixen una sèrie d'esdeveniments. Però aquests contractes intel·ligents necessiten d'un suport que els hi confereixi seguretat, invulnerabilitat i una capacitat de transmetre actius o dines entre diferents usuaris. Aquestes propietats els hi dona la cadena de blocs, per tant, en el següent capítol es realitzarà una introducció a la cadena de blocs de Bitcoin que actualment és la més àmpliament distribuïda.

Tot i que l'objectiu del treball es implementar un contracte intel·ligent en el sistema Ethereum, es considera adequat començar explicant de forma general la cadena de blocs del sistema Bitcoin (que de fet va ser la primera que va tenir una difusió global) per després veure quines modificacions s'han incorporat a la cadena de blocs de Ethereum i perquè la fa més idònia per suportar els codis dels contractes intel·ligents, com el que es pretén implementar en aquest treball.

Segons descriu Luke Anderson [5] blockchain o cadena de blocs és essencialment un llibre de registre on s'hi van anotant esdeveniments digitals anomenats transaccions. Aquest registre està distribuït o compartit per moltes parts diferents.

El registre solament es pot actualitzar si hi ha un consens de la majoria de les parts implicades i un cop s'introdueix una informació o un esdeveniment digital ja no es podrà esborrar mai més, i passarà a formar part de la cadena de blocs de manera indefinida.

Amb aquest sistema, una cadena de blocs conté un registre fiable i al mateix temps verificable de tots els esdeveniments que s'han produït durant la seva història. La cadena de blocs més important que existeix actualment és la de la criptomoneda Bitcoin i es calcula que emmagatzema més de 80Gb de dades referents a transaccions realitzades amb aquesta criptomoneda.

2.1- Cadena de blocs del sistema Bitcoin.

Per entendre una mica més el funcionament d'una cadena de blocs el més indicat serà fer una revisió al funcionament de Bitcoin.

Bitcoin [6] és una criptomoneda virtual, descentralitzada i distribuïda que no depèn de cap entitat financera, sinó que fa ús d'una xarxa P2P,on els usuaris són nodes que participen i ajuden a mantenir el sistema. En aquesta xarxa tots els nodes són iguals i donen lloc a un sistema distribuït (xarxa C, Figura 2) resistent a possibles atacs informàtics, fallades o falsificacions. Així, encara que un node falli, es podria fer arribar una rèplica de la cadena de blocs als altres nodes que estan connectats pers diferents vies alternatives.

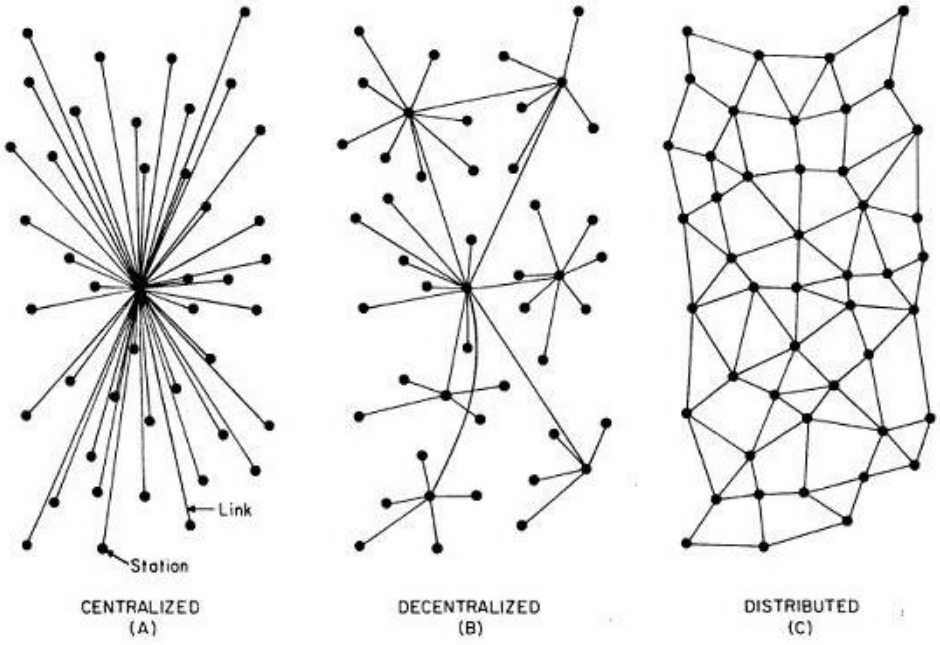

Figura 2 : Esquema de xarxes

El Bitcoin és un sistema de moneda digital basada en criptografia de clau pública i funcions hash. Concretament utilitza la criptografia de clau pública per a realitzar signatures digitals, i les funcions hash per garantir la integritat de la informació.

Un Bitcoin [7], també es pot definir com un apunt comptable en un compte corrent o un compte de bitcoins. Cada compte de bitcoins està identificat per una adreça que simplificant podríem dir que correspon a una clau pública, i té un seguit d'apunts comptables que determina l'import en bitcoins associat a aquest compte.

Un usuari pot generar un compte de bitcoins creant una parell de clau pública-privada. La clau publica determinarà l'adreça del seu compte de bitcoins, mentre que la clau privada li permetrà realitzar pagaments o transaccions des de aquest compte. Les transaccions de bitcoins s'efectuen publicant les transaccions en un registre públic que s'anomena blockchain o cadena de blocs.

De fet tot el sistema Bitcoin es podria veure com un sistema de transició d'estats, on l'estat inicial es defineix la propietat i quantitat de tots els Bitcoins existents en aquell moment, una funció que realitza la transició (transacció) cap a un nou estat i com a resultat es genera un nou estat on s'ha modificat la propietat i quantitat de Bitcoins de l'estat inicial.

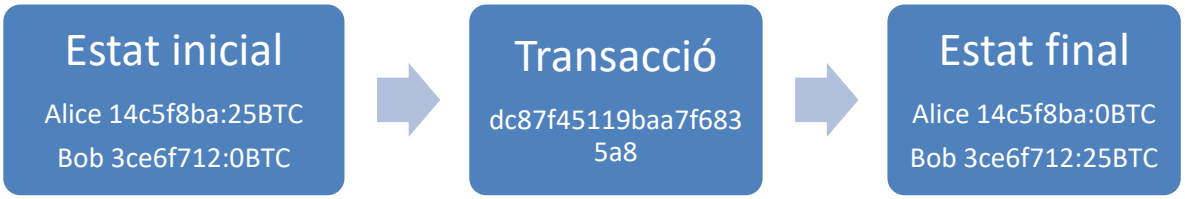

Figura 3 : Transacció com a transacció estats

Així per exemple simulem que Alice realitza un pagament de 25BTC a Bob. Per poder realitzar aquesta operació Alice i Bob han de tenir els seus corresponents comptes de Bitcoin amb les seves corresponents adreces. Bàsicament la informació que contindrà la funció de transacció haurà de incloure les dues adreces dels comptes de bitcoin implicats en l'operació, la quantitat de bitcoins que es vol transferir i tota aquesta informació haurà d'estar continguda en una signatura digital realitzada amb la clau privada de l'emissor, en aquest cas Alice.

Al rebre la transacció Bob haurà de realitzar tota una sèrie verificacions per aceptar la transacció:

1- Verificar que és realment Alice qui realitza la transacció mitjançant la clau pública d'Alice pot validar la signautra digital que ha realitzat Alice.

2- Verificar que no hi ha duplicitat en la utilització dels mateixos bitcoins. El sistema bitcoin registra totes les transaccions que s'ha realitzat desde l'inici, aquest registre és la cadena de blocs o BlockChain i serveix, entre altres, per assegurar que Alice no utilitza de nou aquests 25BTC o que ha realitzat el pagament amb uns bitcoins que no existeixen. Així Bob haurà de verificar la BlockChain de Bitcoin per assegurar-se que existeix una transacció anterior on s'ha traspasat els 25BTC al compte de l'Alice i que aquesta no ha utilitzat aquests 25BTC en una transacció anterior.

   Per tal que es pugui verificar aquest punt a la transacció s'haurà d'incloure un camp anomenat "*Input count*" amb el "*Previous Hash*", aquesta funció hash identifica la transacció anterior en la qual s'han transferit els 25BTC a l'adreça d'Alice.

3- Verificar la integritat de la blockchain de Bitcoin. El sistema BlockChain fa que cada bloc depengui del bloc anterior formant d'aquesta forma una cadena (d'aquí ve el nom de blockchain o cadena de blocs). D'aquesta manera modificar un bloc equival a incloure un nou bloc amb dades diferents a la cadena de blocs, però com que els blocs estan encadenats, la dificultat de modificar la cadena de blocs o una part d'aquesta és proporcional al nombre de blocs que es volen modificar. A més, la cadena de blocs està replicada en tots els nodes que formen el sistema Bitcoin convertint la tasca de modificació de la blockchain en una empresa realment dificultosa.

La xarxa Bitcoin crea un nou bloc aproximadament cada deu minuts, cada bloc conté un segell de temps o Timestamp i un hash doble SHA256 del bloc anterior a més d'una llista de totes les transaccions que s'han produït des del bloc anterior[12].

Degut a que SHA256 està dissenyat per ser una funció pseudoaleatòria la converteix en altament impredictible, la única forma de crear un bloc nou vàlid serà pel sistema d'assaig i error fins a trobar un hash correcte. Els usuaris miners necessiten aproximadament una mitjana de 2^{69} intents per trobar un bloc vàlid. Per tal de compensar als miners per aquest treball computacional per cada bloc nou se li concedeix 25BTC mitjançant una transacció especial de generació[13].

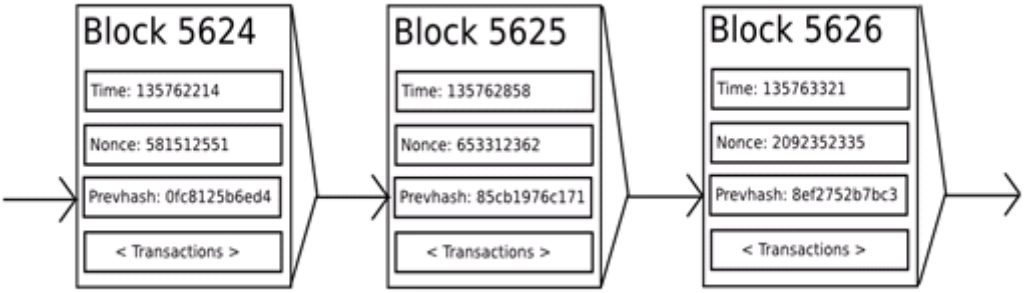

Figura 4 : Cadena de blocs

Una característica de l'escalabilitat de la blockchain és que els blocs s'emmagatzemen en una estructura de dades tipus arbre de múltiples nivells. El hash d'un bloc conté el timestamp del bloc, el nonce (nombre arbitrari per aconseguir el valor target del hash del bloc), el hash del bloc anterior i el hash arrel d'una estructura de dades anomenada arbre de Merkle que emmagatzema les transaccions del bloc[13].

L'objectiu dels arbres de Merkle és permetre que les dades d'un bloc puguin ser entregades per parts i descarregar solament les branques del arbre rellevants pel bloc en qüestió. Qualsevol intent de canviar una part de l'arbre de Merkle acabarà produint una inconsistència en la cadena de transaccions ja que els hash de les transaccions es propaguen cap a l'arrel de l'arbre.

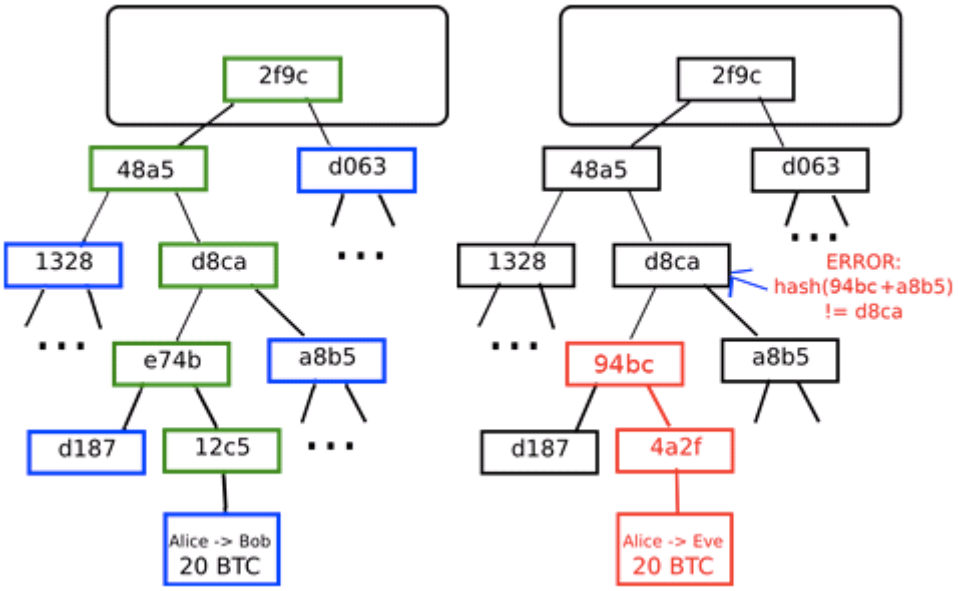

Figura 5 : Arbre de Merkle

Hi han diferents webs, API's que serveixen per consultar els moviments que es van produint dintre del sistema Bitcoin, una de elles és la https://blockchain.info/ . En aquesta web es pot consultar de forma online els moviments o transaccions que es va produint, els blocs que es van creant, de forma detallada.

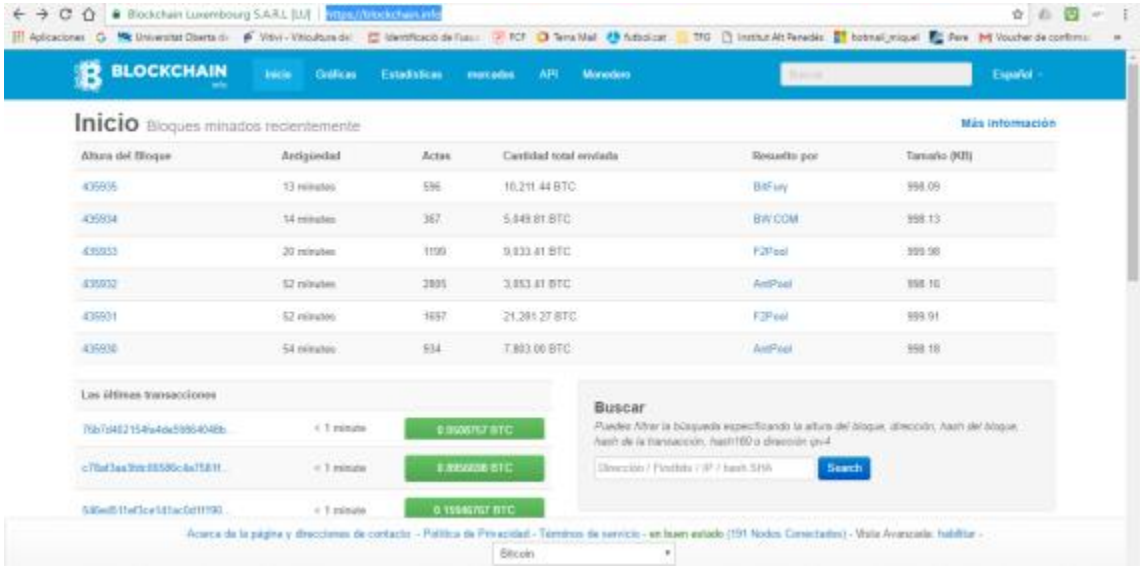

Figura 6 : Pàgina inicial de blockchain.info

Permet consultar el detall d'una transacció mostrant els camps descrits anteriorment:

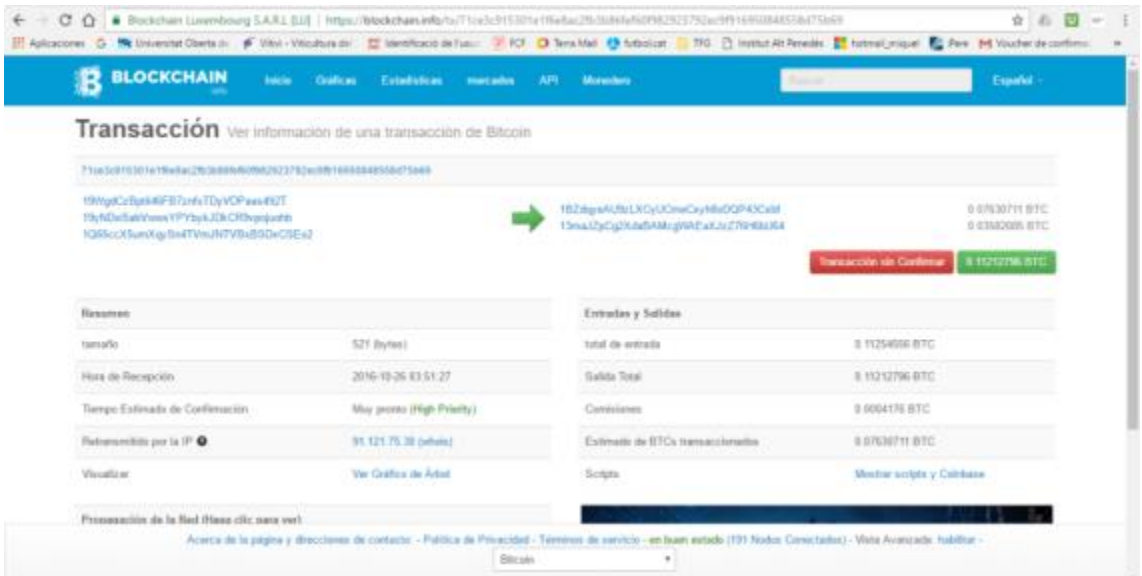

Figura 7 : Blockchain.info informació d'una transacció.

I també la informació detallada dels blocs que conformen la blockchain:

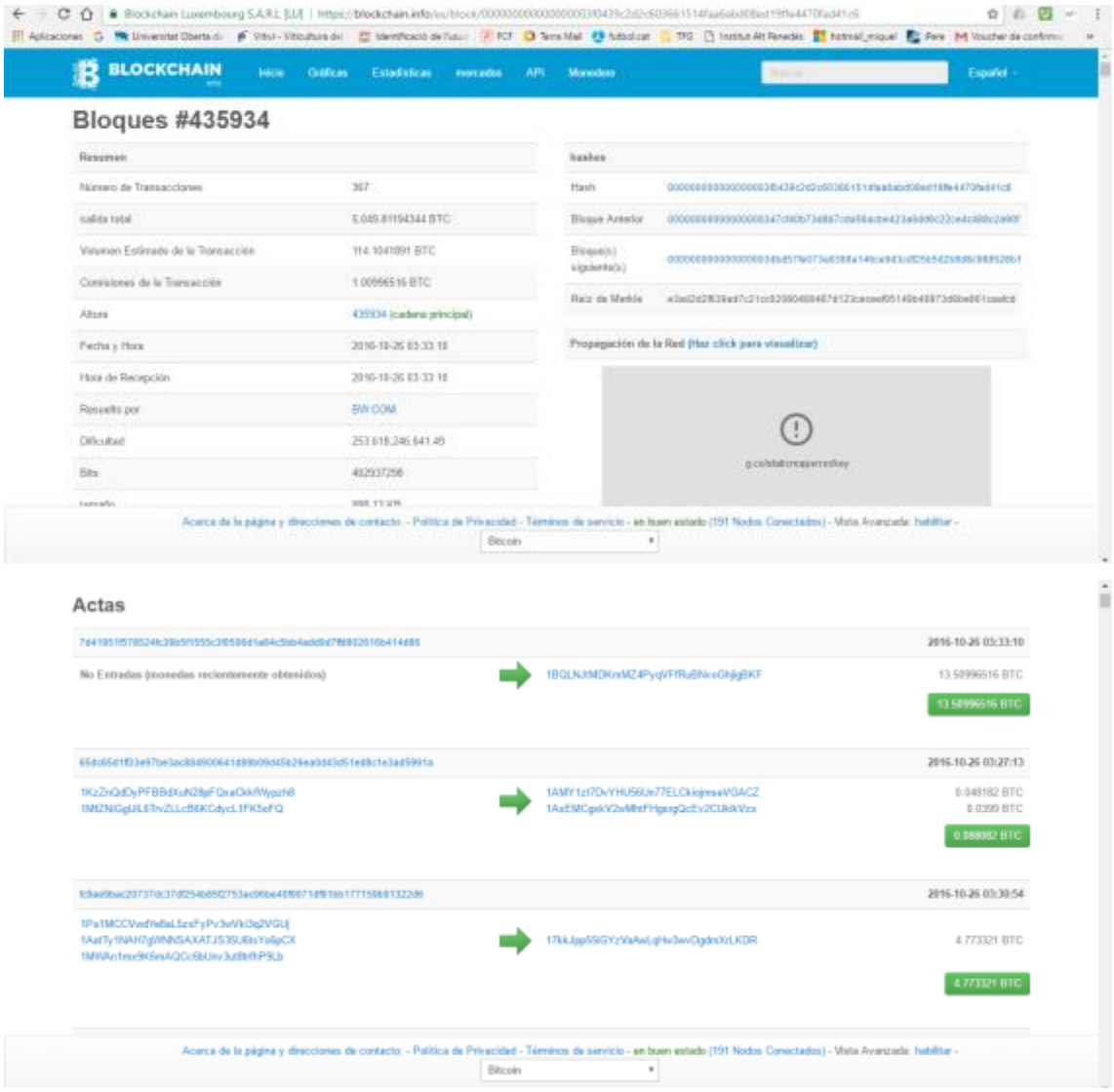

Figura 8 : Blockchain.info informació d'un bloc.

Un explorador del tipus blockchain.info posa de manifest una de les qualitats més important de Bitcoin i de la seva blockchain: la transparència ja que permet a qualsevol usuari consultar les dades de la cadena de blocs, fer estadístiques, etc.

2.2-Conclusió.

De l'anterior descripció del funcionament del sistema Bitcoin s'extreu que una Blockchain és pot considerar una base de dades distribuïda que conté una llista de registres de dades (transaccions) que està en continu creixement. Aquests registres o transaccions es van encadenant dintre d'un agrupament major anomenat bloc, un blocs alhora està lligat al bloc anterior i al següent mitjançant una funció hash. Això impedeix que es pugui modificar un bloc sense canviar la resta de blocs de la cadena.

D'aquesta forma s'aconsegueix que el sistema Bitcoin tingui un alt nivell de seguretat i transparència utilitzant una Blockchain, primer per la dificultat de modificar la mateixa Blockchain per la seva pròpia estructura d'encadenament de blocs i segon pel seu caràcter descentralitzat i distribuït que dificulta molt poder realitzar modificacions sense el consens d'una majoria dels nodes que conformen la xarxa.

Vist des de la perspectiva d'aquest treball es constata que la tecnologia blockchain o cadena de blocs és un suport adequat per poder desenvolupar els contractes intel·ligents, donat que proporciona un alt nivell de seguretat per poder desenvolupar contractes que implicaran transaccions de diners, encara que siguin criptomonedes com Bitcoin entre diferents usuaris.

El sistema Bitcoin però té l'inconvenient de no disposar d'una forma fàcil de desenvolupar els codis de contractes sobre la seva cadena de blocs. Bitcoin incorpora un llenguatge scripting que permet realitzar algunes accions, aquest llenguatge està basat en el llenguatge Forth que data dels anys 60. Forth és un llenguatge d'instruccions simples basat en pila i processat d'esquerra a dreta. Es considera un llenguatge Turing no complert, sense opció d'executar bucles cosa que limita les opcions dels contractes intel·ligents [16].

3-Ethereum.

Ethereum pretén crear un protocol alternatiu per construir aplicacions descentralitzades i contractes intel·ligents. El primer que fa és crear la seva pròpia cadena de blocs però li afegeix una capa abstracte més funcional que la que té el sistema Bitcoin. Això ho fa amb un llenguatge del tipus Turing-complert, cosa que no té Bitcoin. Amb aquest llenguatge Ethereum permet que qualsevol persona pugui escriure aplicacions descentralitzades, contractes intel·ligents etc, amb les seves pròpies funcions de transició d'una forma més senzilla i eficient [8].

3.1 Elements que composen Ethereum.

Els principals components de Ethereum són: comptes, transaccions i missatges.

Comptes.

Un estat de Ethereum està composat per comptes. Un compte consta d'una adreça i tot un seguit de transaccions que ha realitzat aquest compte. Les transaccions que inclouen els comptes poden ser transferències de Ether (és la criptomoneda de Ethereum), traspassos d'informació entre comptes o inclús entre contractes intel·ligents.

Un compte de Ethereum consta dels següents camps:

- Nonce, és un nombre que serveix de comptador per assegurar que cada transacció solament es pot processar una vegada.
- Un balanç actual de Ether que disposa el compte.
- Un codi de contracte del compte , si és que en té cap.
- El magatzem del compte, que està buit per defecte,

Hi ha dos tipus de comptes:

- Els comptes de propietat externes controlades mitjançant una clau privada i
- Comptes de contracte controlades per un codi de contracte.

Per comunicar i activar les aplicacions dels comptes es realitza mitjançant missatges. En els comptes de propietat externa aquests missatges solament es poden enviar mitjançant una transacció entre comptes, mentre que en els comptes de contractes es permet rebre i enviar missatges i prendre accions segons el contingut del missatge.

Transaccions.

Les transaccions en Ethereum són un paquet de dades signades digitalment i que contenen un missatge enviat des d'un compte de propietat externa. Els camps que conté una transacció són:

- L'adreça del receptor del missatge.
- Una signatura digital que identifica a l'emissor de la transacció.
- La quantitat de Ether a transferir des de l'emissor al receptor.
- Un camp de dades opcional.

- Un valor STARTGAS, que representa el nombre màxim de passos computacionals que es permet fer a la transacció.
- Un valor GASPRICE que representa la quota que ha de pagar l'emissor per cada pas computacional.

Els tres primers camps són els que es podrien esperar en una transacció: adreça del receptor, signatura digital per identificar a l'emissor i la quantitat de Ether que es transfereixen en aquesta transacció.

El camp de dades opcional dependrà del tipus de transacció que es faci, pot no incloure cap informació , si per exemple es realitza un pagament entre usuaris, o pot contenir informació rellevant per la transacció, per exemple si el receptor és un contacte que està funcionant com un registre de dominis, en aquest cas el camp de dades podria contenir el nom i l'adreça IP necessaris per registrar-lo.

Els camps STARTGAS i GASPRICE en Ethereum tenen la funció d'evitar atacs maliciosos a la blockchain. Així en cada transacció s'estableix un nombre finit de passos computacionals que pot realitzar definit a STARTGAS. A Ethereum la unitat fonamental de computació és el *gas* i una unitat de *gas* equival a un pas computacional, a més també s'estableix un cost de 5 unitats de *gas* per cada byte de dades de la transacció. Un cop calculat el cost de la transacció el valor s'inclou al camp GASPRICE.

La idea principal d'aquests dos camps és augmentar la seguretat del sistema, a més de la que ja proporciona la cadena de bloc. Consisteix en fer pagar pel consum que es realitza dels recursos de la xarxa de forma proporcional, obligant així a un possible atacant maliciós de la xarxa a pagar una quota de gas proporcional al consum dels recursos que utilitzi.

Missatges.

Els missatges s'envien entre comptes de contracte i solament existeixen en l'entorn d'execució de Ethereum. Els camps que conté un missatge són:

- L'emissor del missatge.
- El receptor del missatge.
- La quantitat de Ether a transferir juntament amb el missatge.
- Un camp de dades opcional.
- Un valor STARTGAS.

Pràcticament un missatge és igual que una transacció però els missatges són originats per un contracte i no per un actor extern al compte. El contracte emissor envia un missatge en el seu codi quan executa l'ordre CALL, llavors envia un missatge al contracte receptor que executarà el codi que tingui especificat al rebre un missatge determinat. D'aquesta forma els contractes poden interactuar entre ells enviant i reben missatges.

3.2- Transició d'estats en Ethereum.

Un exemple e Ethereum d'una funció de transició d'estats podria ser:

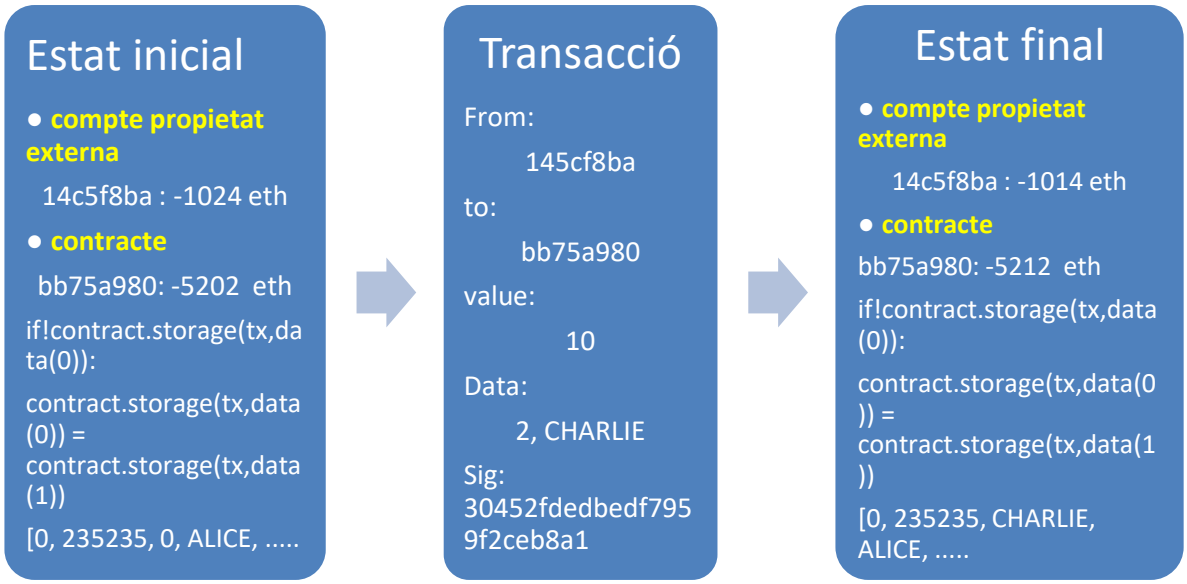

Figura 9 : Exemple transacció en Ethereum.

Els passos a seguir per validar una transacció de Ethereum són:

1. Verificar que la transacció està ben formada, la signatura digital és vàlida i verificar que el valor nonce coincideix amb el nonce de la compte de l'emissor.
2. Calcular la tarifa de la transacció STARTGAS i GASPRICE, verificar la direcció de la signatura digital. A continuació restar la tarifa GASPRICE del compte de l'emissor i augmentar el valor de nonce. Per finalitzar validar que el compte de l'emissor té suficient gas per pagar la transacció i en cas contrari retornar error.
3. Inicialitzar GAS= STARTGAS i separar la quantitat de gas a pagar per cada byte de la transacció (aquest gas serà pel miner que validi la transacció).
4. Transferir els Ether del camp valor del compte de l'emissor al compte del receptor. Si la compte destí és un contracte executar el codi del contracte .
5. Si la transferència falla perquè l'emissor no té suficient Ether o perquè l'execució del codi esgota el gas, es reverteixen tots els moviments a l'estat inicial excepte el pagament de la taxa al miner que ha realitzat la verificació.

3.3- Cadena de blocs de Ethereum.

La cadena de blocs de Ethereum és similar a la de Bitcoin, es diferencien en que els blocs de Ethereum contenen una còpia de la llista de transaccions, l'estat més recent i el nombre de blocs de la cadena de blocs. Segons aquesta visió es pot considerar a Bitcoin com stateless o que no es guarden els estats un cop realitzada un transacció versus Ethereum que es pot considerar stateful ja que guarda i té en compte els estats anteriors en la cadena de blocs. Una altre diferència important és que la cadena de blocs de Ethereum utilitza un algorisme de seguretat propi ETHash[1] en lloc del SAH-256 utilitzat per Bitcoin[9]. L'algorisme EThash encara està en evolució i aquest fet provoca el recel de nombrosos experts en seguretat per la seva utilització en una criptomoneda de la mida de Ethereum.

El procediment que s'ha de seguir per validar un bloc a Ethereum serà el següent:

1. Verificar que el bloc previ existeix i és vàlid.
2. Verificar que el Timestamp del bloc és major que el bloc referenciat prèviament.
3. Verificar diversos conceptes de baix nivell específics de Etherum com són el numero de bloc, la dificultat, la transacció arrel, el node tiet i que el límit de gas són vàlids.
4. Verificar que la prova de treball del bloc, hash, és vàlid.
5. Establir S[0] com estat inicial del bloc previ.
6. Establir TX com la llista de transaccions del bloc.
7. Establir S_FINAL com a últim estat S[n] però afegint la recompensa per al miner.
8. Verificar que l'arrel de l'arbre d'estats, S_FINAL, és igual a l'arrel de l'estat final proporcionada en la capçalera del bloc.

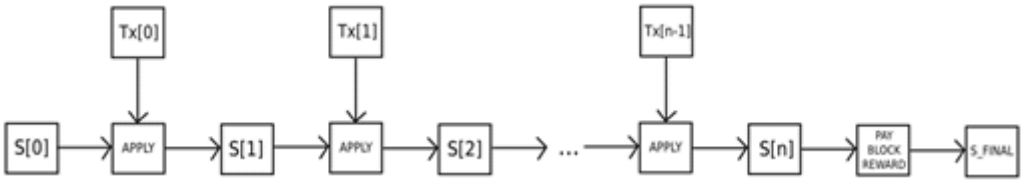

Figura 10 : Cadena de blocs de Ethereum.

S'incrementen les verificacions a realitzar per tal d'acceptar un nou bloc i a més s'emmagatzema l'últim estat en cada bloc. Però el fet de guardar l'estat en una estructura d'arbre provoca que entre dos blocs adjacents la major part de l'arbre sigui el mateix, això fa que les dades puguin ser guardades un sol cop i referenciades, per mitja d'un hash, dos cops [8].

Com s'ha explicat en el capítol anterior l'arbre de Merkel en el sistema Bitcoin emmagatzema les funcions hash de les transaccions que hi ha en el un bloc concret. Així si un usuari vol determinar l'estat d'una transacció pot demanar una prova de Merkel per verificar si la

[1] Ethash. Ethash és l'algoritme POW(prova de treball) previst per Etereum 1.0. És la versió més recent de DaggerHashimoto, introduït per Buterin [2013b] i Dryja [2014]

transacció està inclosa en una de les branques principals de l'arbre, l'arrel del qual està en la capçalera del bloc i així poder validar l'estat de la transacció.

Una limitació d'aquest sistema és que efectivament es pot provar la inclusió d'una transacció, però no es pot provar res sobre l'estat actual com podrien ser registres de noms, l'estat dels contractes financers, tendències d'actius digitals, etc. Aquest sistema de prova pot resultar poc eficient per aplicacions més complexes, donat que l'efecte d'una transacció pot dependre de l'efecte de varies transaccions anteriors que al mateix temps depenen d'altres transaccions anteriors, amb el que, en última instancia podria ser necessari autenticar tota la cadena de blocs.

Etherum porta el concepte d'arbre de Merkle un pas més enllà. Cada capçalera de bloc en Ethereum conté tres arbres per emmagatzemar tres tipus d'objectes.

- Transaccions.
- Receipts (són dades que demostren l'efecte de cada transacció)
- Estats

Amb aquest sistema es permet ampliar la capacitat de resposta a les consultes del usuaris. L'arbre que conté les transaccions permet contestar les consultes típiques sobre transaccions com per exemple si una transacció està inclosa en un bloc en particular. L'arbre de receipts permet contestar consultes com ara conèixer tots els contractes de crowdfunding que han aconseguit el seu objectiu d'una adreça concreta en els últims trenta dies. L'arbre d'estats permet realitzar verificacions sobre l'existència i saldo actual d'un compte o quin serà el resultat d'executar una transacció d'un contracte.

L'arbre d'estat serà doncs més complex. Un estat a Ethereum consisteix essencialment en una mapa clau-valor, on les claus són adreces (identificadors de 160 bits) i els valors són declaracions de comptes, llistat de saldos, el nonce, el codi i el magatzem de cada compte.

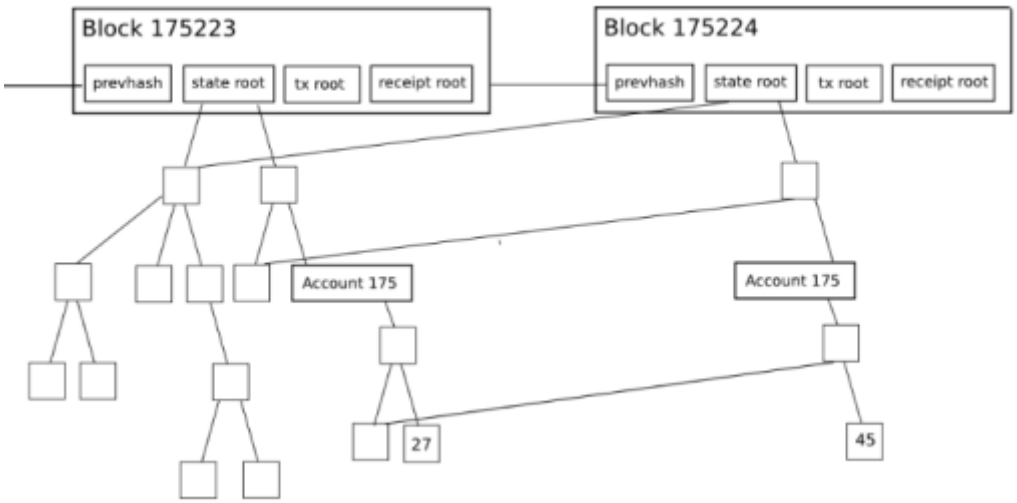

Figura 11 : Capçalera d'un bloc a Ethereum.

Els arbres de Merkel binaris són estructures de dades bones per autentificar informació que està en un format de llista i poden ser adequats per emmagatzemar les transaccions donat que un cop s'ha creat l'arbre ja no s'altera, solament es van afegint transaccions a l'arbre sense modificar la resta de l'estructura.

A diferència de l'historial de les transaccions, un estat s'ha d'actualitzar freqüentment donat que els saldos dels comptes varia molt i tot sovint es creen nous comptes que modifiquen l'estat del sistema. Això fa necessari una estructura de dades que permeti calcular ràpidament l'arrel de l'arbre un cop que s'ha inserit, actualitzat o eliminat un node sense tenir de tornar a calcular tot l'arbre[22].

La estructura també haurà de tenir en compte dos propietats més:

- Que l'alçada de l'arbre sigui limitada per evitar atacs de negació de servei.
- L'arrel d'un arbre depèn solament de les dades i no de l'ordre en que es realitzen les actualitzacions. Fer actualitzacions en un ordre diferent o inclús tornar a calcular l'arbre des de zero o ha de suposar un canvi de l'arrel.

Els desenvolupadors de Ethereum han trobat que l'arbre de Patricia és l'estructura més adequada per donar resposta als requeriments anteriors. Els arbres Patricia són totalment deterministes, el que significa que un arbre Patricia amb els mateixos enllaços clau-valor es garanteix que és exactament el mateix fins l'últim byte i en conseqüència té el mateix hash arrel [14].

Els arbres Patricia són arbres digitals, la principal diferencia amb els tries és la eliminació dels nodes unaris (amb un sol fill), els nodes que romanen en l'arbre són aquells que tenen una quantitat de fills iguals o majors a dos; la cerca o recorregut de l'arbre es continuarà realitzat, però ara per indexació.

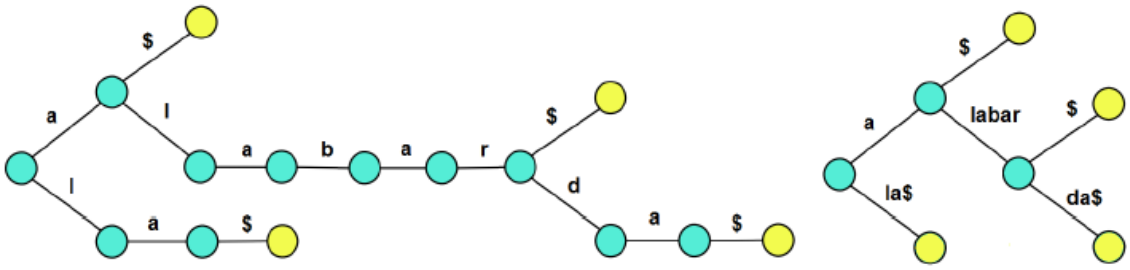

Figura 12 : Exemple simplificació del camí a recorre en una arbre Patricia (dreta).

La idea és que el cas que hi hagi un llarg camí de nodes amb un sol fill, s'escurça el camí amb la creació d'un node clau-valor, on la clau dona el camí hexadecimal a seguir en la codificació compacta de l'arbre i el valor és el hash d'aquest nou node. D'aquesta forma l'espai que necessita l'arbre d'estat es simplifica de forma substancial [15].

3.4- Conclusió.

Ethereum amplia les funcionalitats del sistema Bitcoin i de la cadena de blocs descentralitzada com ara jocs d'atzar, contractes financers, magatzem de dades entre altres. Aquestes noves funcionalitats no les suporta directament la cadena de blocs de Ethereum , sinó que gràcies a l'existència d'un llenguatge Turing-complert (apte per programar qualsevol cosa) com solidity permet que es creïn aquests nous tipus de transaccions i contractes.

Ethereum ha rebut moltes crítiques per potencials problemes de seguretat, la majoria centrats en el fet que el seu software és molt nou, pràcticament està disponible des de solament fa un any i mig i això implica que la xarxa Ethereum hagi estat sotmesa a una quantitat de proves en front de possibles atacs molt inferior en comparació a la xarxa Bitcoin. Aquesta immaduresa del sofware també provoca que executar els contractes intel·ligents sigui car en quant als Ether que consumeixen i limitat en la seva potencia per executar operacions (tot i que supera e escreix a la de Bitcoin). Segons els desenvolupadors de Ethereum el sistema en aquests moments és igual de potent que els telèfons mòbils dels anys 90, per tant el potencial de millora és molt gran i és necessari donar més temps a projecte per veure el potencial real de la criptomoneda.

En aquest capítol s'ha pogut veure les principals diferències que hi ha entre la cadena de blocs de Bitcoin i la de Ethereum. Però potser el punt principal a destacar, en referència a l'objectiu d'aquest treball, és que Ethereum disposa d'una capa abstracte molt més funcional que permet codificar codis complexos amb un llenguatge Turing complert anomenat Solidity. Solidity és un llenguatge d'alt nivell molt similar a JavaScript que es pot codificar i compilar en una Màquina Virtual de Ethereum.

En el següent capítol analitzarem amb més profunditat el llenguatge Solidity per assolir una base suficient que ens permeti desenvolupar un contracte intel·ligent de micromecenatge sobre la plataforma Ethereum.

4-Solidity

Solidity és un llenguatge de programació orientat a objectes molt similar al JavaScript que és utilitzat per a la redacció de contractes intel·ligents. [18] S'utilitza per posar en pràctica els contractes intel·ligents. Va ser desenvolupat per Gavin Wood, Cristiano Reitweissner, i diversos col·laboradors de l'antic nucli Ethereum per permetre l'escriptura contractes intel·ligents en plataformes blockchain com Ethereum.

En la elaboració dels codi dels contractes intel·ligents que desenvoluparem en aquest treball utilitzarem els següents termes i tipus característics del llenguatge Solidity[17] que descrivim a continuació:

4.1-Objectes elementals de Solidity

En aquest apartat es veuran els objectes elementals, així com les estructures algorísmiques, accions i funcions principals usades en aquest treball del llenguatge Solidity.

4.1.1-Tipus elementals de solidity.

Els tipus elementals s'apliquen actualment són booleans (bool), enters i matriu de longitud fixa cadena / octet (des de 0 bytes a 32 bytes) .

Els tipus enter són amb signe (int) o sense signe (uint) de diversos amples de bit (des de int8 / uint8 a int256 / uint256 en passos de 8 bits).

El tipus "addres" o adreça, tenen un valor de fins a 160 bits, que és la mida d'una adreça de Ethereum, que pot ser d'un wallet, d'un compte o d'un contracte. És possible consultar el saldo d'una adreça mitjançant la propietat *"balance"* i enviar Èter (en unitats de wei) a una adreça utilitzant la funció *"send"* com el següent exemple:

```
address x = 0x123;
if (x.balance < 10 && address(this).balance >= 10) x.send(10);
```

Les comparacions (<=,! =, ==, etc.) sempre donen com a resultat booleans (cert o fals) que es poden combinar usant &&, || i !.

4.1.2-Tipus estructurats de dades.

Els tipus estructurats de dades: vectors, tuples, tipus enumeratius són vàlids a Solidity i segueixen les mateixes pautes d'ús que els llenguatges Java i JavaScript.

Exemple declaració vector:

```
uint[] x;
```

Exemple declaració tupla:

```
struct Funder {
                address addr;
                uint amount;
}
```

El tipus estructurat "mapping" emmagatzema parelles de claus / valor on el valor pot ser una adreça, un valor o una tupla.

Exemple declaració mapping:

```
apping (uint => Funder) funders;
```

4.1.3-Estructures de control.

La major part de les estructures de control de C / JavaScript estan disponibles a Solidity a excepció switch i Goto.[19]

4.1.4-Modificadors de funció (Modifiers).

Els modifier són funcions que verifiquen automàticament una condició abans de realitzar una mètode o funció.

Exemple declaració modifier:

```
modifier afterDeadline() { if (now >= deadline) _; }
```

Aquesta funció verificarà si s'ha superat el temps especificat a deadLine.

```
function checkGoalReached() afterDeadline {
```

Posant-lo en la declaració d'una funció indica que solament es pot aplicar quan es compleix la condició que s'ha definit en el modifier, en aquest cas *afterDeadLine*.

El modifier `payable` permet rebre Ether juntament amb una trucada o transacció.

4.1.5-Events.

Els "*events*" permeten interactuar amb la cadena de blocs de Ethereum. Un *event* escriu la seva acció en el registre de la transacció de la cadena de blocs. D'aquesta forma la informació dels arguments del *event* queda ja fixada e immodificable.

Exemple declaració d'un event:

```
event FundTransfer(address backer, uint amount, bool isContribution);
```

En aquest exemple quan es cridi l'event FundTransfer es registrarà a la transacció de la seva implementació l'adreça d'un compte (backer) a on es depositarà una quantitat determinada (amount) i per últim un booleà (isContribution) que indica si la transferència de fons és una contribució o no.

4.2-Desplegament de contractes en el wallet de Ethereum.

El procediment per crear aquest contracte a Ethereum és el següent:

- Escrivim el text del contracte en un editor de text com ara Notepad++ (també si es vol es pot escriure directament el codi en la finestra "SOLIDITY CONTRACT SOURCE CODE" del apartat de contractes del wallet de Ethereum.
- Obrim el wallet per l'apartat contractes i premem la tecla deploy new contract

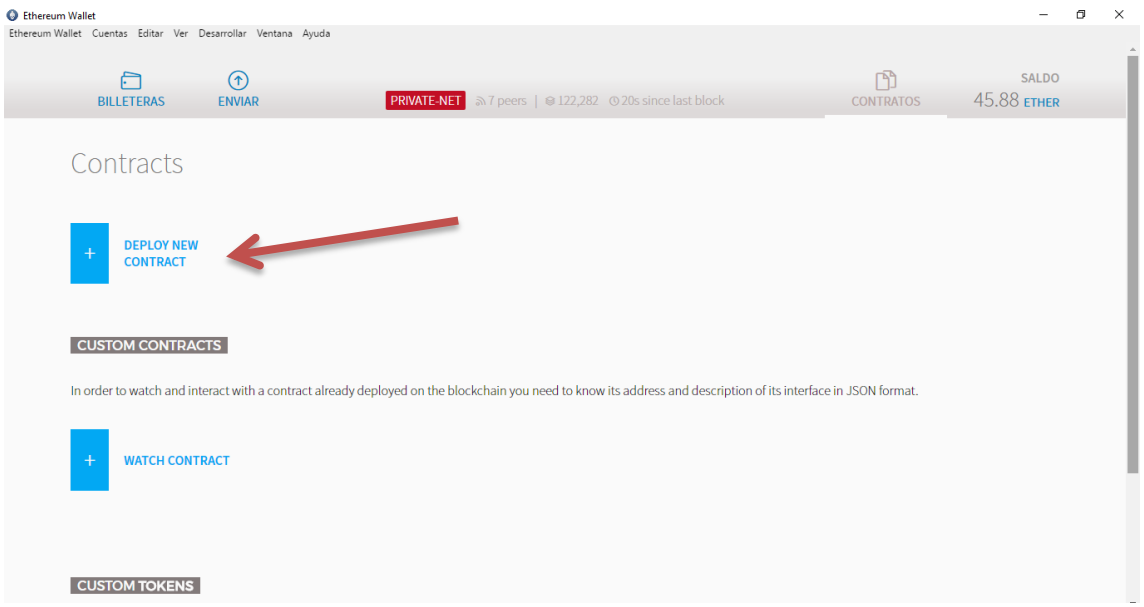

Figura 13 : Secció de contractes de la wallet de Ethereum.

- Enganxem el nostre codi en la finestra “SOLIDITY CONTRACT SOURCE CODE”

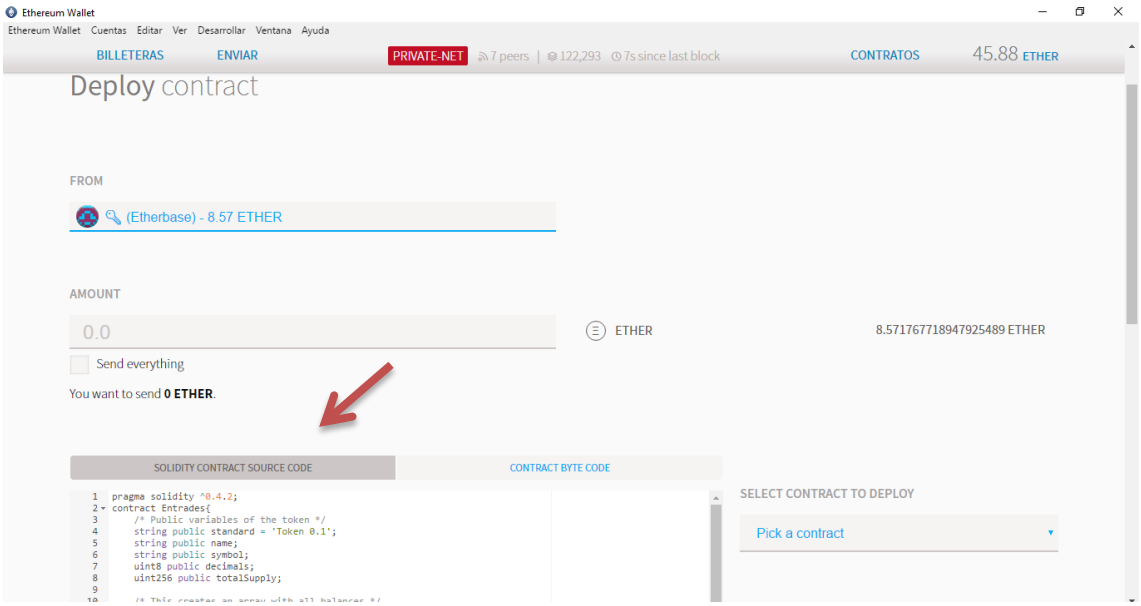

Figura 14: Imatge finestra SOLIDITY CONTRACT SOURCE CODE.

El wallet Mist de Ethereum inclou un compilador *Solidity Compiler* (Solc) to release 0.4.2 que indicarà si és necessari, els possibles errors que pugui tenir el nostre codi.

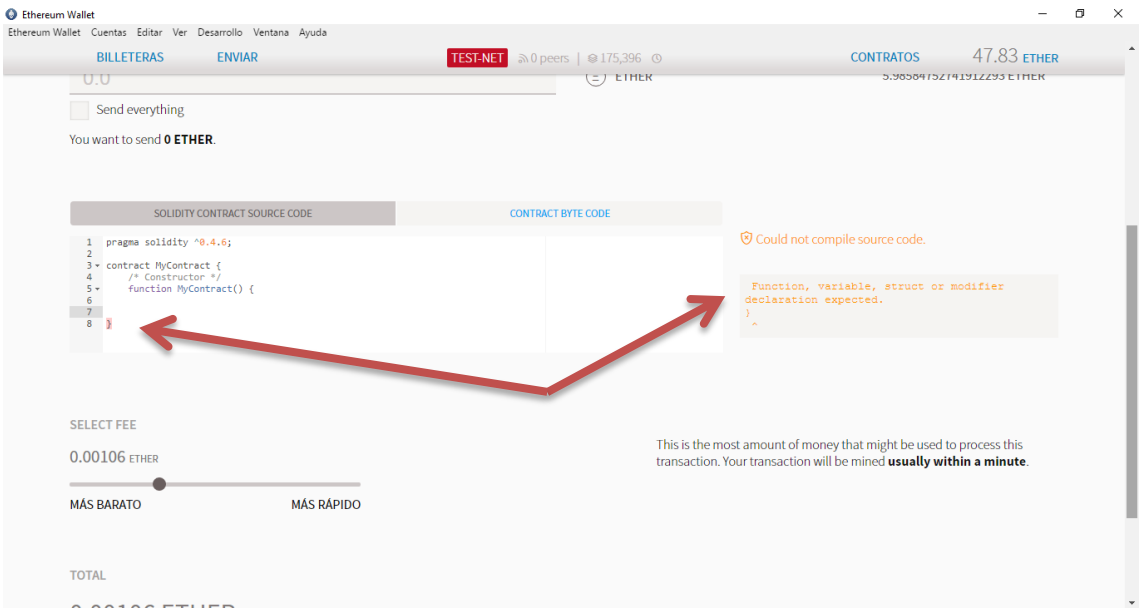

Figura 15 : Compilador del wallet de Ethereum

5- Disseny dels contractes intel·ligents.

En aquest capítol es procedirà a realitzarà una introducció al contracte intel·ligent que es vol desenvolupar, l'escenari que es proposa, els objectius que es volen assolir i els diferents casos d'ús que podrà tenir el contracte intel·ligent.

5.1-Introducció i objectius.

Amb la finalitat de verificar el funcionament d'un contracte intel·ligent sobre Ethereum és planifica un escenari simulat de la vida real on es podria aplicar la teoria que s'ha explicat en els capítols anteriors.

Concretament plantejarem la implementació d'un contracte intel·ligent de micromecenatge sobre la plataforma Ethereum. En aquest contracte es vol desenvolupar els principals casos d'ús o funcionalitats que es suposa ha de tenir un projecte de micromecenatge com són: , fer una contribució al projecte, donar una recompensa als mecenes que contribueixen en el projecte, transferir els fons al beneficiari un cop acabat el projecte o retornar les donacions als mecenes si no s'ha aconsegueix un objectiu establert.

Tots aquests casos d'ús es traslladaran al contracte intel·ligent en forma de funcions del codi del contracte. Aquest codi s'haurà d'inserir a la cadena de blocs de Ethereum per tal de aconseguir les virtuts de la tecnologia de cadena de blocs: executar-se de forma autònoma i no dependre de terceres parts, tenir caràcter distribuït o compartit amb tots els nodes de la xarxa, ser visible per a tothom que el vulgui consultar, ser inalterable i segur.

Per traslladar el nostre contracte intel·ligent a la cadena de blocs de Etherum escriurem el codi en el llenguatge d'alt nivell de la plataforma (Solidity), i utilitzarem el wallet Mist que és el recomanat per la plataforma Ethereum. Per últim podrem consultar que les transaccions fetes s'han introduït a la cadena de blocs de Etherum amb un explorador d'aquesta cadena com pot ser EtherScan.

Un cop implementat el contracte realitzarem diverses proves per verificar que el funcionament sigui l'esperat i que el contracte realitza les funcions descrites en el seu codi de forma autònoma.

5.2-Escenari.

Per tal de mostrar un exemple d'ús dels contractes intel·ligents es procedeix a definir un escenari el més real possible on la utilització d'un contracte intel·ligent pot ser útil per aconseguir els objectius marcats.

Així doncs suposem que una companyia de teatre amateur anomenada "El Folre" està molt interessada en poder representar una versió actual del "Cyrano de Bergerac", però degut al seu caràcter amateur necessiten algun tipus de finançament per poder afrontar les despeses de vestuari i atretzzo que requereix aquest tipus de projecte.

Finalment els membres de la companyia decideixen iniciar una campanya de micromecenatge per tal d'aconseguir el capital necessari per finançar el nou projecte. Aquesta campanya la volen realitzar sobre una plataforma descentralitzada com ara Ethereum donat que es pot implementar d'una forma senzilla sense excessives depeses amb un alt grau de seguretat.

Per tal de simplificar una mica l'escenari, en aquest TFG solament utilitzarem un wallet pel promotor del contracte intel·ligent i els mecenes que aportaran petites quantitats de diners per poder finançar el projecte. En un cas real tan el promotor com els mecenes haurien de disposar del seu corresponent wallet per rebre i poder realitzar les donacions al projecte.

El fet de treballar amb un sol wallet no té una incidència important en la implementació del contracte ja que procedirem a crear diferents comptes dintre el wallet amb noms de possibles mecenes. Al crear un nou compte al wallet de Ethereum assigna directament una adreça igual que tindria si el compte estigues en un wallet diferent. Per tant l'exemple que es dissenyarà serà molt real. També es procedirà a crear un compte pel grup de teatre "El Folre" que serà el beneficiari del projecte de micromecenatge.

5.3-Casos d'ús.

A continuació és defineixen les funcionalitats o casos d'ús que el contracte intel·ligent a implementar hauria de satisfer i els actors que les realitzen:

- Cas d'ús – Creació dels contractes intel·ligents al wallet de Ethereum.
  Aquesta funcionalitat la portaria a terme un promotor del projecte. En el nostre escenari el promotor és un actor diferent al beneficiari, però no hi hauria cap problema si fossin la mateixa persona o actor.
  El fet de no ser el mateix actor és rellevant perquè a Ethereum el compte que porta a terme una transacció es fa càrrec del cost en GAS de la transacció.

- Cas d'ús – Proveir d'entrades de regal al contracte Mecenatge.
  Tal i com es veurà en la implementació s'hauran de realitzar dos contractes intel·ligents. En un primer anomenat Entrades és crearà la figura de les entrades que es donaran com a recompensa als mecenes que col·laborin en el projecte.
  Serà necessari posteriorment transferir el nombre d'entrades que sigui necessari segons les aportacions esperades al segon contracte anomenat Mecenatge, que serà el contracte on es rebran les donacions, per tal de poder recompensar als mecenes del projecte amb una quantitat d'entrades proporcional a la seva donació.

- Cas d'ús - Realitzar donacions al projecte.
  Aquesta funcionalitat la realitza l'actor anomenat mecenes i consisteix com indica la seva definició en realitzar una contribució econòmica al projecte.

- Cas d'ús – Entregar entrades de regal.
  Aquesta funcionalitat la té de realitzar automàticament el contracte intel·ligent. Bàsicament consistirà en verificar si a la quantitat de diners que ha a aportat al mecenes li correspon una recompensa o una entrada de regal.
  Prèviament en el contracte Mecenatge s'haurà definit la quantitat mínima de contribució a la que li correspon una entrada de regal.

- Cas d'ús - Finalitzar contracte mecenatge.
  Un cop superat el límit de temps, establert en el contracte en el que els mecenes poden fer col·laboracions al projecte, el promotor podrà activar la clàusula de finalització del contracte.
  Aquesta clàusula ha de realitzar de forma automàtica un dels dos següents casos d'ús i solament un.

- Cas d'ús – Transferir els fons al compte del beneficiari.
  Aquesta funcionalitat l'ha de realitzar de forma automàtica el contracte. Un cop el promotor ha activat la clàusula de finalització, el codi del contracte ha de verificar si la quantitat recaptada en el projecte és major o igual a l'objectiu a recaptar establert en el contracte, es cas afirmatiu, el contracte transferida automàticament els fons recaptats al compte del beneficiari.

- Cas d'ús – Retornar les aportacions al mecenes.
  Aquesta funcionalitat també la realitza de forma automàtica el contracte si un cop activada la clàusula de finalització, no s'ha assolit el objectiu de recaptació de fons establert en el contracte.
  En aquest cas el contracte ha de retornar automàticament a cada mecenes que hagi col·laborat l'import amb el que ha contribuït en el projecte. Per tal de poder satisfer aquesta funcionalitat, el contracte haurà de tenir la capacitat de poder emmagatzemar totes les contribucions fetes associades cadascuna a l'adreça del mecenes que l'ha realitzada per tenir la informació necessària per portar a terme aquest cas d'ús.

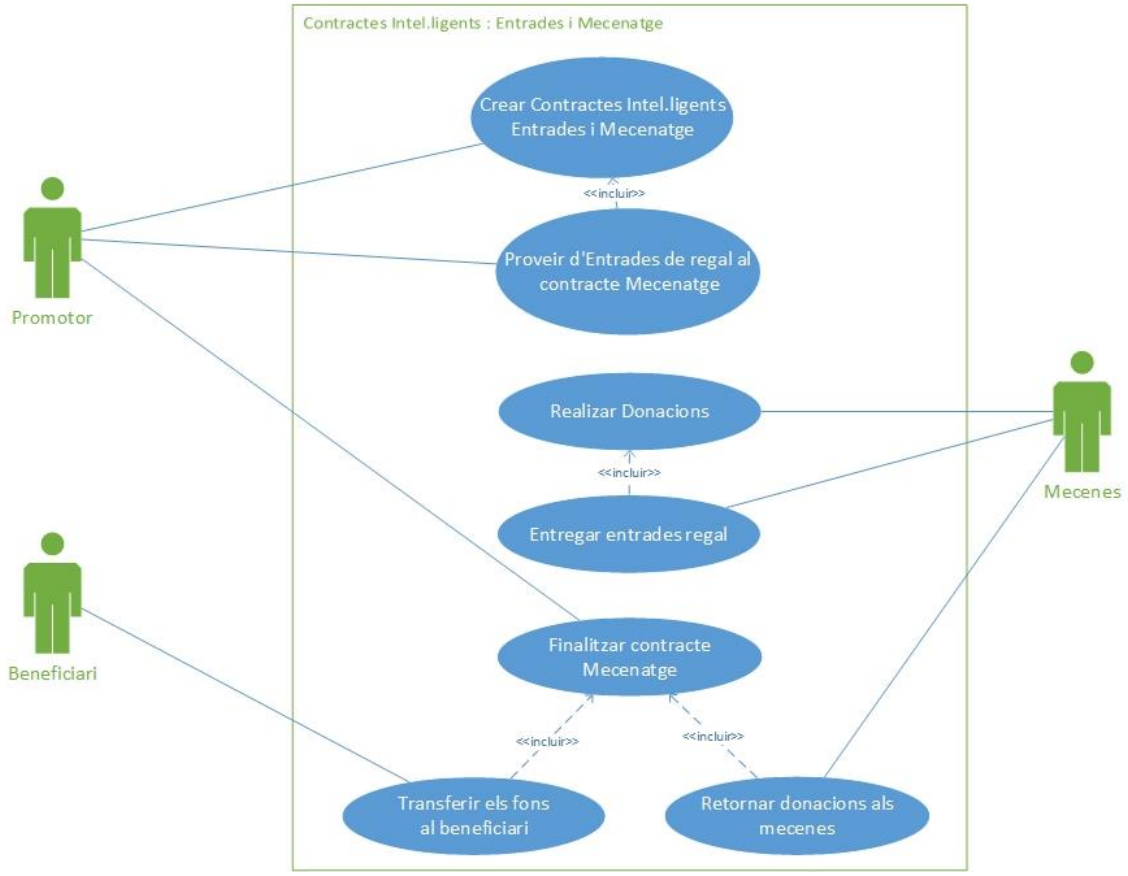

Figura 16 : Diagrama de Casos d'ús

6- Implementació dels contractes:

En aquest capítol es descriuran els passos que s'han seguit per realitzar la implementació dels contractes intel·ligents descrits en el capítol anterior.

6.1-Preparació de l'entorn.

La implementació del contracte intel·ligent es realitzarà utilitzant el wallet Mist de Ethereum. Es pot descarregar directament l'executable des de la web https://www.ethereum.org/ . Per simplificar la baixada i la implementació realitzarem les proves sobre la Private-net que és una xarxa de proves que ofereix Ethereum:

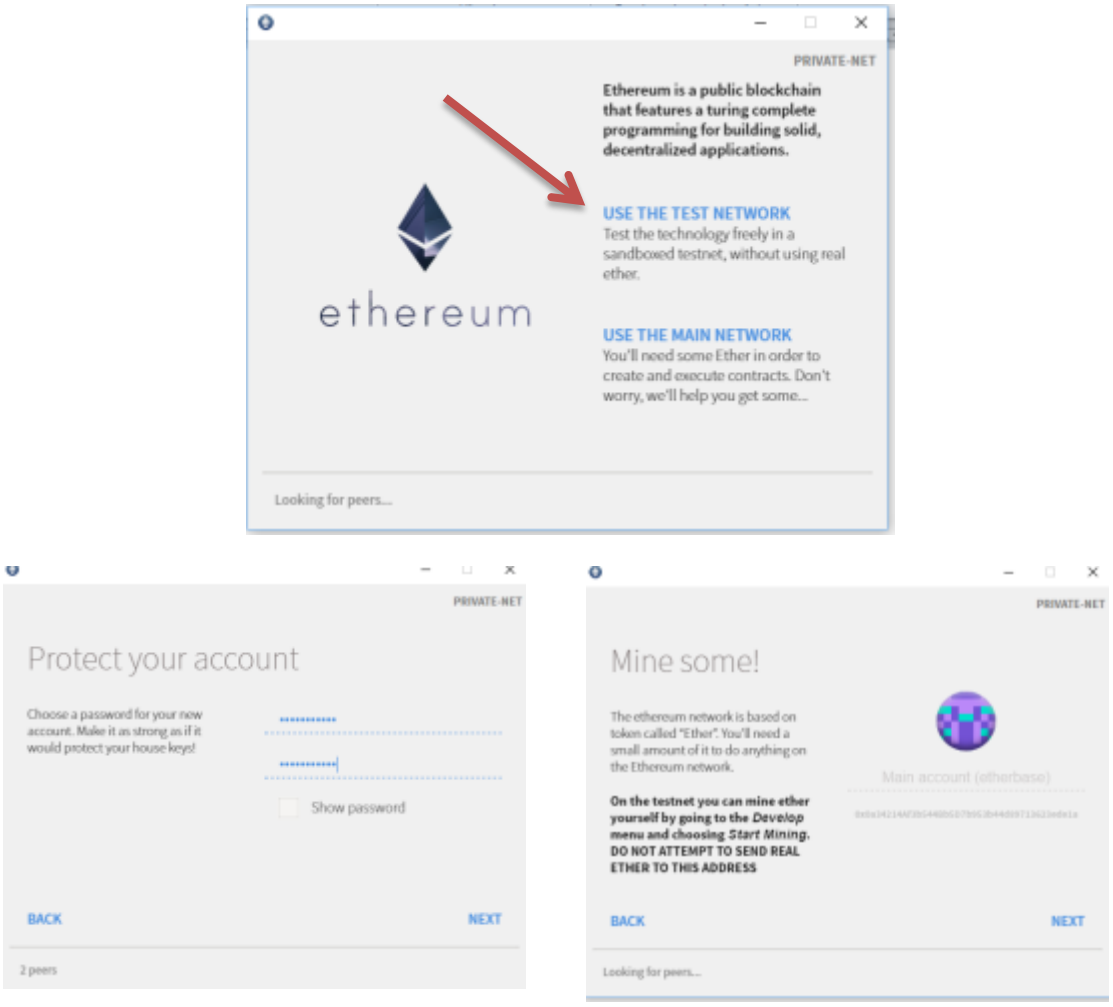

Figura 17 : Instal·lació del Wallet de Ethereum

El que s'ha de fer a continuació és descarregar una rèplica de la xarxa de prova de Ethereum en el nostre ordinador. Aquesta baixada por resultar força lenta depenen de la velocitat de descàrrega que disposi la nostra connexió d'internet. En el nostre cas va trigar 12 hores en descarregar-se els 100.000 blocs que composen la xarxa de proves PRIVATE-NET.

Un cop preparat l'escenari l'aspecte del nostre wallet serà el següent.

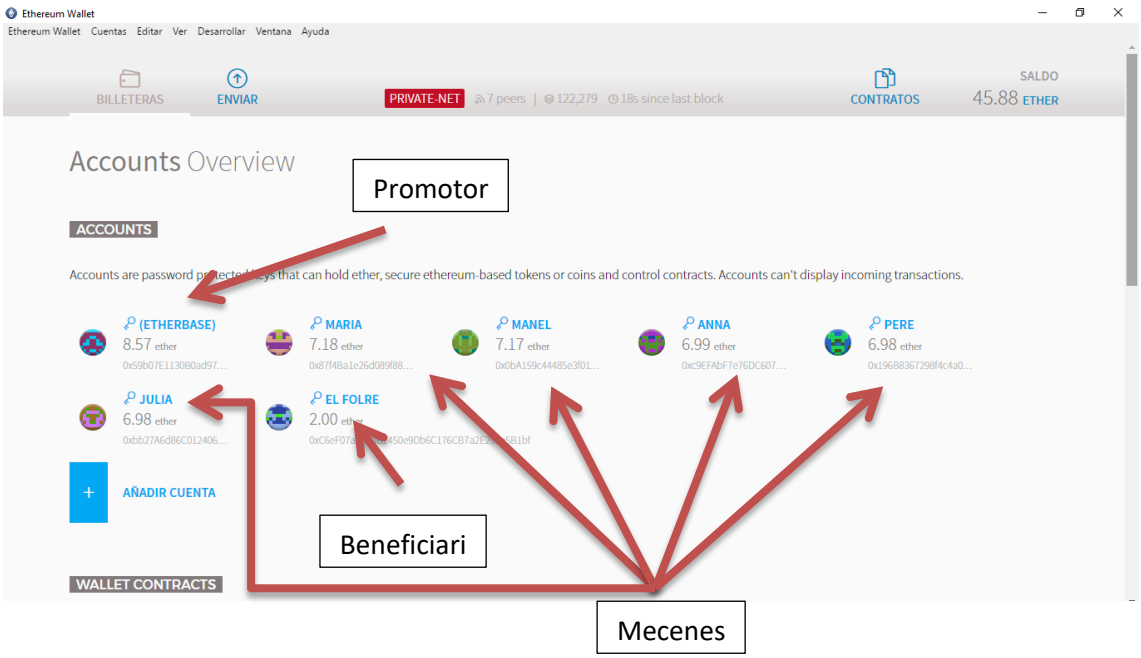

Figura 18 : Situació inicial del wallet.

En aquesta figura s'observa com dins el wallet s'han creat diferents comptes cadascuna amb la seva corresponent adreça Ethereum. Així el compte base del wallet l'utilitzarem i assignarem al promotor del projecte, el compte anomenat el Folre correspondrà al beneficiari i la resta de comptes seran els possibles mecenes del projecte: Maria, Manel, Anna, Pere i Julia.

6.2- Introducció.

El codi estarà format de dos contractes intel·ligents.

- Un primer contracte intel·ligent servirà per crear la recompensa que es donarà als mecenes del projecte que en el nostre cas, seran unes entrades per l'estrena de l'obra. Aquest contracte l'anomenarem "Entrades".
- Un segon contracte on es definiran els termes del projecte com ara: procediment per fer les donacions, forma de emmagatzemar questes donacions, vida del contracte i tancament del contracte enviant els fons recaptats al beneficiari o retornant-los als mecenes si no s'ha aconseguit l'objectiu de finançament desitjat. Aquest contracte l'anomenarem "Mecenatge".

Amb aquest exemple es pretén mostrar el funcionament dels contractes intel·ligents sobre la cadena de blocs. Els Contractes es crearan sobre la cadena de blocs de Ethereum amb les clàusules del contracte especificades en el codi dels contractes. Un cop inserits en la cadena de blocs ja no es podran modificar, solament realitzar o executar les funcions en el temps que s'hagin especificat en el codi.

Així per exemple un cop finalitzada la duració del contracte de "mecenatge" i només un cop finalitzada, és podrà executar la funció checkGoalRaised que solament pot donar dos accions de forma automàtica: transferir l'import recaptat en el projecte al beneficiari acordat en el contracte, o si no s'arriba a l'objectiu de recaptació especificat en el contracte retornar de forma automàtica a cada mecenes l'import que havia donat.

Els codis els contractes intel·ligents a implementar seran el següents:

6.3-Contracte intel·ligent "Entrades"

```
pragma solidity ^0.4.2;
contract Entrades {
  /* Public variables of the token */
  string public standard = 'Token 0.1';
  string public name;
  string public symbol;
  uint8 public decimals;
  uint256 public totalSupply;

  /* This creates an array with all balances */
  mapping (address => uint256) public balanceOf;

  /* This generates a public event on the blockchain that will notify clients */
  event Transfer(address indexed from, address indexed to, uint256 value);

  /* Initializes contract with initial supply tokens to the creator of the contract */
  function Entrades(uint256 initialSupply, string tokenName, uint8 decimalUnits, string
tokenSymbol    ) {
      balanceOf[msg.sender] = initialSupply;    // Give the creator all initial tokens
      totalSupply = initialSupply;                    // Update total supply
      name = tokenName;                                // Set the name for display purposes
      symbol = token Symbol;                         // Set the symbol for display purposes
      decimals = decimalUnits;                       // Amount of decimals for display purposes
  }

  /* Send coins */
  function transfer(address _to, uint256 _value) {
      if (balanceOf[msg.sender] < _value) throw;         // Check if the sender has enough
      if (balanceOf[_to] + _value < balanceOf[_to]) throw;          // Check for overflows
      balanceOf[msg.sender] -= _value;                     // Subtract from the sender
      balanceOf[_to] += _value;                            // Add the same to the recipient
      Transfer(msg.sender, _to, _value);                  // Notify anyone listening that this transfer took
place
  }
}
```

Codi 1 : Contracte intel·ligent "Entrades"

Aquest codi de contracte està extret de la pàgina principal de Ethereum [20] on es descriu el codi necessari per la creació d'un *token*[2] de Ethereum. Nosaltres el que hem fet ha estat adaptar aquest *token* perquè correspongui a les entrades que es donaran com a recompensa als mecenes que col·laborin en el projecte.

En la primera part del contracte es defineix els atributs del contracte, en aquest cas serà el nom, el símbol, si pot tenir decimal el *token* que volem crear. En el nostre cas el *token* que volem crear seran unes entrades de regal pels mecenes. Per últim també indiquem quina serà la quantitat d'entrades que volem crear.

```
string public standard = 'Token 0.1';
string public name;
string public symbol;
uint8 public decimals;
uint256 public totalSupply;
```

També s'incorpora una estructura mapping per emmagatzemar les entrades que es transfereixen i l'adreça on s'han enviat. En el nostre cas aquestes entrades es transfereixen al contracte intel·ligent Mecenatge que serà el responsable de rebre les donacions dels mecenes i calcular si els hi pertoca una recompensa.

```
mapping (address => uint256) public balanceOf;
```

Per finalitzar definirem un event per enregistrar la transacció de l'enviament de les entrades a la cadena de blocs de Ethereum.

```
event Transfer(address indexed from, address indexed to, uint256 value);
```

A continuació hi ha el constructor , on es definiran els valors dels atributs definits anteriorment.

```
  function Entrades(uint256 initialSupply, string tokenName, uint8 decimalUnits, string
tokenSymbol    ) {
    balanceOf[msg.sender] = initialSupply;     // Give the creator all initial tokens
    totalSupply = initialSupply;                 // Update total supply
    name = tokenName;                            // Set the name for display purposes
    symbol = token Symbol;                      // Set the symbol for display purposes
    decimals = decimalUnits;                    // Amount of decimals for display purposes
  }
```

El contracte "Entrades" solament tindrà una funció "transfer" que consistirà en una funció que transfereix les entrades des de el contracte a l'adreça que s'indiqui, pot ser un altre contracte o un compte particular. Un cop fetes les verificacions que l'operació és possible(hi ha prou estoc d'entrades, no es distribuiran més entrades de les que s'han creat inicialment), es procedirà a fer l'enviament de les entrades mitjançat l'event "Transfer" que farà definitiu la transferència d'entrades en la cadena de blocs.

```
  function transfer(address _to, uint256 _value) {
    if (balanceOf[msg.sender] < _value) throw;          // Check if the sender has enough
    if (balanceOf[_to] + _value < balanceOf[_to]) throw;          // Check for overflows
```

[2] Els Token en l'ecosistema Ethereum poden representar qualsevol bé fungible negociables: monedes, punts de fidelitat, certificats d'or, pagarés, en els objectes del joc, etc.

```
        balanceOf[msg.sender] -= _value;                    // Subtract from the sender
        balanceOf[_to] += _value;                           // Add the same to the recipient
        Transfer(msg.sender, _to, _value);                  // Notify anyone listening that this
    transfer took place
      }
```

6.3.1- Implementació del contracte Entrades.

A continuació es descriuen els passos a seguir per implementar aquesta contracte al wallet de Ethereum.

Com s'ha indicat en l'apartat 4.2 desplegament de contractes en el wallet de Ethereum, el codi del contracte s'enganxa en la finestra "Solidity Contract Source Code" de l'apartat contractes del wallet de Ethereum.

Seguidament desplegarem el contracte Entrades seguint els següents passos:

- Seleccionen el contracte Entrades e introduïm el valors dels atributs descrits anteriorment per crear-lo:

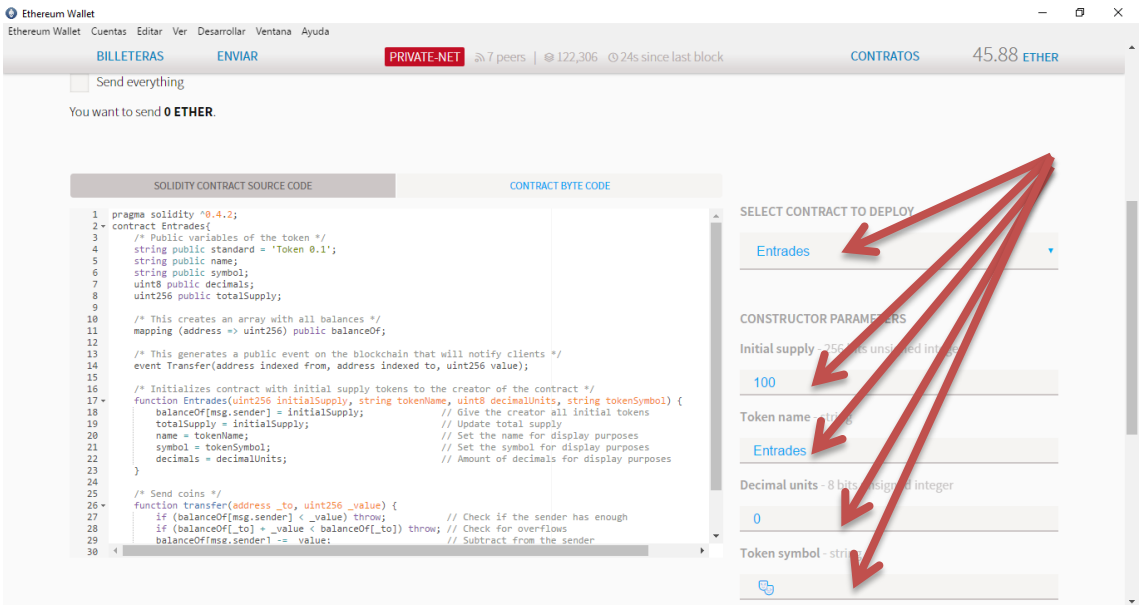

Figura 19 : Inserció dels paràmetres del contracte "Entrades".

- Seguidament premem el botó per desplegar el programa i es crearà la transacció de creació del contracte:

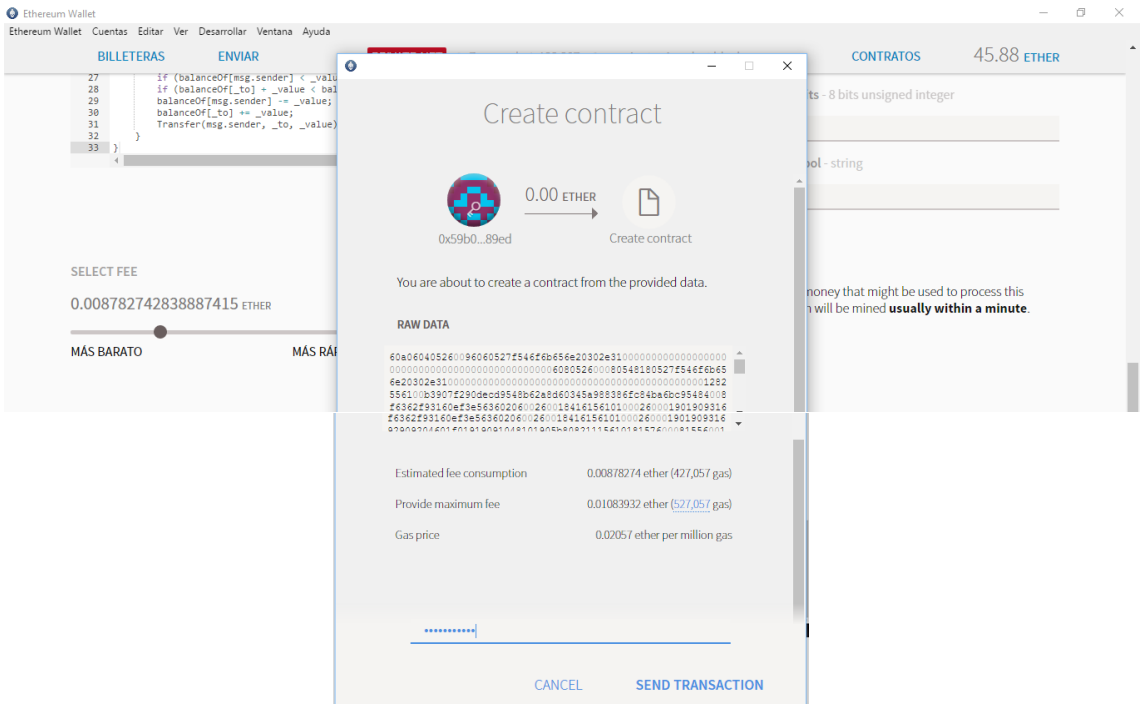

Figura 20 : Creació del contracte "Entrades".

Es procedeix a inserir el "password" personal i s'envia la transacció. Un cop executada la transacció és inserida en la cadena de blocs i ja es pot visualitzar en nou contracte creat en la nostra llista de contractes:

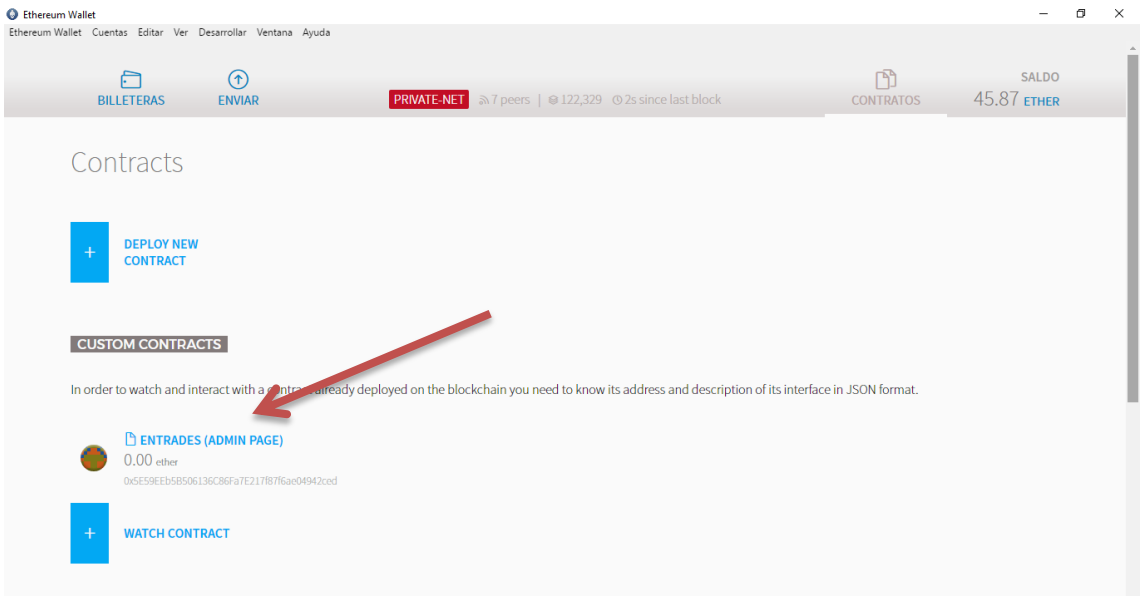

Figura 21 : Secció de contractes amb el nou contracte inserit.

Aquesta transacció també és pot consultar si s'ha inserit en la cadena de blocs de la xarxa de Ethereum amb un explorador de la xarxa Ethereum públic, com pot ser Etherscan on podrem obtenir la confirmació que el procés s'ha realitzat satisfactòriament:

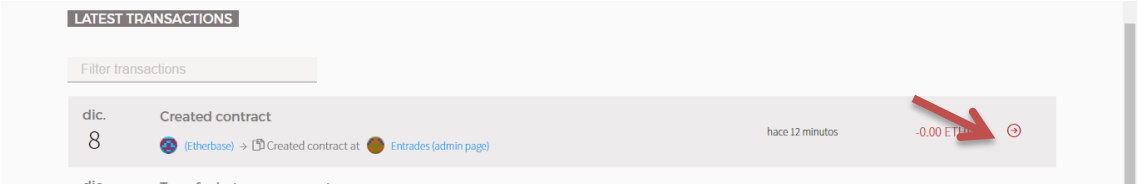

Figura 22 : Llistat de les transaccions del wallet Mist.

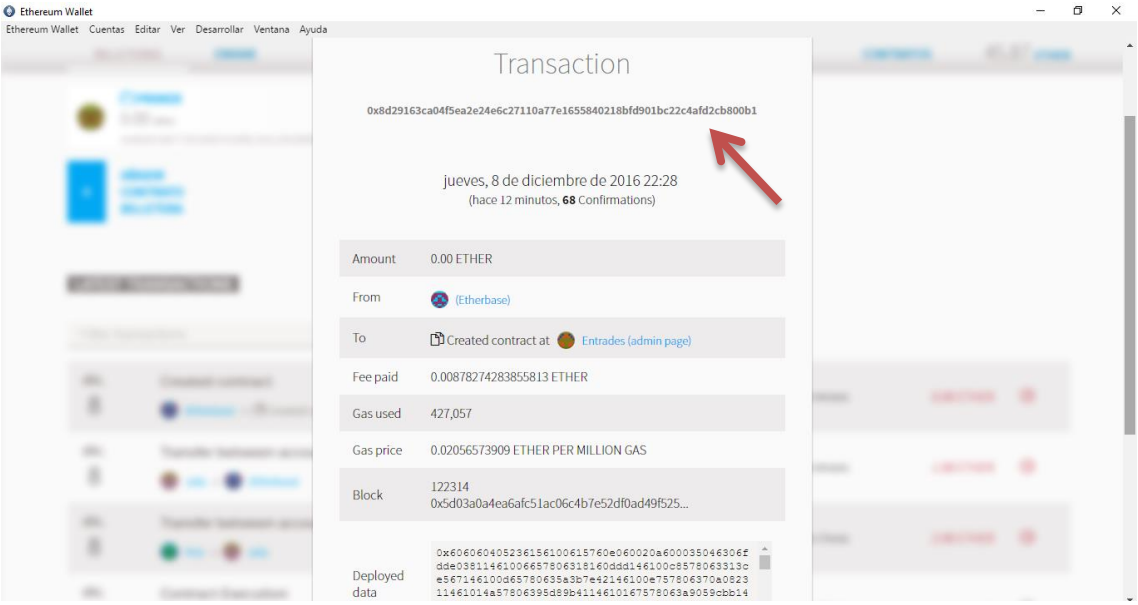

Figura 23 : Detall de la transacció a Ethereum.

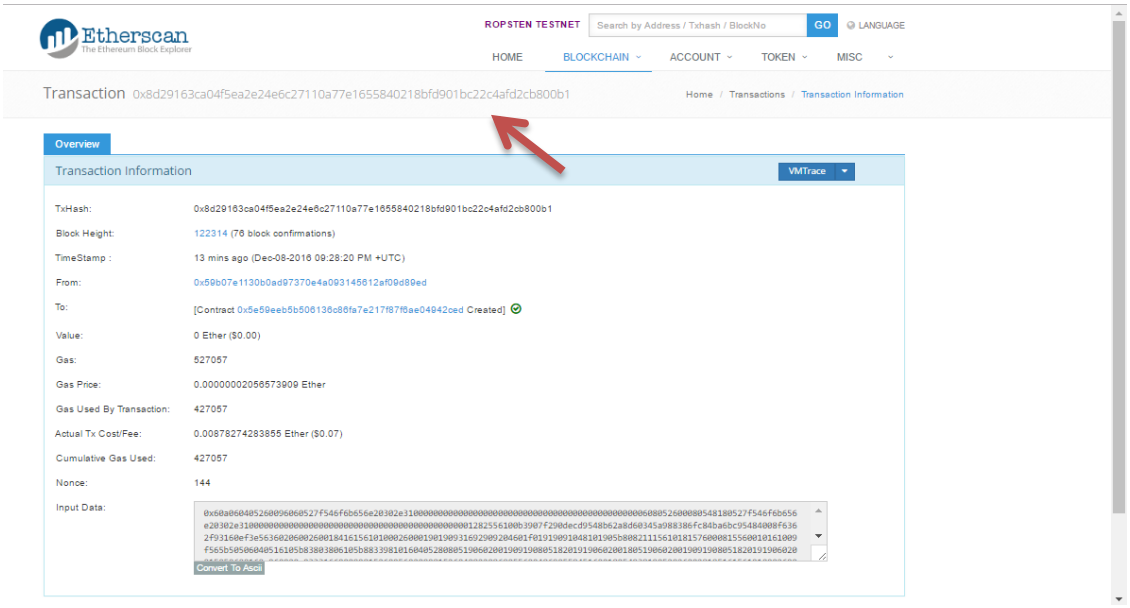

Figura 24 : Detall de la transacció a Etherscan.

6.4-Contracte intel·ligent "Mecenatge"

```
pragma solidity ^0.4.2;
contract token { function transfer(address receiver, uint amount){  } }

contract Mecenatge {
  address public beneficiary;
  uint public fundingGoal;
  uint public amountRaised;
  uint public deadline;
  uint public price;
  token public tokenReward;
  mapping(uint256 => Funder) public funders;
  event FundTransfer(address backer, uint amount, bool isContribution);
  bool crowdsaleClosed = false;
  uint public numFunders;

  /* data structure to hold information about campaign contributors */
  struct Funder {
    address addr;
    uint amount;
  }
  /*  at initialization, setup the owner */
  function Mecenatge(address ifSuccessfulSendTo, uint fundingGoalInEthers, uint
durationInMinutes, uint etherCostOfEachToken, token addressOfTokenUsedAsReward) {
    beneficiary = ifSuccessfulSendTo;                    //if Successful Send To
    fundingGoal = fundingGoalInEthers * 1ether;          //funding Goal In Ethers
    deadline = now + durationInMinutes * 1 minutes       //duration In Minutes
    price = etherCostOfEachToken * 1ether;               //ether Cost Of Each Token
    tokenReward = token(addressOfTokenUsedAsReward) //address Of Token Used As Reward
  }

  /* The function without name is the default function that is called whenever anyone sends funds
to a contract */
  function () payable  {
    if (crowdsaleClosed) throw;
    uint amount = msg.value;
    Funder c= funders[numFunders++];
    c.addr = msg.sender;
    c.amount = msg.value;
    amountRaised += amount;
    tokenReward.transfer(msg.sender, amount / price);
    FundTransfer(msg.sender, amount, true);
  }

  modifier afterDeadline() { if (now >= deadline) _; }

  /* checks if the goal or time limit has been reached and ends the contract */
  function checkGoalReached() afterDeadline {
    if (amountRaised >= fundingGoal){
      if(beneficiary.send(amountRaised)){
       FundTransfer(beneficiary, amountRaised, false);}
      amountRaised = 0;
    }else{
         for (uint i = 0; i < numFunders; ++i) {
                 if(funders[i].addr.send(funders[i].amount)){
                    FundTransfer(funders[i].addr, funders[i].amount, false);}
         }
    }
```

```
        crowdsaleClosed = true;
    }
}
```

Codi 2 : Contracte intel·ligent "Mecenatge"

Aquest nou contracte intel·ligent es torna a construir seguint la mateixa estructura que l'anterior, però afegint noves funcionalitats com són:

La primera línia e contracte indica que pot utilitzar la funció transfer del contracte" Entrades" per transferir *token* (en el nostre cas *token* seran entrades de regal) als meecenes.

```
contract token { function transfer(address receiver, uint amount){ } }
```

Els atributs del contracte seran:

beneficiary	És l'adreça on s'enviarà l'import recaptat si s'arriba a l'objectiu fixat.
fundingGoal	És l'objectiu de recaptació que es pretén assolir en el projecte.
amountRaised	Són els fons recaptats actualment pel projecte.
deadline	És el límit de temps que s'ha acordat per aconseguir recaptar el objectiu de finançament del projecte.
price	És el preu de la donació mínima que té com a premi l'enviament d'una entrada com a premi per la col·laboració en el projecte.
tokenReward	És l'adreça del contracte on s'han creat les entrades , en el nostre cas serà l'adreça del contracte "Entrades"

Es crea també un mapping amb una clau integer i un valor que serà una tupla que contindrà l'adreça del mecenes i la quantitats aportada. Hi haurà un comptador del nombre de mecenes numFunders que ens servirà posteriorment per fer recorreguts pel mapping

```
mapping(uint256 => Funder) public funders;
uint public numFunders;

struct Funder {
   address addr;
   uint amount;
}
```

També tindrà un event "FundTransfer" per registrar a la cadena de blocs les aportacions realitzades pels mecenes. Afegirem un tipus booleà per indicar si és una aportació al projecte o no , ja que aquest mateix event l'utilitzarem per transferir l'import recaptat al beneficiari o retornar l'import als mecenes si no s'arriba a l'objectiu de recaptació.

```
event FundTransfer(address backer, uint amount, bool isContribution);
```

Seguidament tindrà un mètode o funció constructora per incloure els valors dels atributs descrits anteriorment.

```
function Crowdsale(address ifSuccessfulSendTo, uint fundingGoalInEthers, uint
durationInMinutes, uint etherCostOfEachToken, token addressOfTokenUsedAsReward) {
    beneficiary = ifSuccessfulSendTo;              //if Successful Send To
    fundingGoal = fundingGoalInEthers * 1ether;   //funding Goal In Ethers
    deadline = now + durationInMinutes * 1 minutes   //duration In Minutes
    price = etherCostOfEachToken * 1ether;          //ether Cost Of Each Token
    tokenReward = token(addressOfTokenUsedAsReward);          //address Of Token
Used As Reward
  }
```

La següent funció sense nom serà la que s'utilitzarà cada cop que un mecenes faci una donació al projecte. En aquesta funció s'inclourà l'adreça del mecenes i la quantitat aportada en el mapping funders, també s'actualitzarà el valor de l'import recaptat "amountRaised". Seguidament cridarem al mètode "transfer" del contracte entrades i s'enviarà el premi per l'aportació feta que serà el numero d'entrades corresponents segons l'import aportat. Per finalitzar solament quedarà fixar la transacció d'aquest moviment a la cadena de blocs mitjançant el event "FundTransfer".

```
function ()  payable {
  if (crowdsaleClosed) throw;
  uint amount = msg.value;
  Funder c= funders[numFunders++];
  c.addr = msg.sender;
  c.amount = msg.value;
  amountRaised += amount;
  tokenReward.transfer(msg.sender, amount / price);
  FundTransfer(msg.sender, amount, true);
}
```

El modifier `payable` permet rebre Ether juntament amb una trucada o transacció.

En aquest contracte hem definit un "modifier" que verifiqui si ja s'ha sobrepassat el límit de temps "deadLine" especificat en la definició del projecte.

```
modifier afterDeadline() { if (now >= deadline) _; }
```

La última funció checkGoalReached solament es podrà portar a terme un cop sobrepassat el límit de temps de contracte i bàsicament, segons si s'ha arribat a assolir l'import especificat en el projecte, el que farà és enviar els fons recaptats a l'adreça del beneficiary (en el nostre cas l'adreça de "El Folre") o es procedirà a retornar l'import que ha donat cada mecenes si no s'ha arribat al límit de recaptació especificat.

En els dos casos s'inclourà aquest moviment en la cadena de blocs per mitja del event "FundTransfer" posant ,però el booleà en false ja que no es tracta d'una donació.

```
function checkGoalReached() afterDeadline {
   if (amountRaised >= fundingGoal){
          if(beneficiary.send(amountRaised)){
       FundTransfer(beneficiary, amountRaised, false);}
      amountRaised = 0;
   }else{
         for (uint i = 0; i < numFunders; ++i) {
                  if(funders[i].addr.send(funders[i].amount)){
                     FundTransfer(funders[i].addr, funders[i].amount, false);}
         }
   }
   crowdsaleClosed = true;
}
```

6.4.1- Implementació del contracte Mecenatge.

La implementació del nou contracte a la wallet de Ethereum seguirà el mateix procediment que s'ha descrit en el contracte "Entrades". A continuació mostrarem els punts més importants:

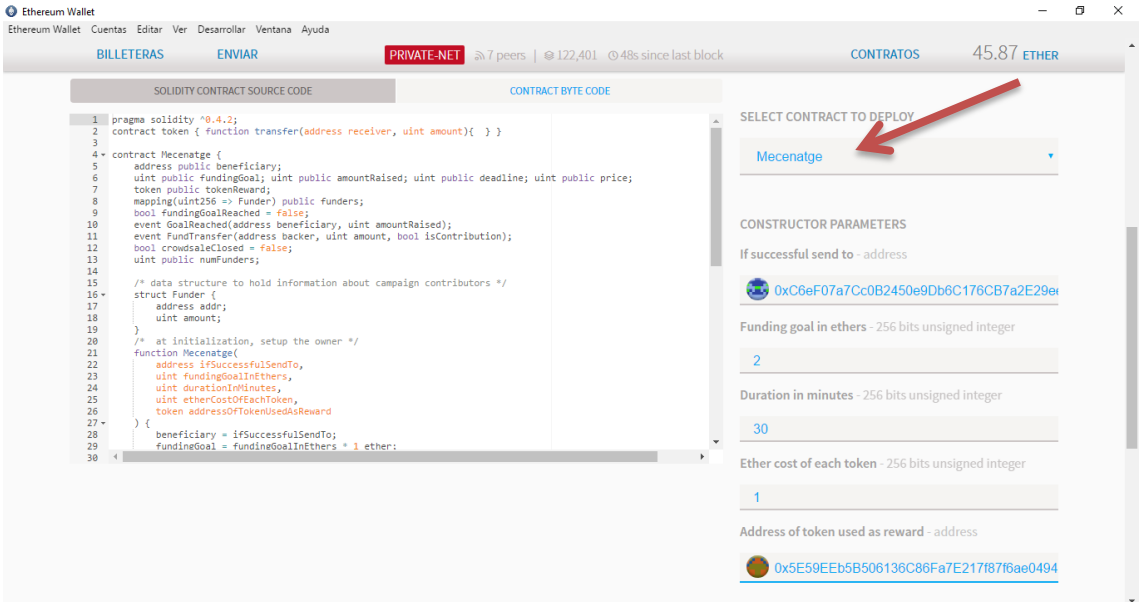

Figura 25 : Inserció dels paràmetres del contracte "Mecenatge".

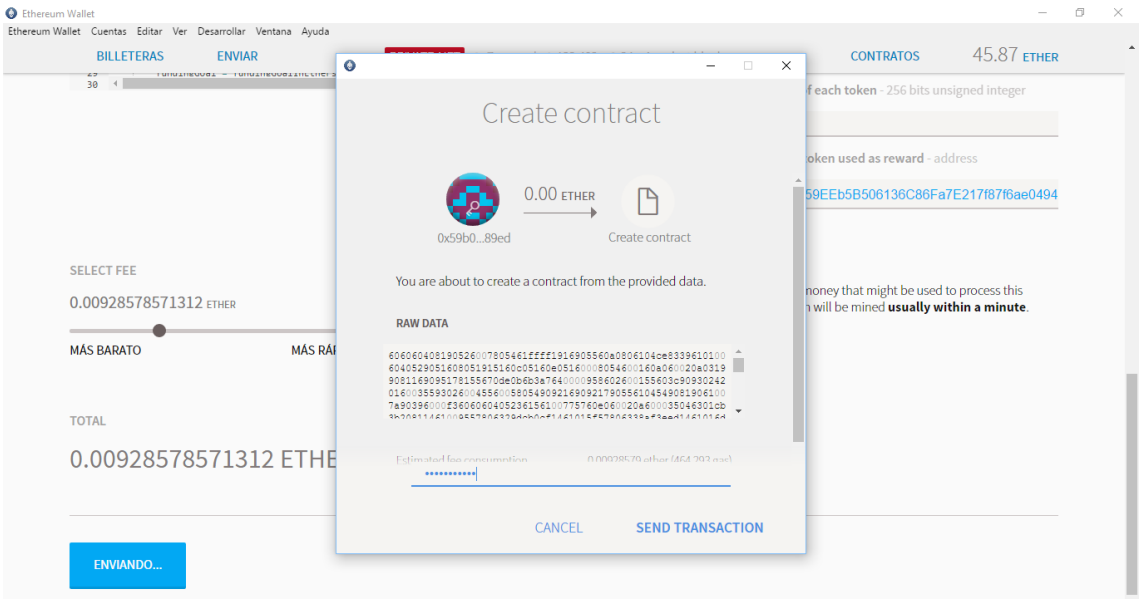

Figura 26 : Transacció de creació del contracte "Mecenatge".

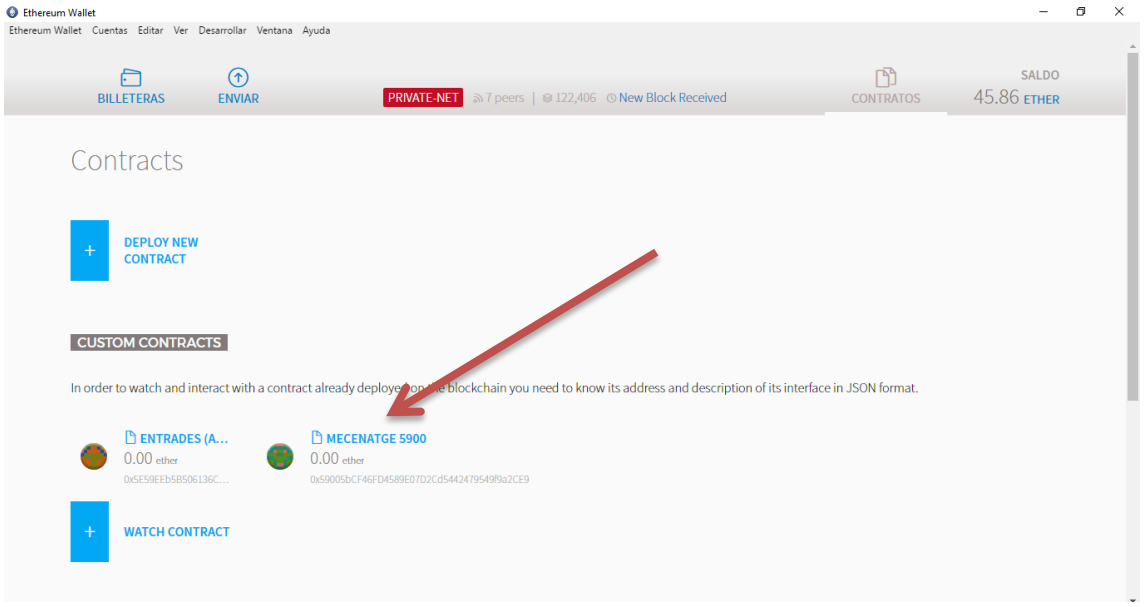

Figura 27 : Llistat dels programes existents a la wallet.

6.4.2- Transferència d'entrades entre els Contractes Intel·ligents .

El que es de fer a continuació és transferir una quantitat d'entrades de regal (en el nostre cas 10 que seran suficients per realitzar les proves de funcionalitat) al contracte" mecenatge" perque pugui fer l'intercanvi amb el mecenes un cop feta una aportació. Es torna a mostrar la situació inicial del wallet per fer més visible la transferència que es farà seguidament:

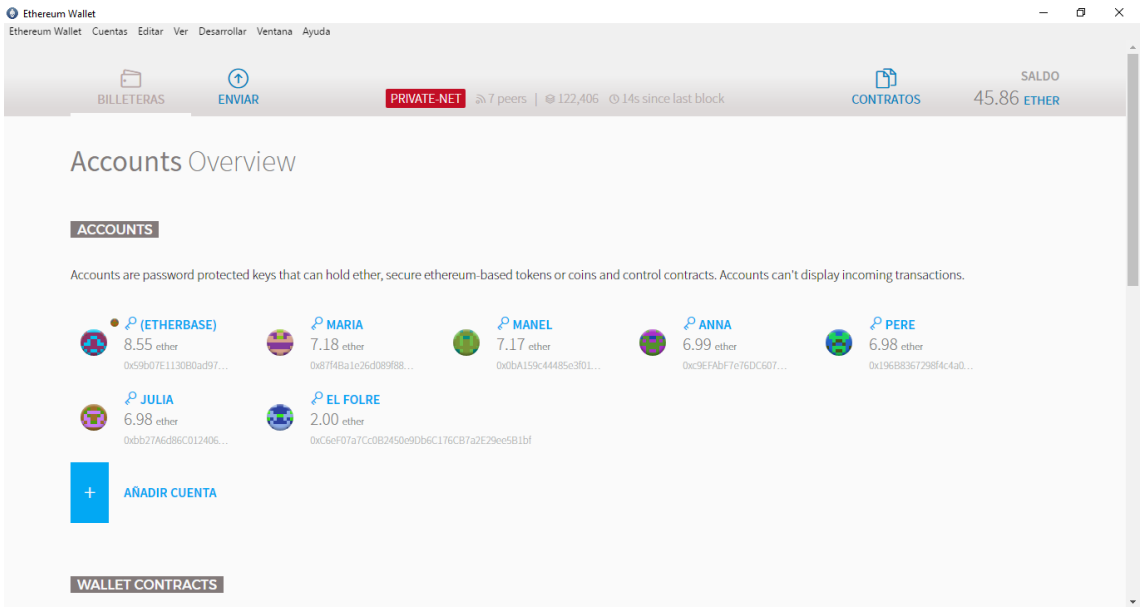

Figura 28 : Situació de partida del wallet.

Per tal de realitzar la transferència de les entrades s'obre el contracte "entrades" i és selecciona la funció "Transfer". Es desplegarà un formulari on s'haurà d'introduir l'adreça del contracte al que volem transferir les entrades, en el nostre cas serà l'adreça del contracte "Mecenatge" i la quantitat d'entrades que es vol transferir.

Seguidament s'ha indicar a quin compte s'ha de carregar els costos (GAS) de la transacció,en el nostre cas utilitzarem el compte principal o del promotor del wallet "Etherbase" i ja solament queda prémer executar.

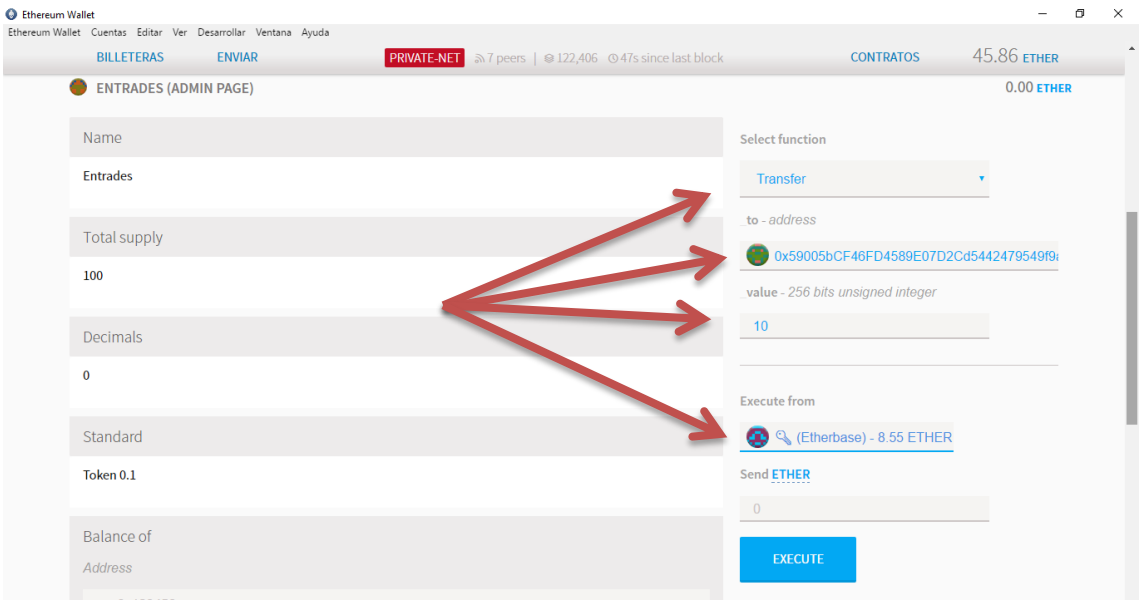

Figura 29 : Transferència d'entrades al contracte "Mecenatge".

Un cop executada la transacció apareixerà una icona al costat del nom del contracte indicant que conté el *token* transferit (en el nostre cas el *token* són les entrades recompesa als mecenes per realitzar les aportacions)

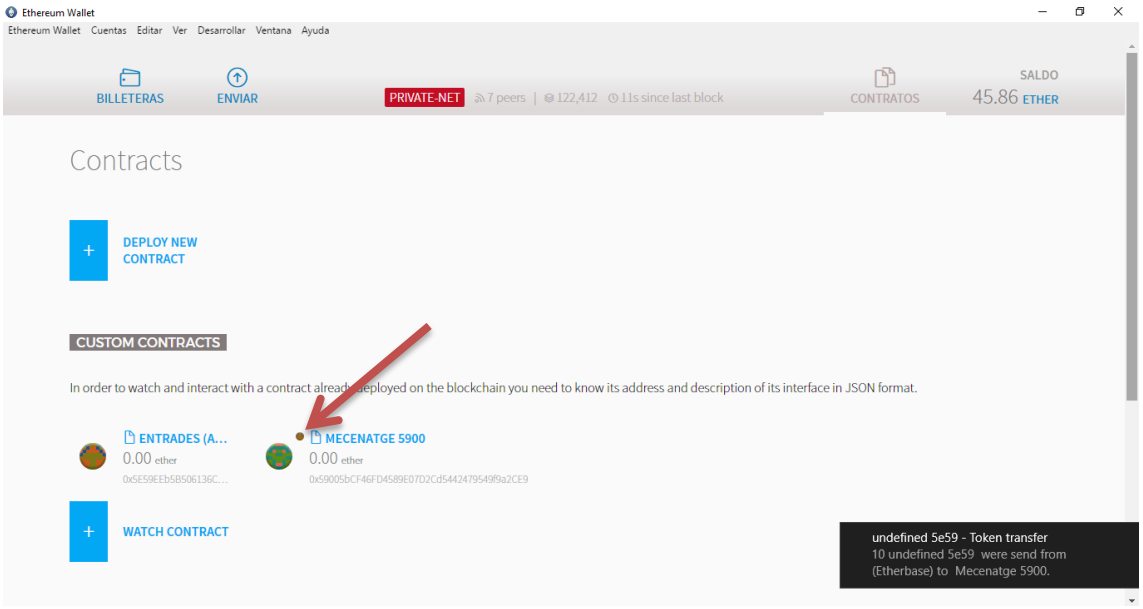

Figura 30 : Llistat dels contractes del wallet

7-Proves de conceptes.

Per verificar la funcionalitat del contracte i per simplificar el procés es realitzaran dues proves:

1. Prova d'execució del contracte un cop s'ha sobrepassat l'objectiu de recaptació "fundingGoal" del projecte. Amb aquesta prova quedarà inclòs també el cas en que les aportacions siguin iguals a l'objectiu.

   Els paràmetres que introduirem per realitzar aquesta prova serà l'adreça del compte de "El Folre" com a beneficiary, un objectiu "fundingGoal" de 2 Ethers , una duració del contracte "deadLine" de 20 minuts, i un preu "price" o valor d'aportació mínima a la que correspon com a premi una entrada de la inauguració de 1 Ether. Per finalitzar s'ha d'introduir l'adreça del contracte "Entrades" on s'ha creat el token entrades per saber la naturalesa del premi que es concedeix.

2. Prova d'execució del contracte quan no s'ha aconseguit l'objectiu de recaptació "fundingGoal" del projecte. En aquest cas s'ha de retornar l'import que han aportat a tots els mecenes que han participat en el projecte.

   Els paràmetres que introduirem oer realitzar aquesta prova serà l'adreça del compte de "El Folre" com a beneficiary, un objectiu "fundingGoal" de 4 Ethers , una duració del contracte "deadLine" de 30 minuts, i un preu "price" o valor d'aportació mínima a la que correspon com a premi una entrada de la inauguració de 1 Ether. Per finalitzar s'ha d'introduir l'adreça del contracte "Entrades" on s'ha creat el token entrades per saber la naturalesa del premi que es concedeix.

Prova 1 – Execució del contracte si s'ha assolit l'objectiu de finançament.

Un cop ja s'ha proveit d'entrades de regal al contracte "Mecenatge" ja podem simular les aportacions que fan els mecenes. Agafarem el compte de la Maria i seleccionarem l'opció "enviar". S'obrirà una finestra nova amb un formulari on s'haurà d'inserir l'adreça del contracte "Mecenatge" a on es vol fer la transferència i la quantitat que es vol aportar, en aquest cas 1 Ether.

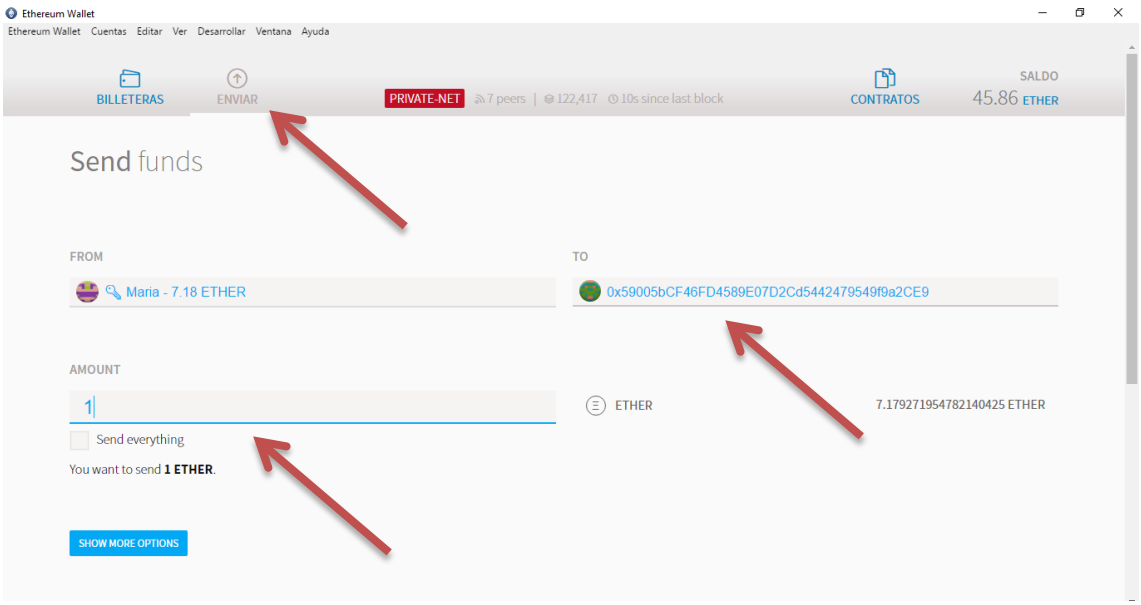

Figura 31 : Aportació d'un mecenes al contracte.

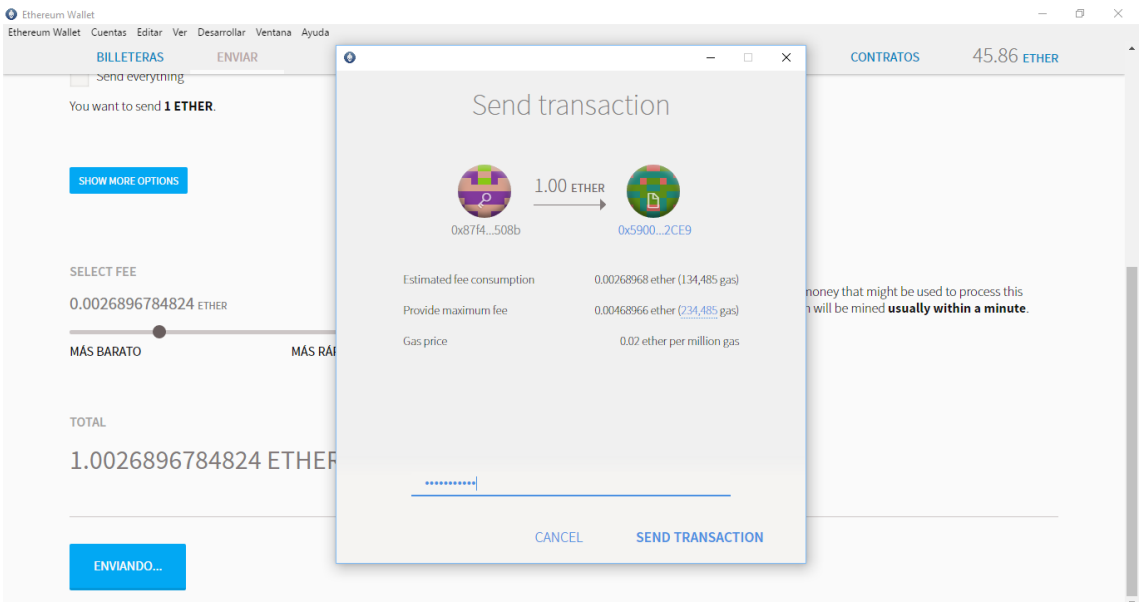

Figura 32 : Confirmació de la transacció per part del mecenes

Repetirem el mateix procediment amb els comptes de Anna, Pere i Manel, de tal forma que s'haurà fet aportacions al contracte per un valor de 4 Ether, superior als 2 Ether que s'han especificat com a objectiu del contracte.

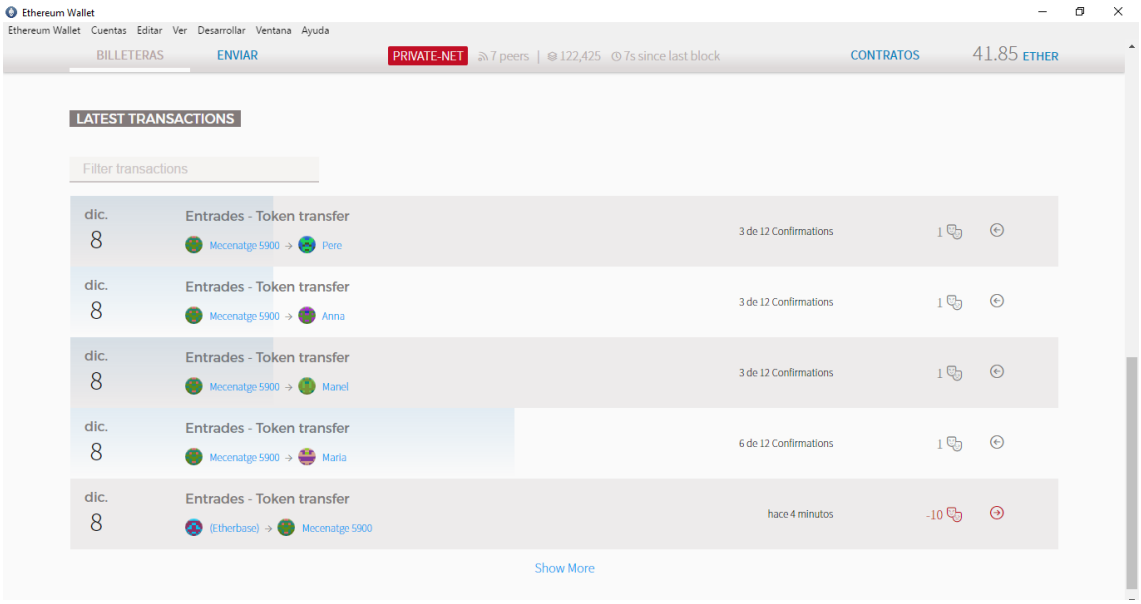

Figura 33 : Confirmació de les transaccions

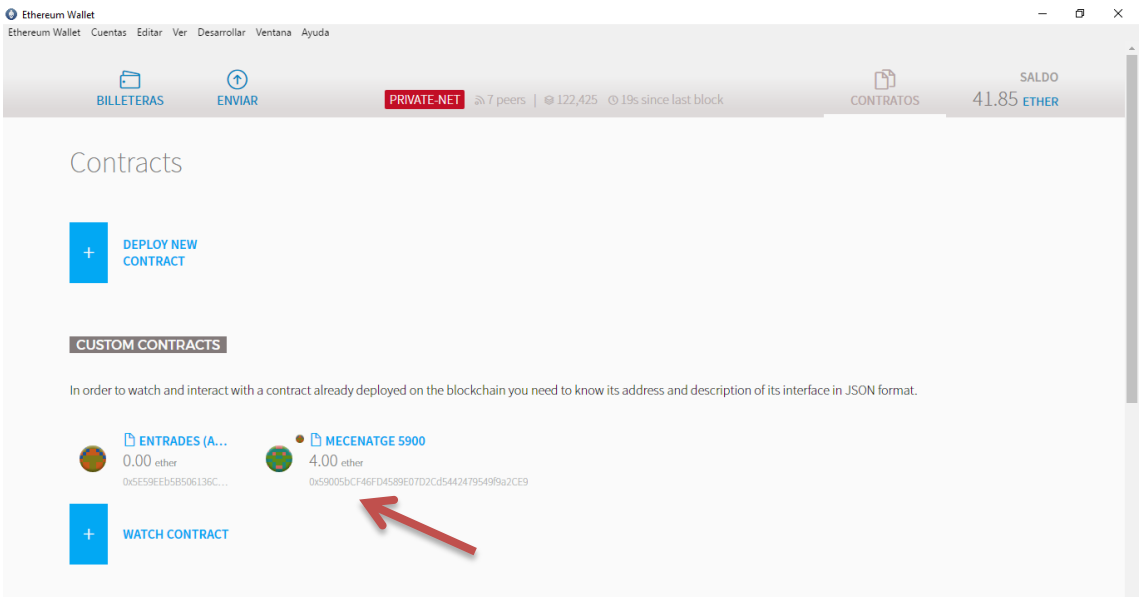

Figura 34 : Llistat de contractes del wallet amb el nou estat

En aquesta figura es pot verificar com el contracte Mecenatge té un saldo de 4 Ethers, que són les aportacions que han fet els mecenes amb les últimes transaccions.

Seleccionant el contracte Meenatge, entrarem a la seva finestra on podrem veure tota la informació rellevant del contracte com ara: els parametres especificats a l'inici, una llista de les aportacions fetes per cada mecenes i també un llistat dels events que s'han realitzat en aquest contracte.

Amb aquesta informació podrem verificar que el procés de donacions fetes anteriorment han funcionat correctament.

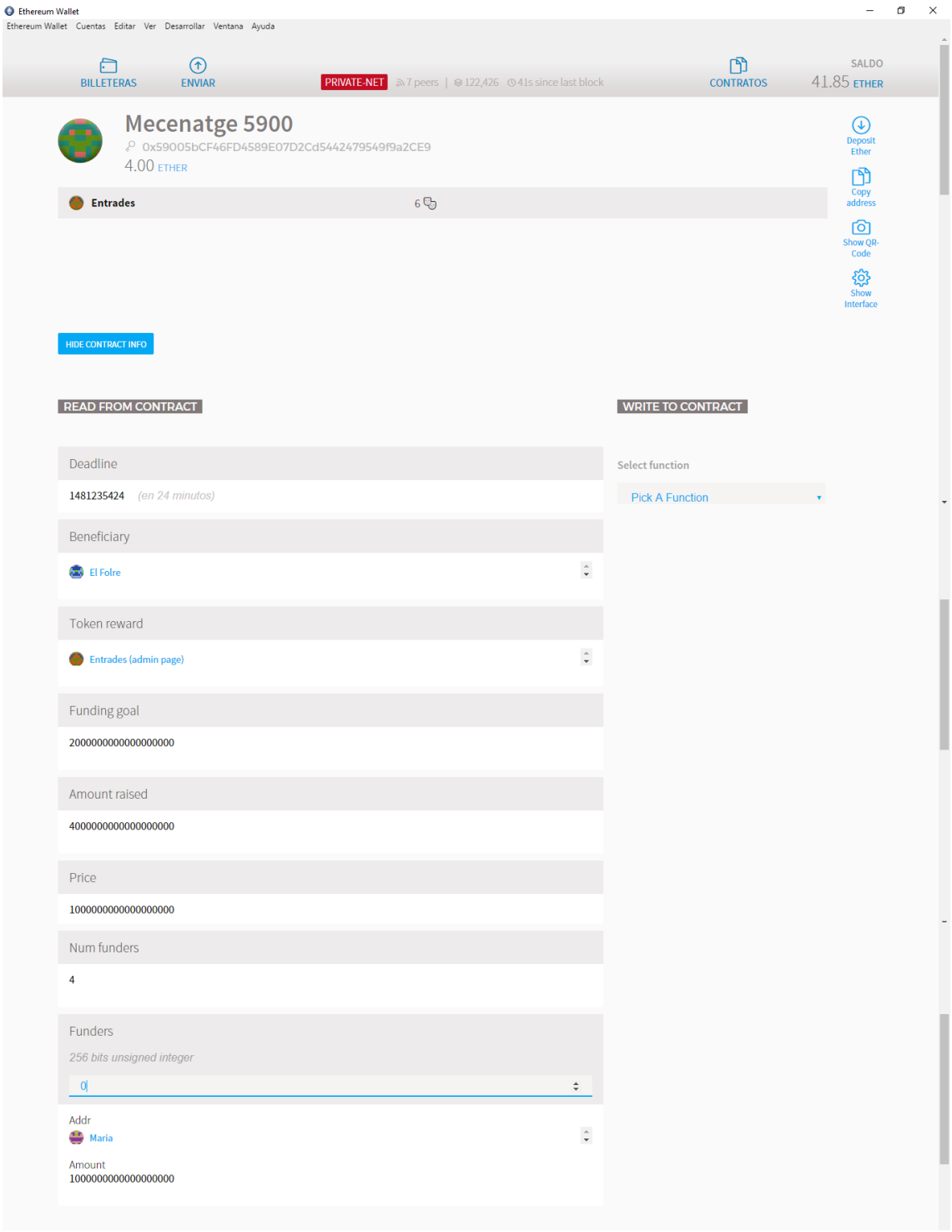

Figura 35 : Finestra informativa contracte "Mecenatge"

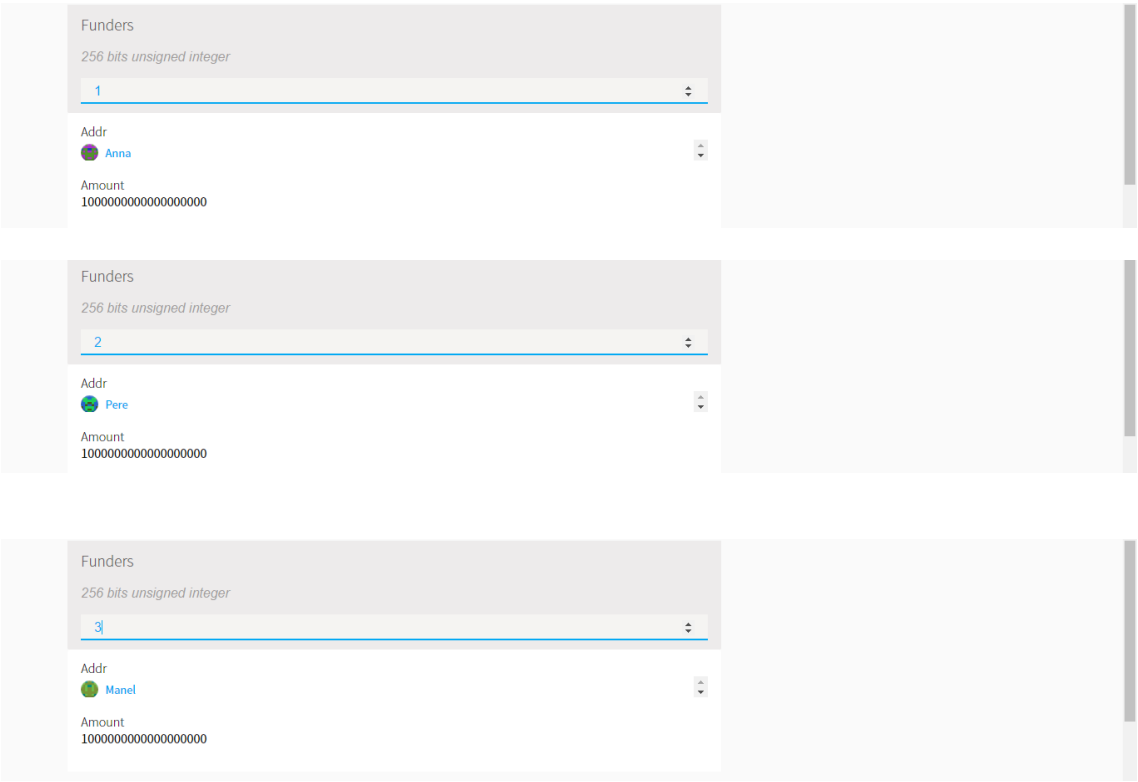

Figura 36 : Llistat de les aportacions fetes pels mecenes

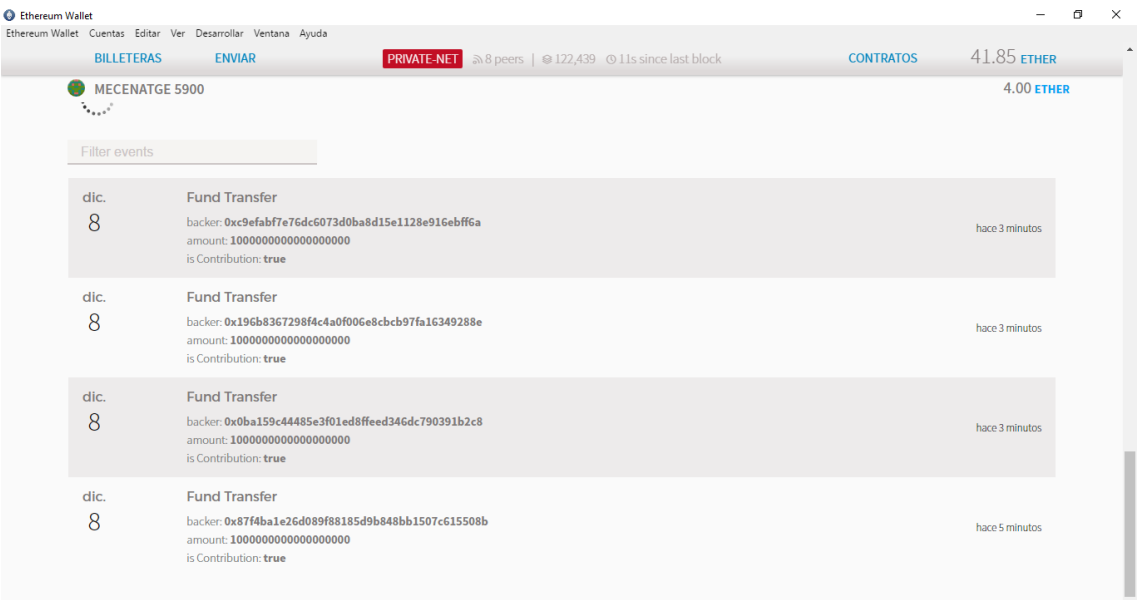

Figura 37 : Llistat dels “events” aplicats pel contracte.

I també verifiquem l'estat dels comptes del wallet on es pot observar com en els comptes dels mecenes del projecte s'ha reduit la quantitat de Ether equivalent al import que han donat al projecte. També es pot veure com han rebut el premi o entrada per haver realitzat la donació (icona al costat del nom del compte).

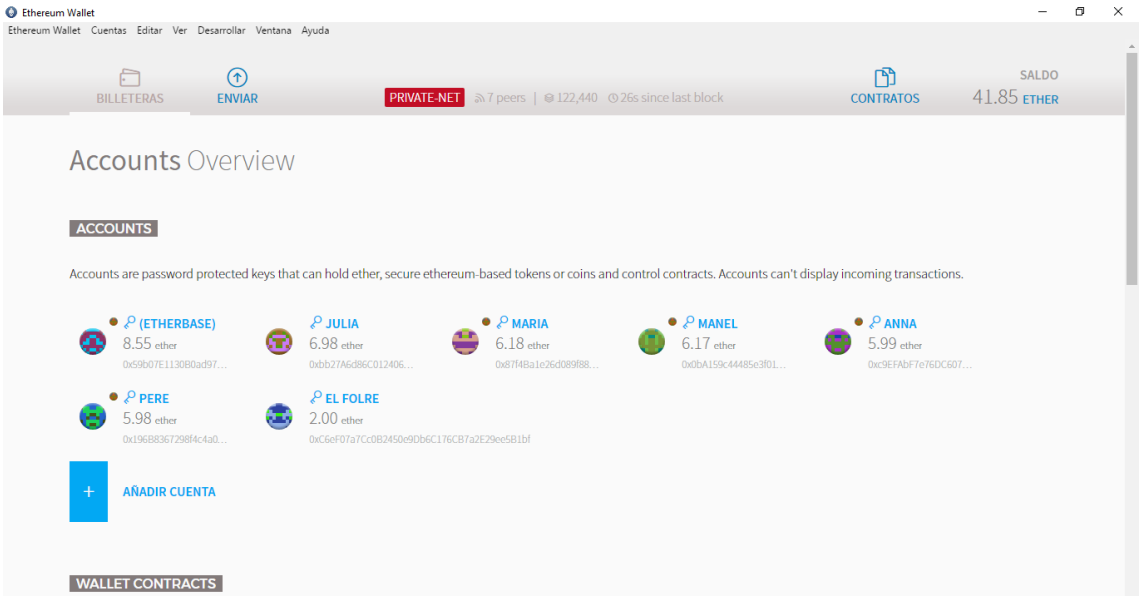

Figura 38 : Situació del wallet un cop fetes les donacions.

Ja solament queda esperar a que pasi el temps i es sobrepasi la duració o "deadLine" especificat al inici per poder executar la funció "chechGoalRaised" del contracte "Mecenatge" amb el compte del promotor per realitzar de forma automàtica la finalització del contracte. En quest cas s'hauria de transferir l'import recaptat a l'adreça del beneficiari.

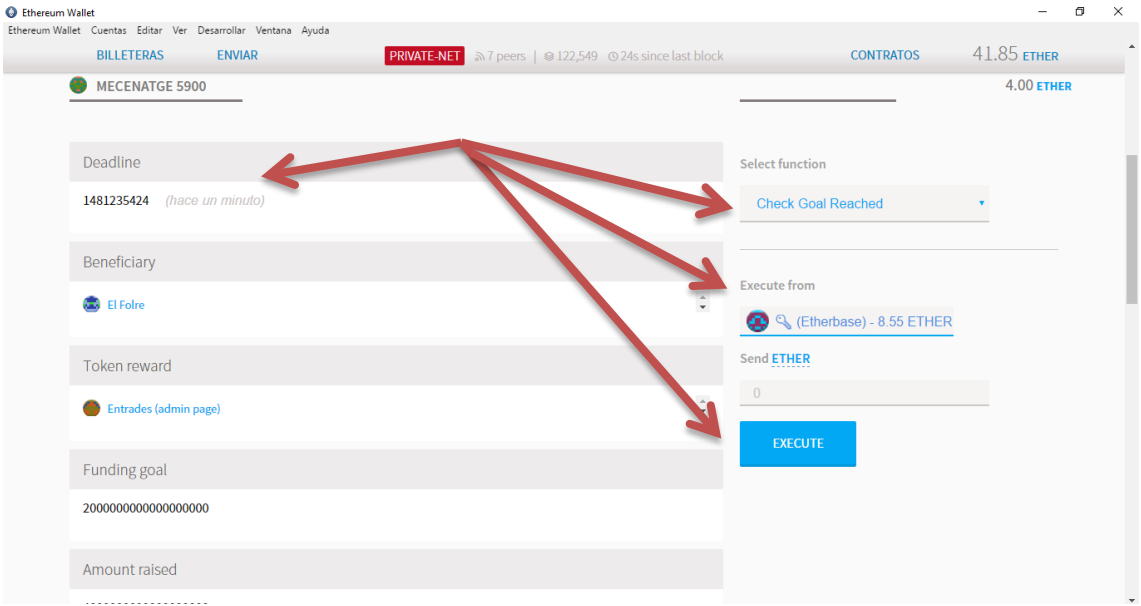

Figura 39 : Execució mètode final del contracte

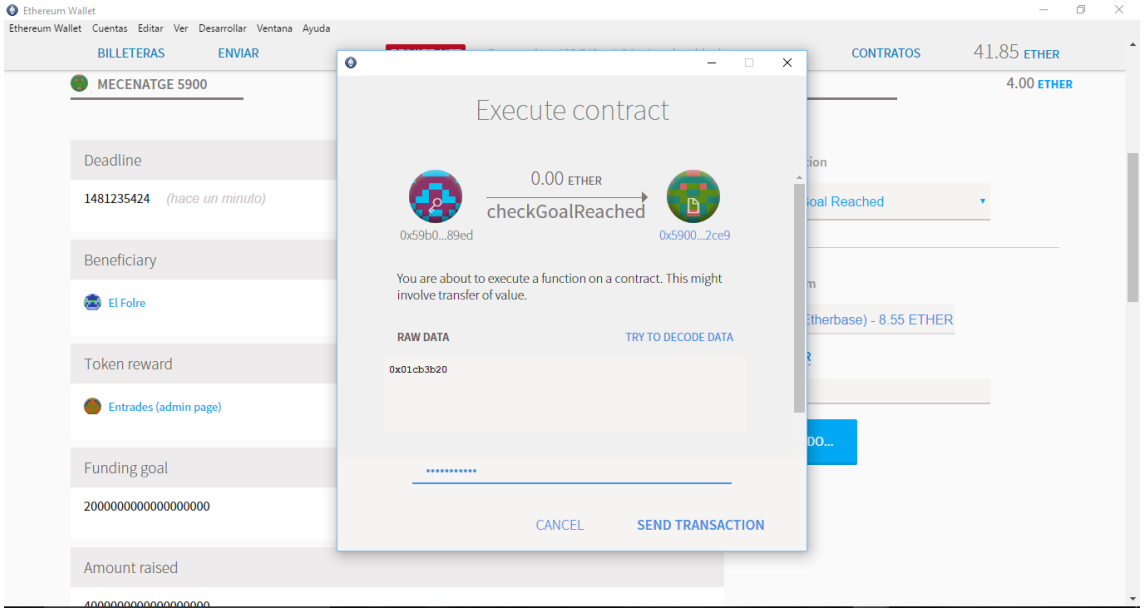

Figura 40 : Transacció execució funció "checkGoalRaised".

Un cop confirmada la transacció ja podem verificar si s'han transferit els fons recaptats a l'adreça del beneficiari "El Folre". Aquesta verificació la farem mirant el llistat de comptes del wallet per observant si s'ha augmentat en 4 Ether el contingut del compte "El Folre" i que el contrate "Mecenatge" s'ha tornat a quedar amb 0 Ether, a més del llistat de events realitzat pel contracte.

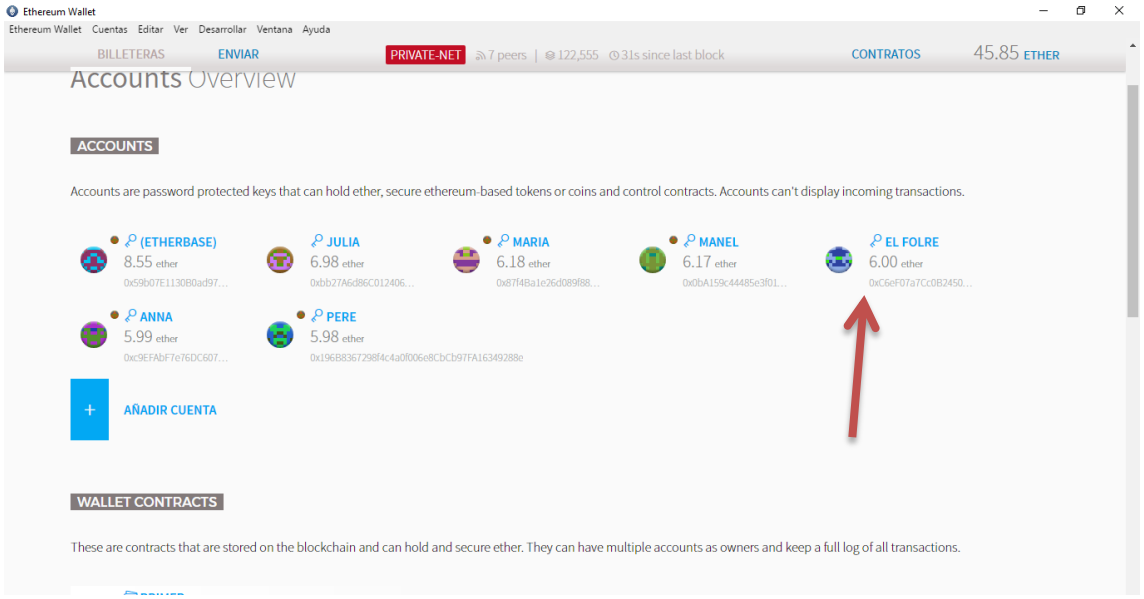

Figura 41 : Situació wallet un cop executat el contracte "Mecenatge"

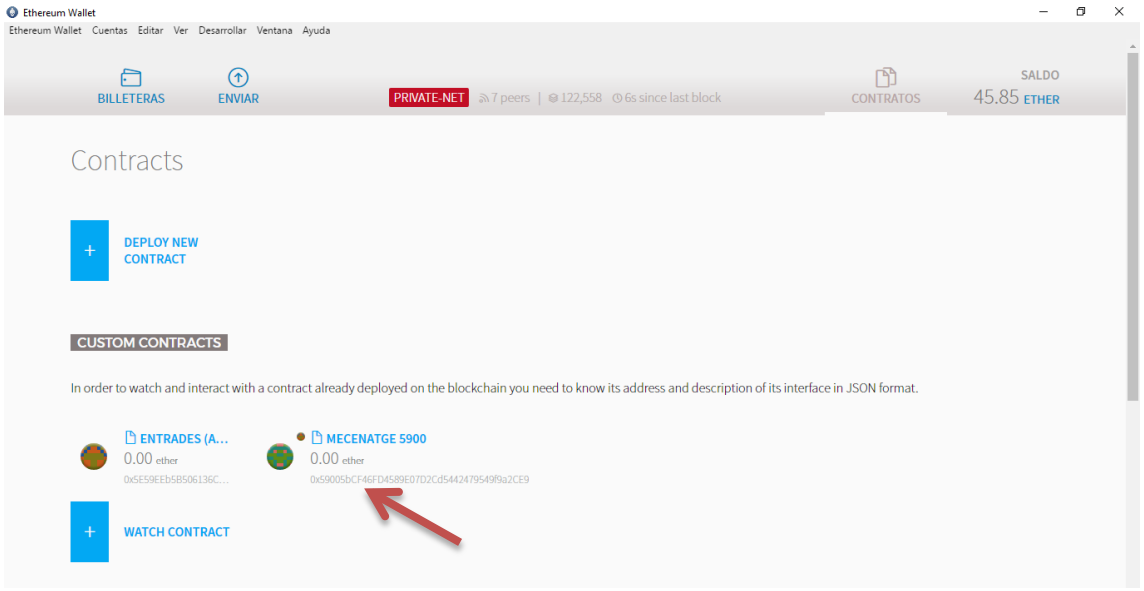

Figura 42 : Llistat de contractes de la wallet.

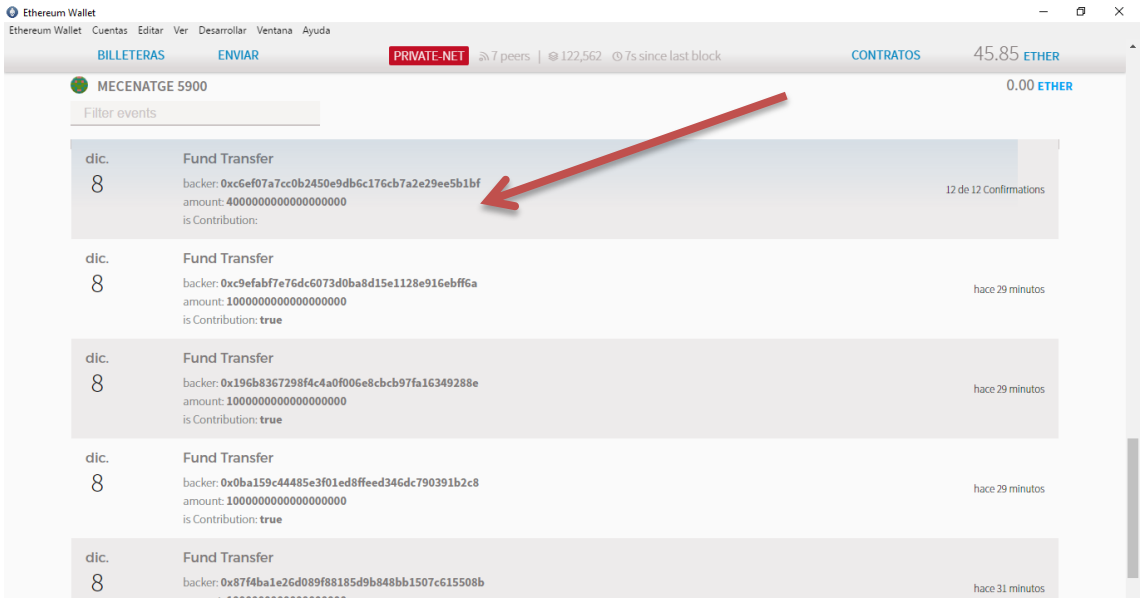

Figura 43 : Llistat events del contracte “Mecenatge”

Hem pogut verificar que el contracte ha realitzat de forma automàtica y correcta la clausula o funció definida en el seu codi i ha enviat l'import recaptat pel projecte a l'adreça del beneficiari "El Folre", ha pasat de tenir un balanç inical de 2 Ethers als 6 Ethers finals..

Queda una última verificació per realitzar i és comprovar que un cop s'ha executat la funció de tancament del contracte "checkGoalRaised" ja no es poden realitzar noves aportacions al contracte. Així que el que fem és intentar fer una nova aportació al contracte per veure quin és el seu comportament:

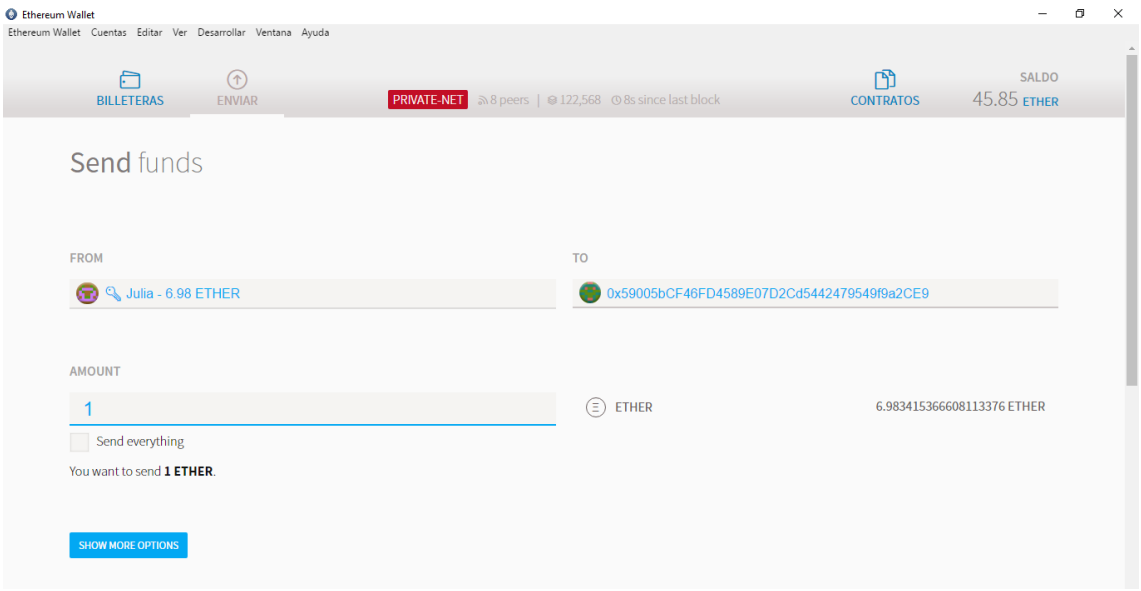

Figura 44 : Intent de nova aportació al contracte

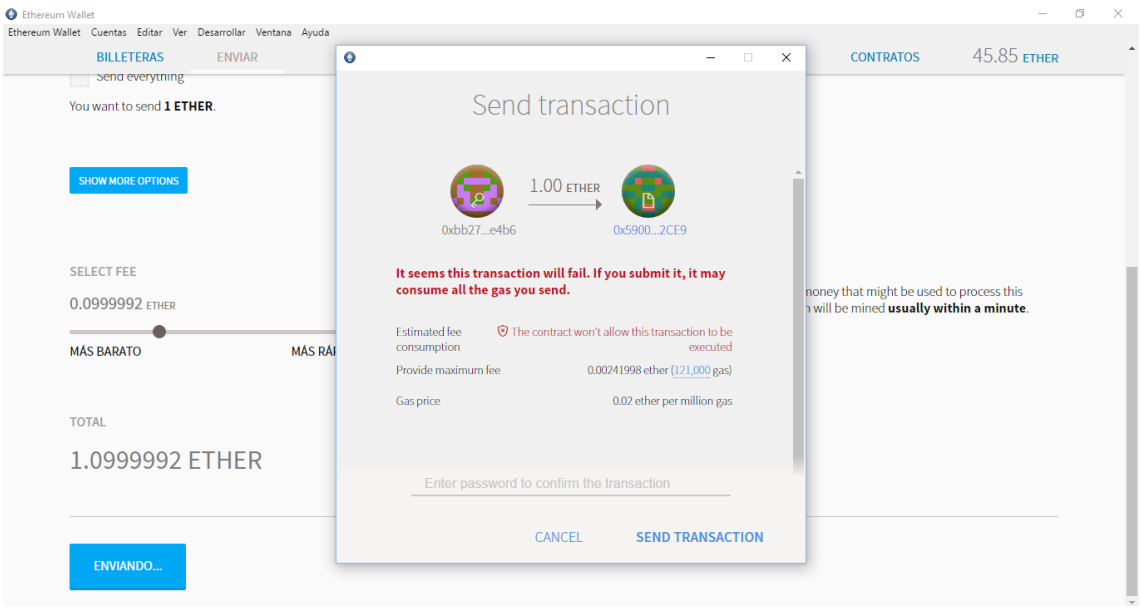

Figura 45 : Validació de la transacció.

Es verifica que no deixa realitzar la transacció apareixent un missatge d'error que indica que el contracte no acceptarà aquesta transacció, per tant funciona correctament.

Prova 2 – Execució del contracte si no s'ha assolit l'objectiu de finançament.

Per realitzar aquesta segona prova necessitem tornar a crear un segon contracte de mecenatge seguint el procés descrit en el punt anterior

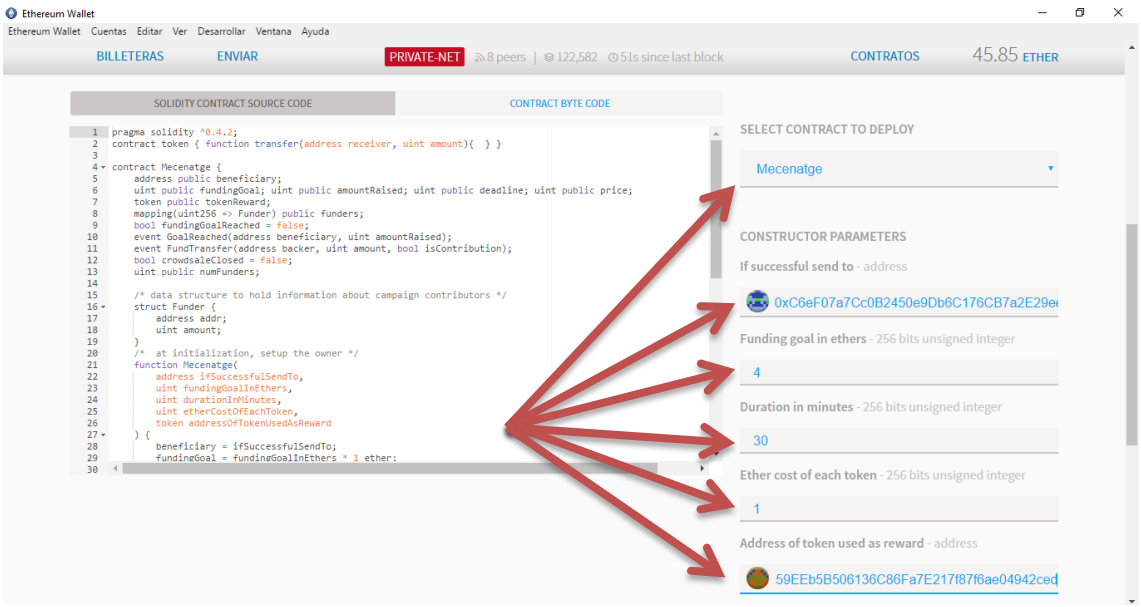

Figura 46 : Introducció paràmetres inicials nou contracte Mecenatge

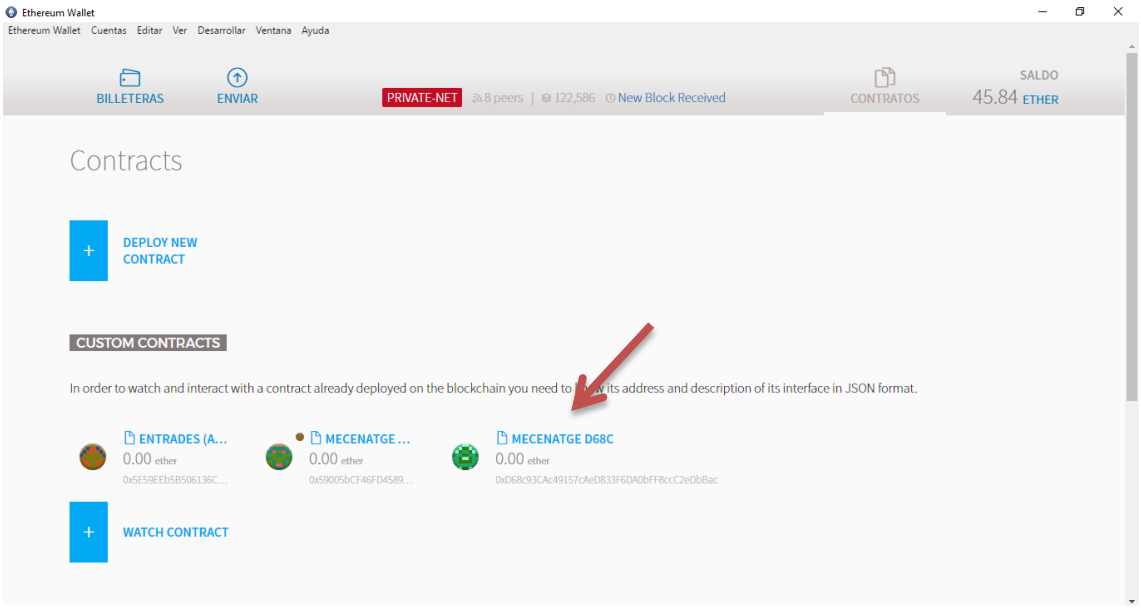

Figura 47 : Llistat dels contractes del wallet

També serà necessari transferir-li els *token* o entrades de recompensa al nou contracte mecenatge per poder oferir el premi als mecenes que decideixen participar en el procés.

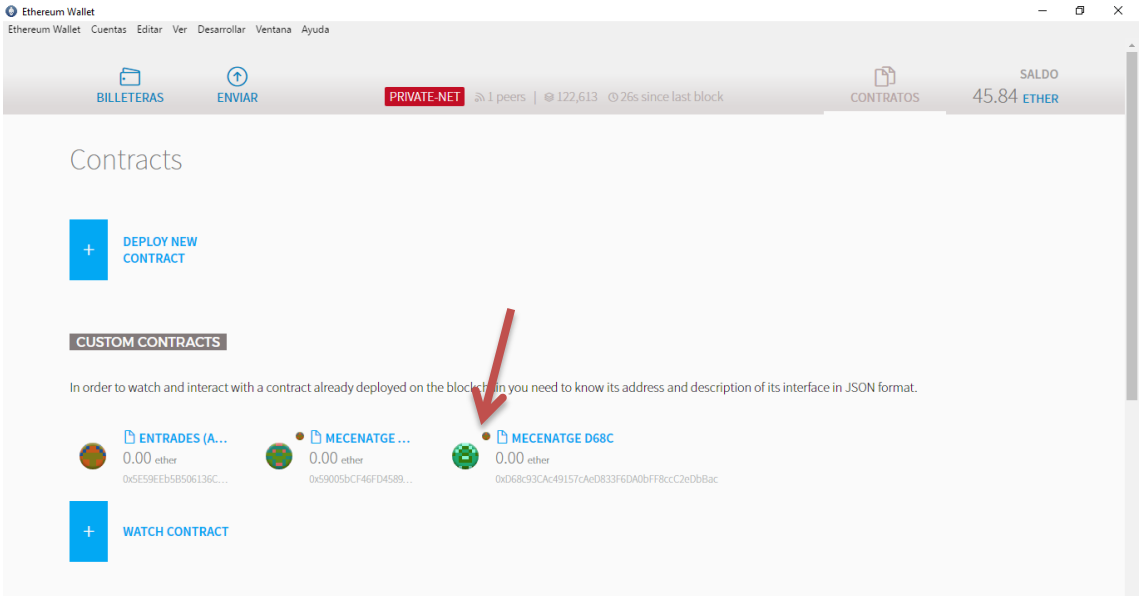

Figura 48 : Llistat dels contractes del wallet amb les entrades de regal.

Seguint el procés de la prova 1 simularem aportacions al nou contracte "Mecenatge" però solament aportarem 3 Ether per no arribar a l'objectiu de recaptació del contracte que és 4 Ether.

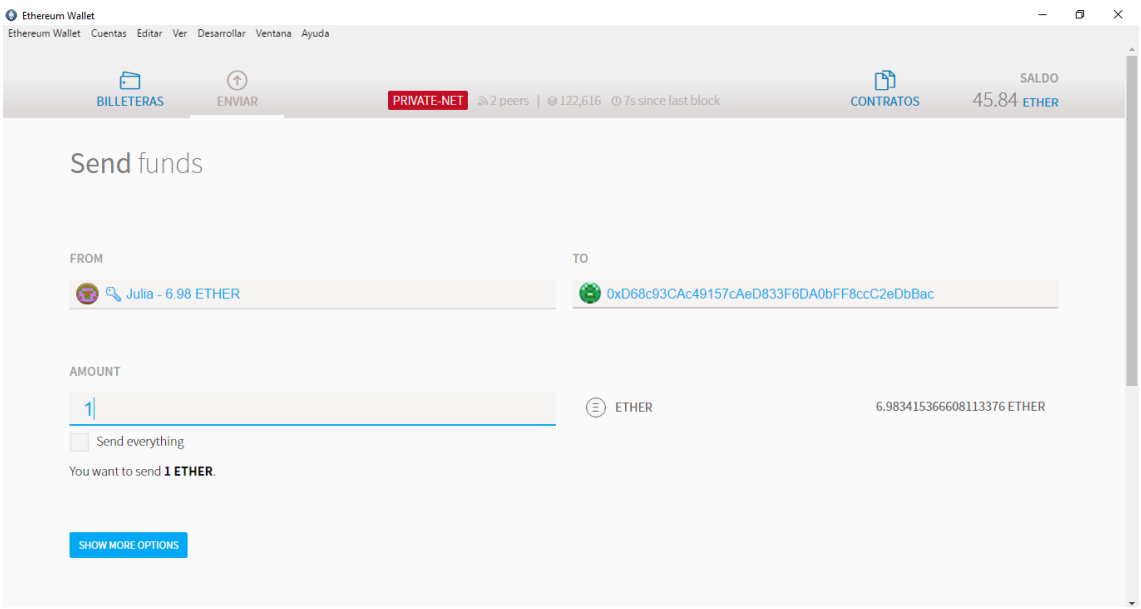

Figura 49 : Aportació al nou contracte "Mecenatge"

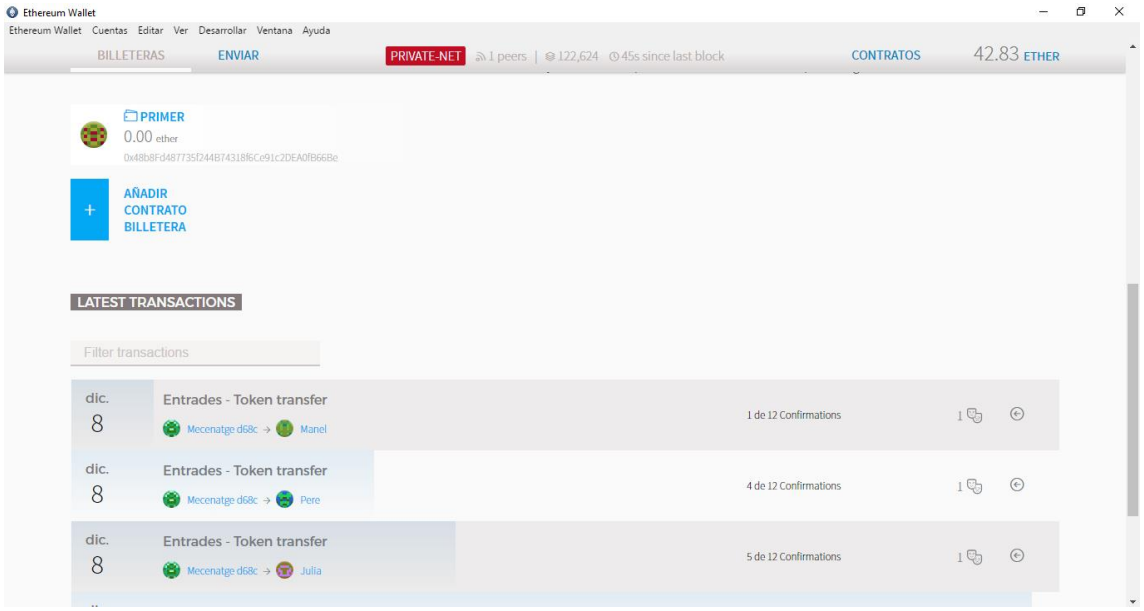

Figura 50 : Llistat de les transaccions

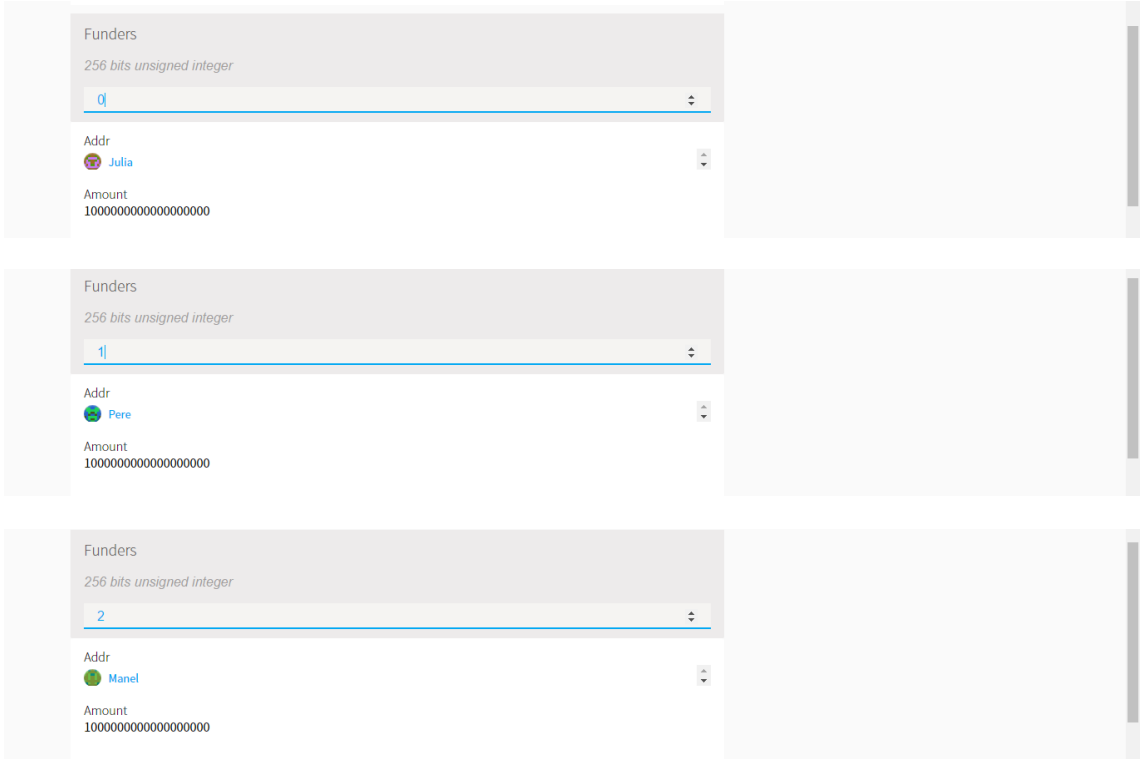

Figura 51 : Llistat de les aportacions fetes al nou contracte.

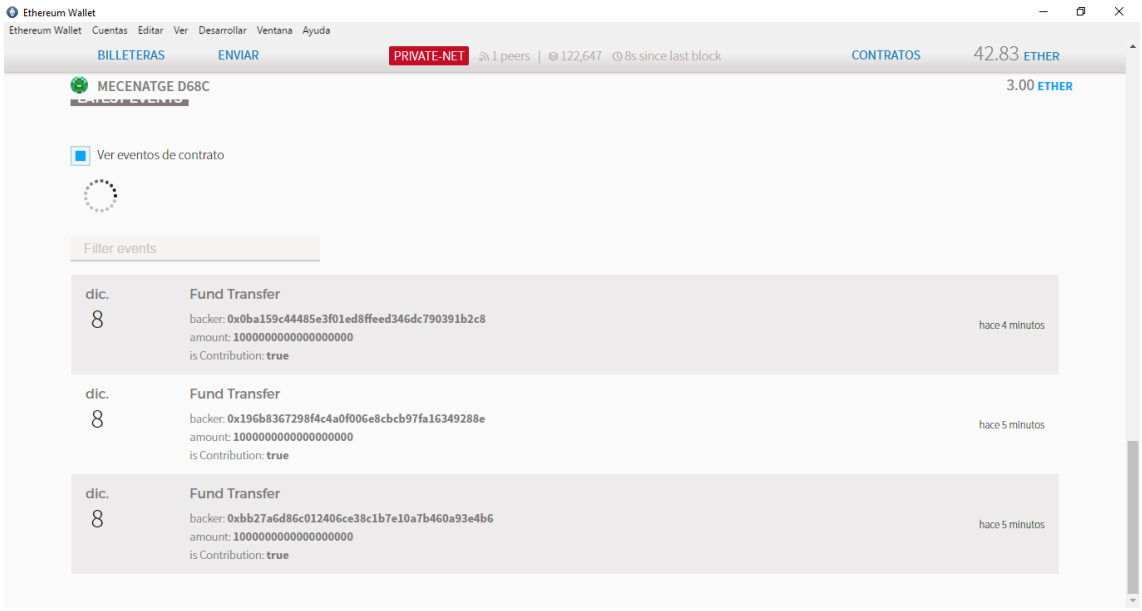

Figura 52 : Llistat dels "event" del nou contracte Mecenatge.

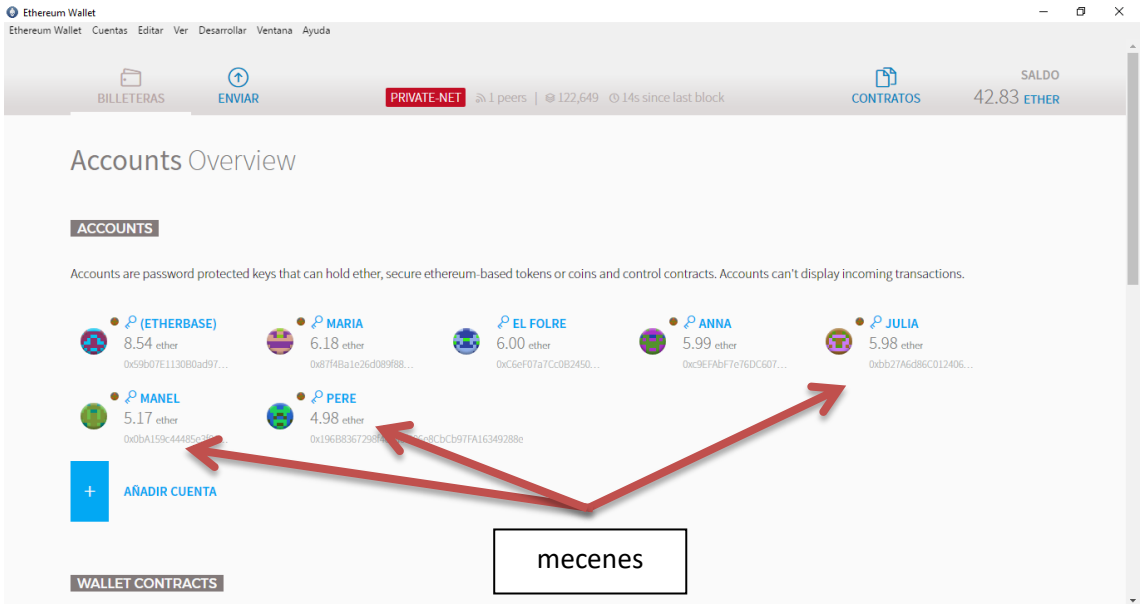

Figura 53 : Situació del wallet un cop fetes les aportacions

Un cop ja hem verificat que les transaccions de les donacions dels mecenes al nou contracte "Mecenatge s'ha realitzat d forma correcte i de que tenim l'escenari adequat per poder realitzar la segona prova: no s'ha assolit l'objectiu de recaptació esperat pel contracte. Solament ens queda esperar a es superi la duració "deadLine" especificat en el codi del contracte per realitzar el mètode de tancament del contracte "checkGoalRaised" i verificar el resultat.

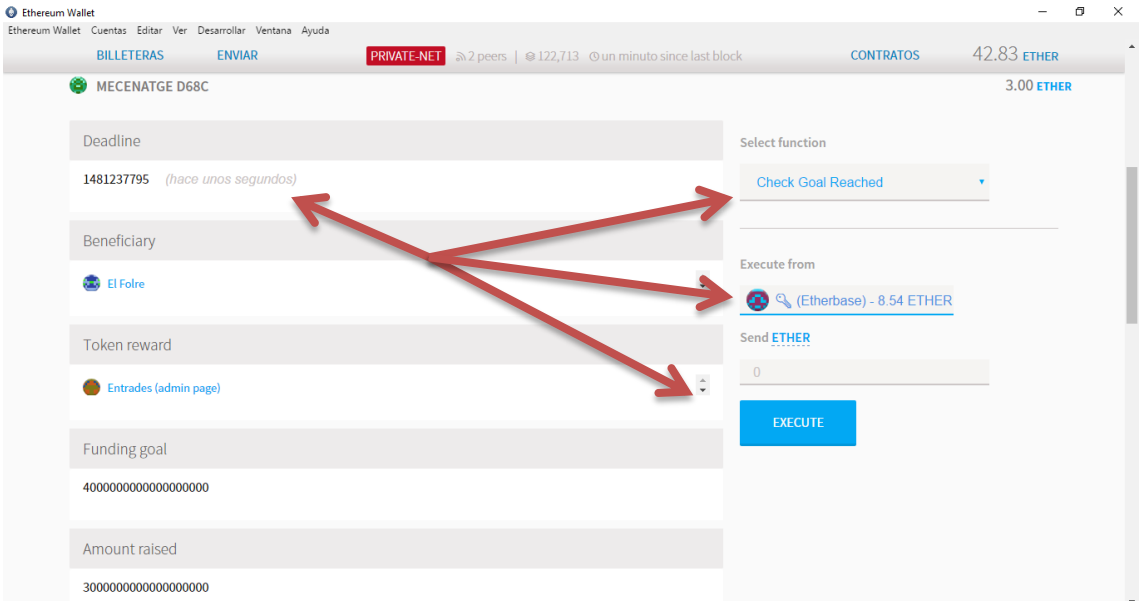

Figura 54 : Execució de la funció "checkGoalRaised".

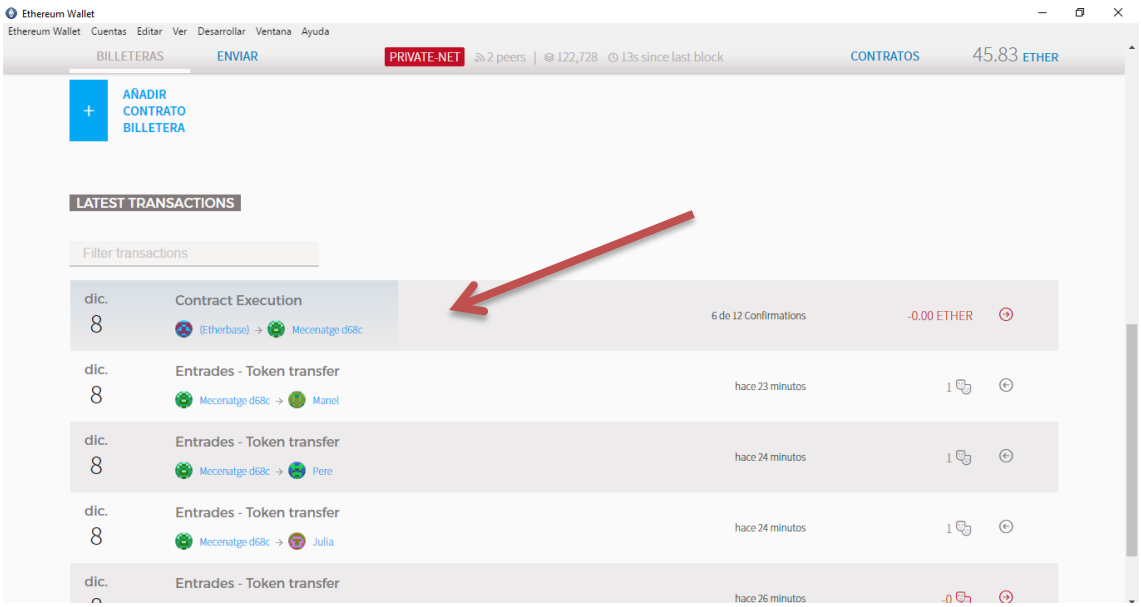

Figura 55 : Llistat transaccions en procés.

Seguidament verifiquem els resultats del mètode en el llistat dels contractes, revisem els events del contracte i la situació final del wallet per validar que el resultat ha estat correcte i s'han retornat les donacions realitzades a cada mecenes..

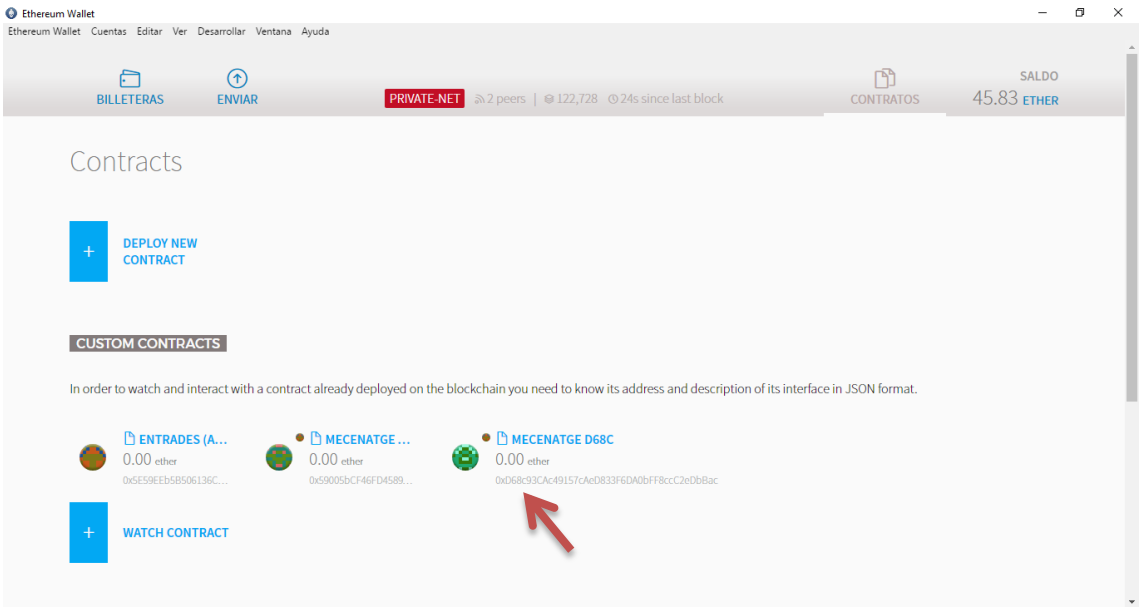

Figura 56 : Llistat de contractes del wallet

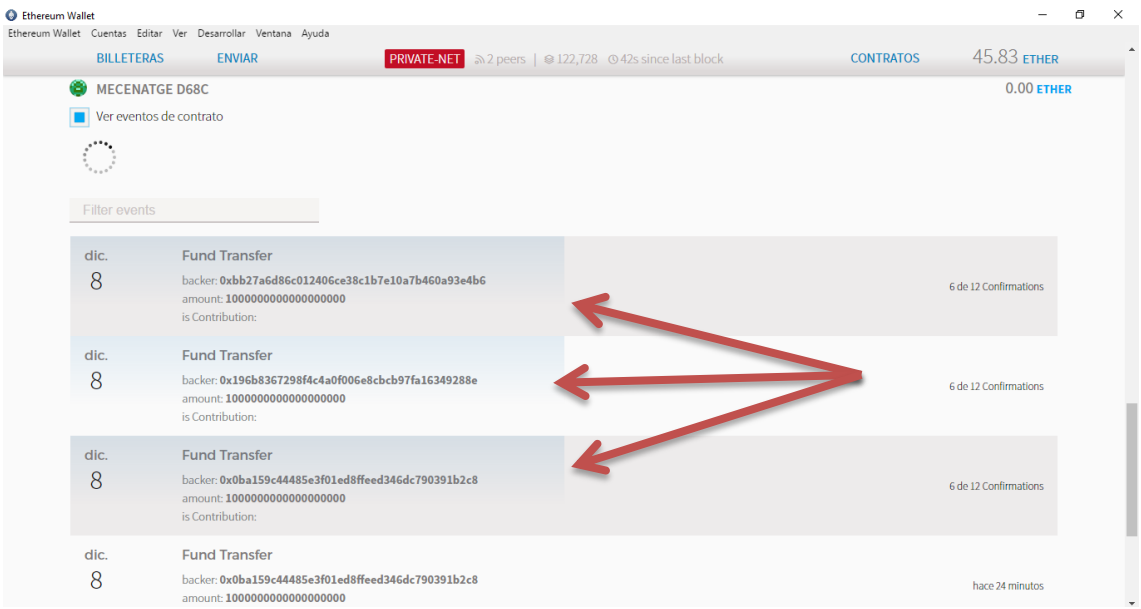

Figura 57 : Llistat dels "events" del nou contracte Mecenatge.

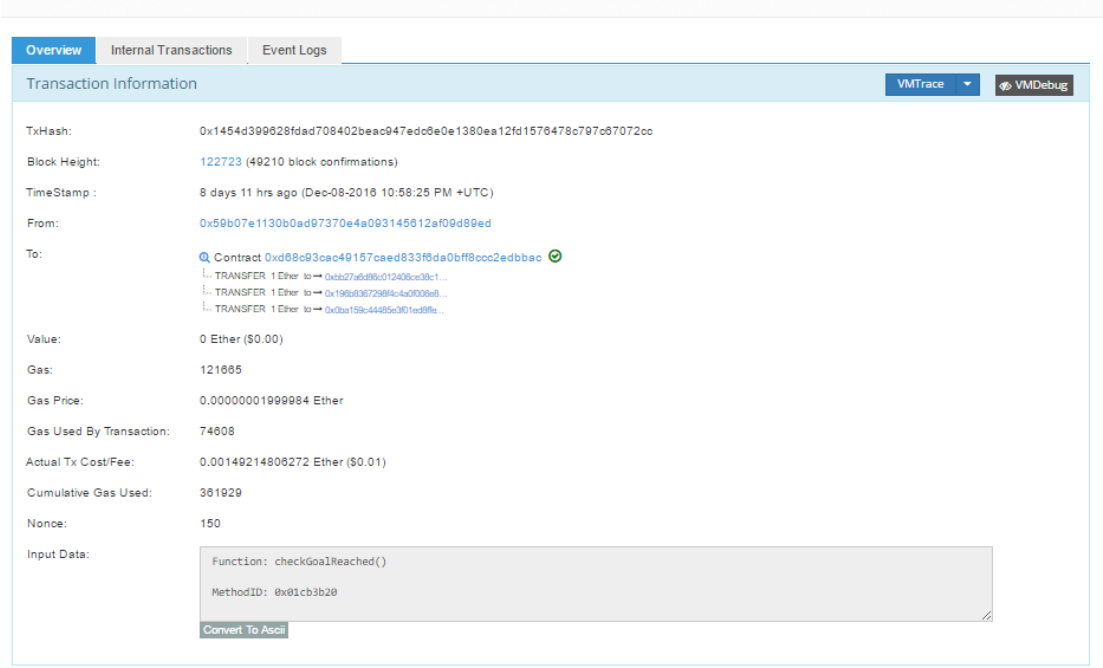

Figura 58 : Verificació de la transacció a Etherscan

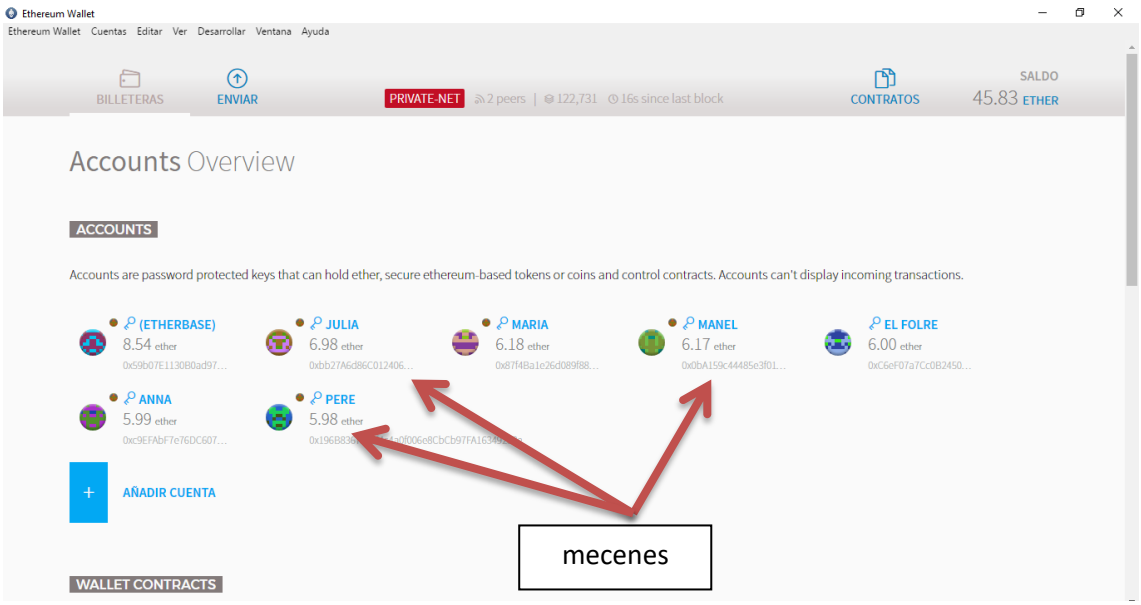

Figura 59 : Situació final del wallet.

Els resultats són els esperats i es retorna de forma automàtica l'import de les donacions als mecenes, es retorna un ether a Pere, al Manel i a la Julia i així es relecteix en el balanç dels seus comptes.

També realitzem la verificació que un cop tencat el contracte ja no es poden realitzar noves aportacions al cotracte.

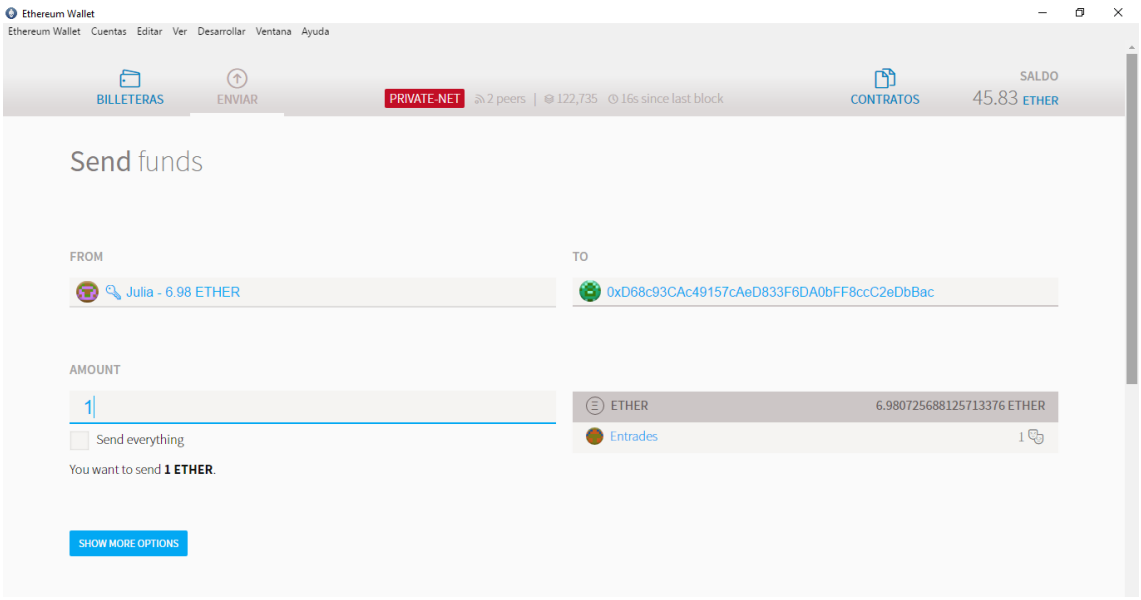

Figura 60 : Enviament de nova aportació al contracte.

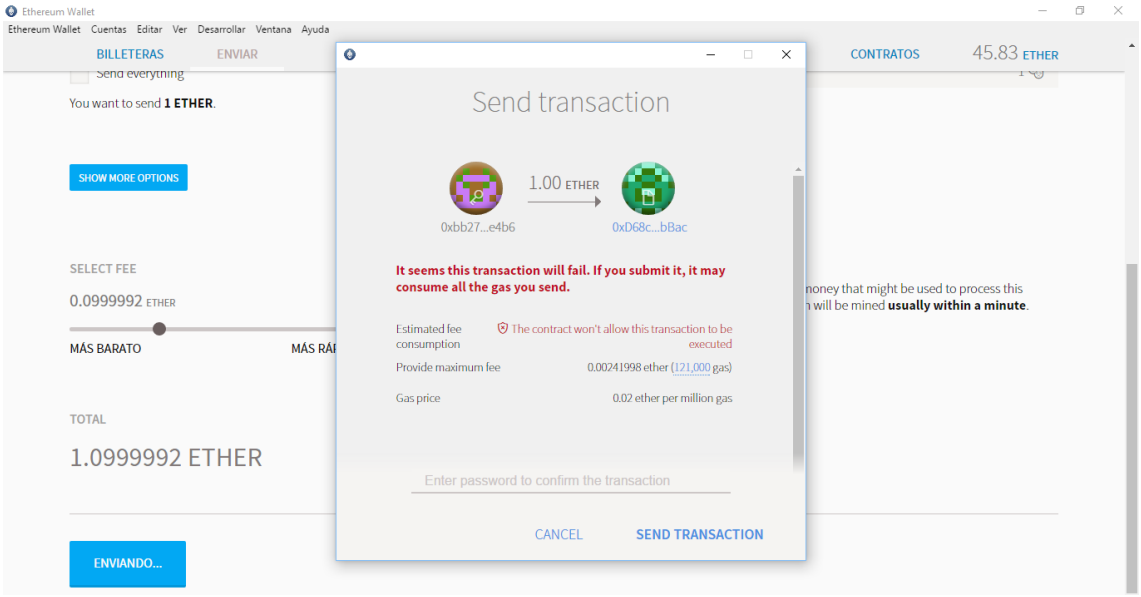

Figura 61 : Transacció del nou enviament de fons.

Es verifica que un cop tancat el contracte amb el mètode “checkGoalRaised” ja no es pot realitzar noves aportacions al contracte quedant definitivament tancat.

8-Conclusions.

El primer objectiu del projecte era investigar, conèixer i comprendre el funcionament dels Smart Contracts o Contractes intel·ligents i els seus usos més habituals en la actualitat. Aquest objectiu s'ha assolit principalment en el capítol 1 quan s'ha analitzat la situació actual dels contractes intel·ligents.

El segon objectiu del projecte era conèixer el funcionament de la plataforma Ethereum i de la tecnologia Blockchain o cadena de blocs. Aquest objectiu s'ha assolit entre els capítols 2 i 3 on s'ha començat a estudiar el funcionament del Sistema Bitcoin per seguidament veure com vol complementar i ampliar les funcionalitats de Bitcoin la plataforma Ethereum.

El tercer objectiu era implementar un petit Smart Contract de micromecenatge sobre la plataforma de Ethereum. Aquest objectiu s'ha assolit entre els capítols 4,5 i 6. Concretament, en el capítol 4 hem fet una immersió en el llenguatge Solidity que és el llenguatge que utilitza Ethereum per desenvolupar els contractes intel·ligents. En els capítols 5 s'ha explicat com serà el disseny del contracte, s'han definit un possible escenari i els casos d'ús que s'espera del contracte. Per finalitzar en el capítol 6 s'ha documentat la implementació del contracte en la plataforma Ethereum a traves del seu wallet Mist.

El quart objectiu era estudiar el comportament del contracte intel·ligent. Aquest objectiu s'ha assolit en el capítol 7 on s'han verificat les funcionalitats, casos d'ús, que es pretenien assolir i que estaven descrites en el capítol 5.

Per tant, considerem que hem assolit tots els objectius planificats a l'inici del projecte i els resultats de les proves realitzades en el capítol 7 han estat satisfactoris. Tot i això, desenvolupant el contracte intel·ligent s'han vist més possibilitats d'automatització de les clàusules del contracte i noves funcionalitats que es descriuran en el capítol següent.

De l'experiència que s'ha assolit en el desenvolupament d'aquest projecte destacaria com a punt fort de la plataforma Ethereum la interfície en la que interactuen els usuaris amb la cadena de blocs. El wallet Mist de Ethereum resulta molt visual e intuïtiu, això fa que la corba d'aprenentatge de la interfície sigui molt ràpida. Un segon punt fort de la plataforma és el llenguatge Solidity que és Turing-complert i molt similar al JavaScript fet que permet desenvolupar pràcticament totes les funcionalitats que es necessiten en un contracte intel·ligent o altres aplicacions com ara jocs d'atzar, contractes financers, magatzem de dades entre altres.

Com aspecte negatiu cal ressaltar la lentitud en que es descàrrega la replica de la cadena de blocs el primer cop en el node local, tot i tenir una bona connexió a internet es pot trigar de l'ordre de dos o tres dies en la descàrrega completa.

9- Futures línies de treball.

En aquest capítol pretenem enumerar possibles millores que he anat visualitzant i que es podrien aplicar en el contracte de micromecenatge implementat en aquest projecte.

Una millora estaria vinculada en la capacitat de reutilització del contracte. En el contracte que hem desenvolupat en aquest projecte un cop s'ha finalitzat la seva vida executant el mètode CheckGoalReached i s'han transferit els fons recaptats al compte del beneficiari o s'han retornat les donacions fetes a cada mecenes, el contracte ja no té cap nova funcionalitat amb el que es perd tota nova possibilitat d'interactuar amb ell.

Per tal d'aconseguir una reutilització del contracte es proposa les següents modificacions:

- La primera seria crear un nou mapping anomenat Campaign on s'anirien emmagatzemant campanyes de micromecenatge. La clau d'aquest nou mapping seria un enter o un string per diferenciar una campanya d'una altra i el valor seria una tupla on s'inclourien els atributs que hem definit en aquest projecte per una campanya en concret: beneficiary, fundingGoal, amountRaised, deadLine, price, tokenReward i el mapping Funders per portar el registre dels mecenes que han col·laborat en una campanya de mecenatge.

- La segona és que necessari un mètode constructor per les campanyes de mecenatge que es vulguin anar creant.

- La tercera es que també serà necessari modificar les funcions o mètodes del contracte, concretament la funció que s'utilitzen els mecenes per fer donacions i la funció CheckGoalRaised per poder finalitzar un projecte de micromecenatge. La modificació seria relativament senzilla ja que consistiria en referenciar aquestes funcions a una campanya en concret.

D'aquesta forma el contracte Mecenatge es podria continuar reutilitzant per diferents campanyes de mecenatge ja que solament es tindria de crear una nova campanya e iniciar el procés que s'ha descrit en aquest projecte. Aquest sistema també permetria que es pugues portar a terme més d'una campanya de mecenatge de forma concurrent ja que cada cop que es realitzes una donació o es tanques un contracte de mecenatge es referenciaria a una campanya en concret sense interferir en la resta de campanyes.

Aquest nou contracte podria ser útil per organitzacions que volguessin oferir un servei de planificació e implementació de projectes de mecenatge pels seus associats o terceres parts interessades en un projecte de mecenatge.

10-Referències.

[1] Dean Walsh , «A Beginner's Guide to Smart Contracts,» [En línia]. Disponible: http://cryptorials.io/a-beginners-guide-to-smart-contracts/. [Últim accés: 09 10 2016].

[2] BBVA Research, « Smart Contracts: ¿lo último en automatización de la confianza?,» [En línia]. Disponible: https://www.bbvaresearch.com/wp-content/uploads/2015/10/Situacion_Ec_Digital_Oct15_Cap1.pdf . [Últim accés: 09 10 2016].

[3] V. Frias Garcia, « Smart Contracts, ¿el futuro del blockchain?,» [En línia]. Disponible: http://www.intelygenz.es/smart-contracts-el-futuro-del-blockchain/ . [Últim accés: 10 10 2016].

[4] Wikipedia, « Contrato Inteligente,» [En línia]. Disponible: https://es.wikipedia.org/wiki/Contrato_inteligente. [Últim accés: 10 10 2016].

[5] L. Anderson, « New kids on the block: an analysis of modern blockchains [En línia]. Disponible: https://arxiv.org/pdf/1606.06530.pdf. [Últim accés: 23 10 2016].

[6] Andreas M. Antonopoulos. Mastering Bitcoin. Editorial O'Reilly (Primera edicio), 2014

[7] Pràctica 3 assignatura de Criptografia 05.601 de la UOC 2013-2014

[8] Vitalik Buterin, « A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM» [En línia]. Disponible: http://blog.lavoiedubitcoin.info/public/Bibliotheque/EthereumWhitePaper.pdf. [Últim accés: 29 10 2016].

[9] Gavin Wood, « ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER » [En línia]. Disponible: http://gavwood.com/paper.pdf. [Últim accés: 30 10 2016].

[10] Nick Szabo, « Formalizing and Securing Relationships on Public Networks » [En línia]. Disponible: http://firstmonday.org/ojs/index.php/fm/article/view/548/469. [Últim accés: 30 10 2016].

[11] OroyFinanzas.com, « La primera Organización Autónoma Descentralizada (DAO) de Ethereum ha recaudado ya 33 millones de dolares » [En línia]. Disponible: https://www.oroyfinanzas.com/2016/05/primera-organizacion-autonoma-descentralizada-dao-ethereum-recaudado-33-millones-dolares/. [Últim accés: 30 10 2016].

[12] Jose Felip Darás, « Bitcoin vs >Etheerum» [En línia]. Disponible: https://cointelegraph.es/news/bitcoin-vs-ethereum. [Últim accés: 30 10 2016].

[13] Satoshi Nakamoto, « Bitcoin: A Peer-to-Peer Electronic Cash System. » Disponible: https://bitcoin.org/bitcoin.pdf. [Últim accés 30-10-2016].

[14] Shamari Feaster, « Patricia Tree. » Disponible: https://github.com/ethereum/wiki/wiki/Patricia-Tree . [Últim accés 23-11-2016].

[15] Ethan Wilding, « Understanding the ethereum trie» Disponible: https://easythereentropy.wordpress.com/2014/06/04/understanding-the-ethereum-trie. [Últim accés 23-11-2016].

[16] BitcoinWiki, « Script» Disponible: https://en.bitcoin.it/wiki/Script. [Últim accés 23-11-2016].

[17] Ethereum builder's guides, « Solidity tutorials» Disponible: https://ethereumbuilders.gitbooks.io/guide/content/en/solidity_tutorials.html. [Últim accés 06-12-2016].

[18] Solidity — Solidity 0.4.8-develop documentation « Solidity» Disponible: https://solidity.readthedocs.io/en/develop/ [Últim accés 08-12-2016].

[19] Solidity - « Solidity style-guide» Disponoble:https://solidity.readthedocs.io/en/develop/style-guide.html [Últim accés 08-12-2016].

[20] Ethereum- «Ethereum Project» Disponoble: https://www.ethereum.org/token [Últim accés 08-12-2016].

[21] KONSTANTINOS CHRISTIDIS- « Blockchains and Smart Contracts for the Internet of Things» Disponoble: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7467408 [Últim accés 21-12-2016].

[22] Dr. GAVIN WOOD- « ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER » Disponoble: http://bravenewcoin.com/assets/Whitepapers/Ethereum-A-Secure-Decentralised-Generalised-Transaction-Ledger-Yellow-Paper.pdf [Últim accés 21-12-2016].

11-Annexos.

Annex I: Codi font.

Tot el codi font desenvolupat en llenguatge Solidity per a satisfer les necessitats d'aquest projecte, incloent-hi els contractes intel·ligents: Entrades i Mecenatge , es poden trobar en el següent repositori GitHub.

https://github.com/PereBS/pbertran_Codi_TFG_012017

Annex II: llicències d'ús.

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. **License grant**.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. **Other rights**.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. **Attribution**.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.
2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.
4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.
Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:
a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;
b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and
c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.
For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.
Section 5 – Disclaimer of Warranties and Limitation of Liability.
a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.
b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.
c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.
a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.
b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.
c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.
d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.
a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.
b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.
a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.
b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.
c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.
d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authorit