Grau d'Enginyeria Informàtica

Treball Final de Grau

Desenvolupament d'una aplicació descentralitzada

Alumne: Daniel Atienza Lopez

Consultor: Félix Freitag

10 de juny de 2018

GNU Free Documentation License (GNU FDL)

FITXA DEL TREBALL FINAL

Títol del treball:	*Desenvolupament d'una aplicació descentralitzada.*
Nom de l'autor:	*Daniel Atienza Lopez*
Nom del consultor:	*Félix Freitag*
Data de lliurament (mm/aaaa):	*06/2018*
Àrea del Treball Final:	*Aplicacions i sistemes distribuïts*
Titulació:	*Grau d'Enginyeria Informàtica (Computació)*

Resum del Treball (màxim 250 paraules):

Aquest treball final de grau té com a propòsit donar una visió de la tecnologia blockchain i les aplicacions descentralitzades, i posteriorment amb aquest coneixement crear una aplicació totalment descentralitzada per a la gestió dels certificats acadèmics.

En la descripció que es fa de la tecnologia blockchain, també es mostra diferents protocols com ara el Bitcoin, Ethereum i EOS. I sobre un d'aquests protocols es desenvoluparà l'aplicació descentralitzada.

Al final el lector podrà tenir un coneixement teòric i tècnic dels dos conceptes, que li pot permetre tenir una idea de com integrar un producte actual amb blockchain o bé crear-n'he un des de zero.

Abstract (in English, 250 words or less):

The present project has the purpose to give a vision about the blockchain technology and the decentralized applications in which I will use all these knowledges to create an application fully-decentralized to manage academic certifications.

In the description about the blockchain technology, I also show different protocols as Bitcoin, Ethereum and EOS. One of these protocols is used to create a decentralized application.

At the end, the reader could have a theoretical and technical knowledge about both concepts that can be useful to integrate a current product with the blockchain technology or make one from scratch.

Paraules clau (entre 4 i 8):

Blockchain, Aplicació descentralitzada, Ethereum, EOS, IPFS, Reactjs

Índex

Llista de figures

1. Introducció

1.1. Context i justificació del Treball

Aquest últims anys s'està parlant força dels projectes o companyies que funcionen de manera descentralitzada en les quals la lògica i les dades estan replicades per la xarxa, i que qualsevol persona pot instal·lar-se el programari per mantenir una copia de la informació.

Un dels primers projectes que es va crear funcionant de manera descentralitzada és el de la primera moneda digital anomenada *Bitcoin*, creada a partir d'un article [1] publicat l'any 2008 amb el títol *"Bitcoin: A Peer-to-Peer Electronic Cash System"* i signat amb el pseudònim de *Satoshi Nakamoto*.

La tecnologia que hi ha darrere va ser descrita l'any 1991, però va ser gràcies a l'article [2] publicat on es va conceptualitzar el terme *blockchain*. Aquesta tecnologia garanteix la integritat de les dades mitjançant la criptografia i no requereix d'intermediaris per validar i certificar la informació.

Després de la creació del *Bitcoin* s'han creat nous projectes per cobrir noves necessitats; alguns permeten crear aplicacions formades per regles programades i al mateix temps immutables.

Un exemple dels nous projecte creats és l'anomenat *Ethereum*, el qual va començar a funcionar l'any 2015 i ofereix la possibilitat de crear aplicacions anomenades *smart-contract* que poden ser executades des de qualsevol node de la xarxa.

Així doncs, la tecnologia *blockchain* obre un ventall de noves possibilitats alhora de crear noves aplicacions permeten prescindir d'intermediaris.

1.2. Objectius del Treball

Els objectius d'aquest treball són donar una visió de la tecnologia *blockchain* i de les aplicacions descentralitzades, i els passos i eines per el desenvolupament i desplegament d'una aplicació.

Descripció de la tecnologia *blockchain*: Es pretén donar una visió general de l'estat actual de la tecnologia *blockchain*, des de la seva creació fins a dia d'avui. Principalment s'investigarà i estudiarà els protocols Bitcoin, Ethereum i EOS.IO.

Descripció de les aplicacions descentralitzades: A partir de la recerca i investigació respecte la tecnologia blockchain s'introduirà el concepte i l'estat actual de les aplicacions descentralitzades. Que són? Com s'estructuren? Quina és la seva projecció a llarg termini?

Descripció de les tecnologies a utilitzar: En aquest objectiu es descriurà cada tecnologia triada (Ethereum, IPFS i Reactjs) per al desenvolupament del treball, i també es justificarà la seva tria.

Creació i desplegament de la dApp: En aquest punt, es detallaran els passos seguits i decisions preses des de l'anàlisi de la dApp fins al seu desenvolupament i desplegament.

Conclusions i futurs passos: En aquest últim punt es descriuran les conclusions extretes del treball i recerca realitzat i també possibles passos per la millora de l'aplicació desenvolupada.

1.3. Enfocament i mètode seguit

Tot i que el concepte de la tecnologia *blockchain* és nou per a moltes persones, durant tot el curs s'han estudiat algunes assignatures que ajuden a entendre el seu funcionament. Tenint en compte aquest coneixement i la manca d'informació dels projectes que es descriuran, se seguirà el següent mètode; planificació, recerca d'informació, desenvolupament i elaboració de la memòria.

És per aquest motiu que seguir l'ordre comentat suposa un avantatge ja que, a mesura que s'avanci, sempre es disposarà del coneixement necessari.

1.4. Planificació del Treball

Les tasques planificades són les següents:

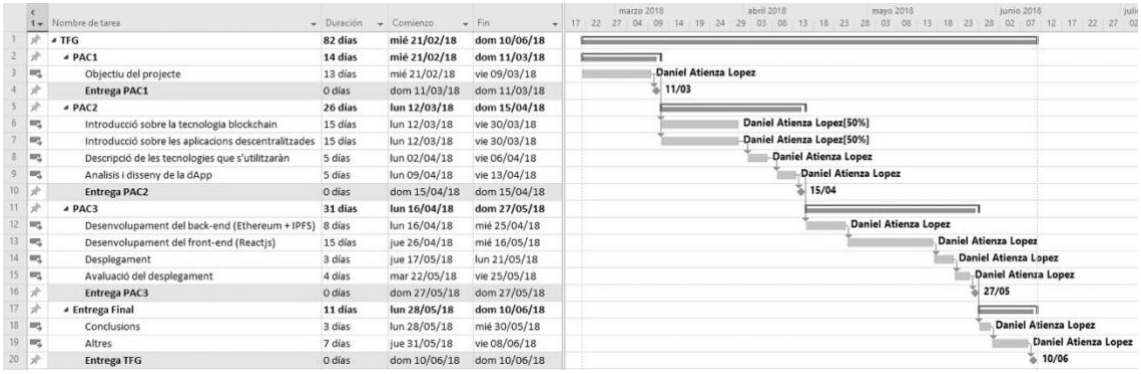
- **Objectiu del projecte:** Preparació del document Pla de Treball amb la descripció del treball final de grau, els objectius i la planificació de tasques. En finalitzar aquesta tasca s'obté el document de seguiment PAC 1.
- **Introducció sobre la tecnologia *blockchain*:** Cerca i estudi de la tecnologia *blockchain* des del seu inici fins a l'estat actual. Aquesta tasca ha d'ajudar a obtenir una visió global de la tecnologia: origen, motius pels quals es va crear, avantatges i inconvenients, exemples de projectes...
- **Introducció sobre les aplicacions descentralitzades:** Cerca i estudi del concepte de les aplicacions descentralitzades. El resultat ha d'aportar informació del concepte i com s'està enfocant cara el futur.
- **Descripció de les tecnologies que s'utilitzaran:** Cerca i estudi de les tecnologies que s'utilitzaran per al desenvolupament de l'aplicació descentralitzada. El resultat de la tasca ha d'aportar informació de les tecnologies i els motius pel quals s'ha triat cadascuna.
- **Anàlisis i disseny de l'aplicació descentralitzada *(dApp)*:** Descripció de les funcionalitats, els mètodes, comunicació entre les diferents tecnologies... La informació obtinguda en aquesta tasca s'utilitzarà per dur a terme el desenvolupament de l'aplicació.
- **Entorn de treball:** Descripció de les eines necessàries per a poder treballar de manera local. Com a resultat, s'obtindrà la informació per la creació pas a pas del entorn de treball.
- **Desenvolupament del *back-end* (Ethereum + IPFS):** Desenvolupament del smart-contract i la integració amb el sistema de fitxers IPFS. Al final d'aquesta tasca el *back-end* quedarà operatiu per tal què el *front-end* hi pugui interactuar.
- **Desenvolupament del *front-end* (Reactjs):** Per a l'execució d'aquesta tasca és imprescindible l'anterior tasca (7) ja que el *front-end* ha de comunicar-se amb el *back-end*. El resultat de la tasca serà l'aplicació funcionant en l'entorn local.
- **Desplegament:** Desplegament de l'aplicació a Internet, on el *back-end* s'implementarà en la xarxa de proves *Rinkeby* d'*Ethereum* i el *front-end* quedarà allotjat en el sistema de fitxers IPFS. Com a resultat s'obtindrà l'aplicació accessible per a qualsevol usuari d'Internet.
- **Avaluació del desplegament:** Avaluació del desplegament en l'entorn local i productiu (xarxa de proves). El resultat és aportar detalls per a les conclusions.
- **Conclusions:** Descripció de les conclusions del que proporciona la tecnologia *blockchain* i les aplicacions descentralitzades a partir del coneixement i experiència obtinguda.
- **Altres:** Aquesta última tasca engloba la elaboració de la memòria del treball, el vídeo de presentació, l'informe d'autoavaluació i qualsevol material a entregar.

Les tasques descrites es poden veure en la Figura 1 agrupades per les entregues que s'han de realitzar durant el semestre. Cada entrega correspon al mètode descrit en el punt anterior que són:

- **PAC1:** Elaboració de la descripció i planificació del treball.
- **PAC2:** Recerca d'informació.
- **PAC3:** Desenvolupament de l'aplicació.
- **Entrega Final:** Preparació del material a entregar (memòria, vídeo, ...).

En el diagrama s'ha configurat un recurs amb una dedicació de 8 hores els dies laborals, i els dies de les entregues que coincideix amb diumenge s'ha creat una excepció passant a ser laborable.

Figura 1: Diagrama de Gantt



	Nombre de tarea	Duración	Comienzo	Fin
1	TFG	82 días	mié 21/02/18	dom 10/06/18
2	PAC1	14 días	mié 21/02/18	dom 11/03/18
3	Objectiu del projecte	13 días	mié 21/02/18	vie 09/03/18
4	Entrega PAC1	0 días	dom 11/03/18	dom 11/03/18
5	PAC2	26 días	lun 12/03/18	dom 15/04/18
6	Introducció sobre la tecnologia blockchain	15 días	lun 12/03/18	vie 30/03/18
7	Introducció sobre les aplicacions descentralitzades	15 días	lun 12/03/18	vie 30/03/18
8	Descripció de les tecnologies que s'utilitzaràn	5 días	lun 02/04/18	vie 06/04/18
9	Analisis i disseny de la dApp	5 días	lun 09/04/18	vie 13/04/18
10	Entrega PAC2	0 días	dom 15/04/18	dom 15/04/18
11	PAC3	31 días	lun 16/04/18	dom 27/05/18
12	Desenvolupament del back-end (Ethereum + IPFS)	8 días	lun 16/04/18	mié 25/04/18
13	Desenvolupament del front-end (Reactjs)	15 días	jue 26/04/18	mié 16/05/18
14	Desplegament	3 días	jue 17/05/18	lun 21/05/18
15	Avaluació del desplegament	4 días	mar 22/05/18	vie 25/05/18
16	Entrega PAC3	0 días	dom 27/05/18	dom 27/05/18
17	Entrega Final	11 días	lun 28/05/18	dom 10/06/18
18	Conclusions	3 días	lun 28/05/18	mié 30/05/18
19	Altres	7 días	jue 31/05/18	vie 08/06/18
20	Entrega TFG	0 días	dom 10/06/18	dom 10/06/18

Font: Elaboració pròpia

1.5. Breu sumari de productes obtinguts

- **Memòria:** És l'actual document on presenta tota la documentació teòrica i tècnica.
- **PACs de seguiment:** Documents en els quals es pot visualitzar el pla de treball i com s'ha avançat durant tot el semestre.
- **Aplicació:** És la part lògica del *front-end* de l'aplicació. En el desplegament a Internet s'allotjarà en el protocol IPFS.
- **Contracte intel·ligent:** Part lògica del *back-end*, que es situarà en la xarxa d'Ethereum.

1.6. Breu descripció dels altres capítols de la memòria

Els pròxim capítols que es trobaran són els següents:

- **Introducció a la tecnologia blockchain:** En aquest apartat es proporciona una visió actual de la tecnologia i del ritme amb què avança.
- **Introducció a les aplicacions descentralitzades:** Visió global del que són les aplicacions descentralitzades i dels seus avantatges i inconvenients.
- **Introducció a l'aplicació descentralitzada:** Anàlisi de l'aplicació que es vol crear i amb quines tecnologies.
- **Desenvolupament:** Descriu el desenvolupament de l'aplicació de manera detallada i també del desplegament en un entorn local i en Internet.
- **Conclusions:** Presenta la conclusió final del desenvolupament amb els problemes que s'han tingut i les millores que es podrien realitzar.

- **Bibliografia:** Mostra totes les fonts consultades per tal de dur a terme el treball final de grau.
- **Annexos:** S'hi pot trobar informació complementària per al desenvolupament de l'aplicació. Entre aquesta informació hi ha els passos per a la instal·lació i configuració dels nodes d'Ethereum i d'IPFS.

Es pot veure que, primer, es descriu la part teòrica dels conceptes blockchain i de les aplicacions descentralitzades i, seguidament, la part pràctica on es desenvolupa l'aplicació.

2. Introducció a la tecnologia *blockchain*

2.1. Historia

En el punt 1.1 es feia referència que el concepte de la tecnologia *blockchain* va ser descrit en un article [2] publicat l'any 1991 per Stuart Haber i W. Scott Stornetta. Els autors plantejaven el problema de com certificar la data de creació o modificació de qualsevol document digital mantenint-ne la seva privacitat.

Un exemple que Stuart Haber i W. Scott Stornetta descriuen és el cas del registre d'una patent en què s'ha de deixar constància del dia que s'ha registrat ja que, en el cas que aparegués una d'igual, hi hauria evidències de quina havia estat registrada primer.

A partir de l'exemple, els autors van proposar com a solució un servei anomenat *"Time-stamping service (TSS)"* que consisteix en un depòsit digital segur on els usuaris envien el document a registrar i aquest guarda la data i hora que s'ha enviat i una còpia del document.

Tot i que amb la solució proposada quedaven temes per solucionar com ara; el document pot arribar corrupte, perd la privacitat, la seva mida pot ser un problema i el servei no garantia el marcatge de la data i hora indicada es va millorar fent us d'una funció *hash*[1] per evitar enviar el document i signar digitalment la informació processada per tal que el client pugui verificar-ne el registre.

Dos esquemes on es pot implementar la solució del servei descrit són el *Linking* i *Distributed trust*. La diferencia que més destaca entre aquests dos esquemes es que el *Distributed trust* està dissenyat per treballar totalment descentralitzat on cap entitat o persona pugui modificar cap dada i així mantenir la seva integritat.

Amb aquest concepte, els de la criptografia i d'altres va aparèixer l'any 2009 la primera moneda digital descentralitzada on cap entitat té el seu control, aquesta és anomenada Bitcoin. Posteriorment, han aparegut d'altres com per exemple Ethereum i EOS.IO que ofereixen noves funcionalitats com ara contractes intel·ligents i la possibilitat de crear aplicacions descentralitzades.

2.2. Concepte

El concepte del *blockchain* és concatenar cronològicament les dades que estan signades digitalment per l'autor i emmagatzemades en blocs, fent impossible qualsevol modificació ja que cada bloc està relacionat amb l'anterior per valors criptogràfics i al mateix temps distribuïts per tots els nodes de la xarxa. Aquesta estructura proporcionen característiques de seguretat com ara; integritat de les dades, autenticitat del remitent i el no repudi en l'origen i destinació.

Actualment aquest concepte del *blockchain* pot ser implementat de varies maneres. Hi ha casos que funciona de manera centralitzada, on el servei i manteniment està gestionat per una única entitat i d'altres totalment descentralitzat on qualsevol persona pot instal·lar el programari i contribuir a la creació de blocs.

[1] **Funció *hash*:** És un algoritme que disposa d'una entrada -normalment per introduir una cadena de text- que ho converteix retornant-ho en una cadena amb longitud finita. Aquestes funcions són unidireccional, és a dir, a partir de la sortida mai és pot saber el text d'entrada.

Totes les dades transmeses a la cadena de blocs poden estar disponibles públicament, però també es pot implementar de manera privada per a que només una sèrie de persones o servidors puguin tenir accés.

2.3. Funcionament

El funcionament que es descriu en el article [2] de Stuart Haber i W. Scott Stornetta fa les següents operacions alhora de registrar un document. Primer el client ha d'enviar el *hash* del document i un identificador numèric al servei TTS on aquest aplicarà la següent funció:

$$L_n = (n, t_n, ID_n, y_n;\ Hash(L_n))$$

El paràmetres que accepta són; *n* que és un nombre seqüencial, *t* és la data i hora, *ID* és l'identificador numèric proporcionat pel client, *y* el *hash* del document i *L* el resultat de l'última operació. Un cop processada la petició, el servei envia al client un certificat signat on pot verificar l'operació i l'identificador numèric que ha d'utilitzar en la següent petició.

En els projectes de les monedes criptogràfiques (Bitcoin, Ethereum, EOS.IO,..) funcionen de forma descentralitzada i els usuaris disposen d'una clau publica / privada amb la que signaran cada acció que realitzin anomenada transacció.

Les transaccions són agrupades en blocs que posteriorment es replicaran per tots els servidor connectats a la xarxa. Si el bloc i les transaccions són correctes, es concatenarà a la cadena de blocs en cas contrari es rebutja. Aquests blocs es generen automàticament cada *x* temps i de forma interrompuda, si en algun moment no existís cap transacció els blocs quedarien buits.

Cal comentar que quan un projecte comença a funcionar sempre ho fa amb un bloc inicial anomenat *genesis*, aquest conté paràmetres de configuració com per exemple les adreces que disposen d'alguna quantitat de monedes.

En la Taula 1 és mostra els cinc primers blocs que es van processar en Ethereum, on veu la relació que sempre hi ha amb el bloc anterior mitjançant el seu *hash -hash* pare-. El bloc 0 (Gensis) mai estarà relacionat amb cap bloc anterior ja que és l'inicial, per aquest motiu mostra 0x0.

Taula 1: Relació dels cinc primers blocs d'Ethereum

Cinc primers blocs d'Ethereum	
Bloc 0 (Genesis)	
Hash	0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3
Hash pare	0x0000000000000000000000000000000000000000000000000000000000000000
Bloc 1	
Hash	0x88e96d4537bea4d9c05d12549907b32561d3bf31f45aae734cdc119f13406cb6
Hash pare	0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3
Bloc 2	
Hash	0xb495a1d7e6663152ae92708da4843337b958146015a2802f4193a410044698c9
Hash pare	0x88e96d4537bea4d9c05d12549907b32561d3bf31f45aae734cdc119f13406cb6
Bloc 3	
Hash	0x3d6122660cc824376f11ee842f83addc3525e2dd6756b9bcf0affa6aa88cf741
Hash pare	0xb495a1d7e6663152ae92708da4843337b958146015a2802f4193a410044698c9
Bloc 4	
Hash	0x23adf5a3be0f5235b36941bcb29b62504278ec5b9cdfa277b992ba4a7a3cd3a2
Hash pare	0x3d6122660cc824376f11ee842f83addc3525e2dd6756b9bcf0affa6aa88cf741

Font: https://etherscan.io/blocks

Aquesta relació que existeix amb el bloc anterior incrementa la dificultat de manipular les transaccions. Per exemple a dia d'avui 15 d'abril de 2018 existeixen més de 5.000.000 de blocs en Ethereum, en cas de voler

modificar alguna transacció en bloc 150.000 s'haurien de manipular tots els blocs que hi ha a continuació i replicar els canvis a tots els servidors de la xarxa. Aquests canvis difícilment serien reproduïts a tots els servidors ja que s'hauria d'accedir de manera il·lícita i evitant totes les mesures de seguretat.

Si es volgués comprovar el bloc *genesis* que va processar la xarxa Ethereum, és pot accedir en el següent enllaç i comprovar que va ser el 30 de Juliol de 2015 on contenia 8893 transaccions. També al ser un projecte de codi lliure, aquesta informació es pot verificar descarregant els seu programari i tota la cadena de blocs. En la Figura 2 és mostra algunes de les transaccions del bloc *genesis* on es veu l'assignació de monedes.

Figura 2: Transaccions del bloc genesis d'Ethereum

TxHash	Block	Age	From	To	Value	[TxFee]
GENESIS_756f45e3...	0	990 days 20 mins ago	GENESIS	0x756f45e3fa69347...	200 Ether	0
GENESIS_f42f9052...	0	990 days 20 mins ago	GENESIS	0xf42f905231c770f...	197 Ether	0
GENESIS_2489ac1...	0	990 days 20 mins ago	GENESIS	0x2489ac126934d4...	1,000 Ether	0
GENESIS_ddf5810a...	0	990 days 20 mins ago	GENESIS	0xddf5810a0eb2fb2...	17,900 Ether	0
GENESIS_c951900...	0	990 days 20 mins ago	GENESIS	0xc951900c341abb...	327.6 Ether	0
GENESIS_6806408...	0	990 days 20 mins ago	GENESIS	0x680640838bd07a...	1,730 Ether	0
GENESIS_9d0f347e...	0	990 days 20 mins ago	GENESIS	0x9d0f347e826b7d...	4,000 Ether	0
GENESIS_9328d55...	0	990 days 20 mins ago	GENESIS	0x9328d55ccb3fce5...	4,000 Ether	0
GENESIS_7e7f18a0...	0	990 days 20 mins ago	GENESIS	0x7e7f18a02eccaa5...	66.85 Ether	0
GENESIS_3c869c0...	0	990 days 20 mins ago	GENESIS	0x3c869c09696523...	1,000 Ether	0
GENESIS_551e778...	0	990 days 20 mins ago	GENESIS	0x551e7784778ef8e...	600 Ether	0
GENESIS_f0c081da...	0	990 days 20 mins ago	GENESIS	0xf0c081da52a9ae3...	111 Ether	0
GENESIS_cf888235...	0	990 days 20 mins ago	GENESIS	0xcf8882359c0fb23...	6,000 Ether	0
GENESIS_457bcef3...	0	990 days 20 mins ago	GENESIS	0x457bcef37dd3d6...	20 Ether	0
GENESIS_562105e...	0	990 days 20 mins ago	GENESIS	0x562105e82b0997...	6,000 Ether	0
GENESIS_42d3494...	0	990 days 20 mins ago	GENESIS	0x42d34940edd2e7...	158 Ether	0

Font: https://etherscan.io/txs?block=0

Tots tres projectes -Bitcoin, Ethereum i EOS.IO- comentats en aquest treball, són *blockchain* públics permetent que qualsevol persona pugui consultar i veure totes les transaccions processades fins al dia d'avui. Aquesta informació també es pot consultar mitjançant pagines disponibles a Internet anomenades *block explorer* o bé descarregant en local tot el seu programari.

2.4. Seguretat

La seguretat en aquests projectes és molt important ja que qualsevol indici de manipulació en les dades és pot perdre la confiança en el sistema. A part de la cadena de blocs i les claus publica / privada per signar les transaccions, els projectes de les monedes criptogràfiques que treballen de manera descentralitzada utilitzen mètodes de consens per acceptar o rebutjar blocs ja que qualsevol persona pot afegir un servidor a la xarxa per processar transaccions.

Tot i que existeixen diferents mètodes de consens, en aquest treball només faré menció dels mètodes utilitzat Bitcoin, Ethereum i EOS.IO.

En el cas de Bitcoin i Ethereum utilitzen el mètode *proof-of-work* que consisteix en la resolució d'un algorisme per poder afegir blocs. En aquest cas com més capacitat de processament es disposi més probabilitat té entre tots els servidors per afegir el bloc. En aquest mètode si es volgués tenir el control total de les transaccions de la xarxa cal disposar del 51% del processament total de la xarxa per a que l'altre part validi els possibles blocs amb transaccions alterades.

Pel que fa EOS.IO utilitza una variant del *proof-of-stake* anomenada *delegate proof-of-stake*. El mètode *proof-of-stake* estableix que el servidor que vulgui processar transaccions i generar blocs cal que disposi de

monedes -en aquest cas de EOS.IO-, així doncs com més monedes disposi més probabilitat tindrà per ser seleccionat en la creació de nous blocs. En la variant *delegate proof-of-stake* cal que els usuaris de la xarxa votin per un nombre de servidors per a que processin les transaccions on en el cas d'EOS.IO el nombre establert és de 21 servidors, la resta queden en una llista d'espera .

En aquest últim mètode, si algun servidor seleccionat envies un bloc amb alguna transacció alterada serà penalitzat sent eliminat de la llista i en alguns casos congelant totes les monedes que disposa. Un cop eliminat, es seleccionarà el següent de la llista.

Per últim, comentar que els servidor que treballen per mantenir la xarxa activa i segura són recompensats amb una quantitat de monedes cada cop que concatenen un bloc a la cadena de blocs.

2.5. Projectes

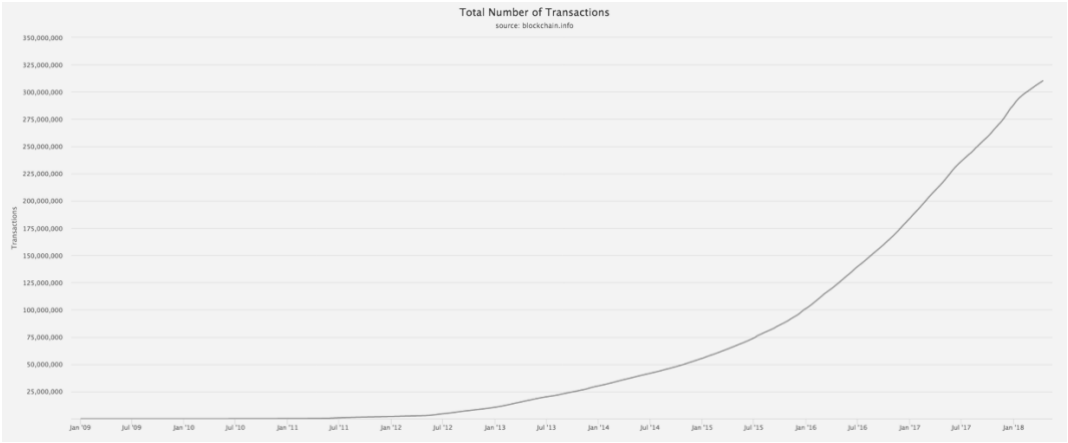
2.5.1. Bitcoin

Bitcoin va ser la primera moneda digital descentralitzada, i va néixer amb el propòsit d'oferir als usuaris un nou mitja de pagament que no depengués de cap entitat (p.ex. govern, bancs i entitats financeres) i proporcionar transferències rapides sense cap intermediari.

La seva infraestructura porta activa des de l'any 2009 i els dos anteriors mesos, març i abril de 2018, ha processat una mitjana de ~170.000 transaccions al dia. En la Figura 3 és pot veure com en els últims anys l'ús de la moneda a augmentat.

Tot i que l'ús principal que es fa del Bitcoin són pagaments, també ofereix la possibilitat de desenvolupar petits *scripts* [3] per automatitzar les transaccions al produir-se un esdeveniment.

Figura 3: Gràfic amb el nombre total de transaccions de Bitcoin



Font: https://blockchain.info/charts/n-transactions-total?timespan=all

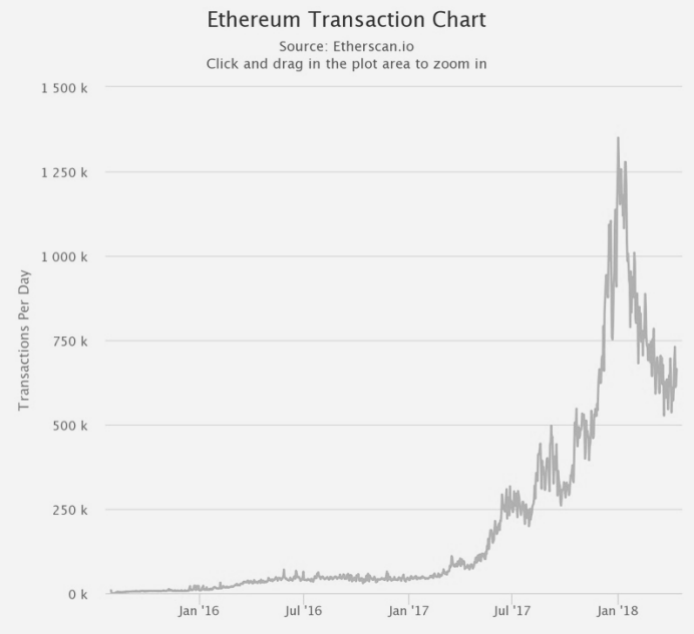
2.5.2. Ethereum

Ethereum va aparèixer l'any 2015 i algunes persones el defineixen com la versió 2.0 de Bitcoin ja que ofereix una nova funcionalitat anomenada *smart-contracts*. El *smart-contracts* que en català s'anomena contractes intel·ligents, són petits programes on a partir d'una sèrie de regles definides per dos o més parts es pot automatitzar l'acció d'algun fet al complir-se alguna de les regles.

Aquests *smart-contracts* han de ser desenvolupats utilitzant amb el llenguatge Solidity, a diferencia dels scripts que ofereix Bitcoin en aquest cas el llenguatge ofereix més funcionalitats alhora de programar. Un cop desenvolupat un contracte s'envia a la cadena de blocs i passa a ser un programa immutable que no podrà ser mai modificat. Cal destacar que al enviar un contracte a la blockchain de Ethereum s'ha de pagar una comissió per a que els servidors processin la petició, igual succeeix al realitzar una transacció.

El nombre de transaccions que s'han processat els últims dos mesos -març i abril de 2018- ha estat de ~600.000 transaccions de mitjana al dia. La Figura 4 mostra la seva evolució i com s'ha incrementat el seu ús respecte l'any 2017.

Figura 4: Gràfic amb el nombre de transaccions / dia d'Ethereum



Font: https://etherscan.io/chart/tx

2.5.3. EOS.IO

EOS.IO és un projecte que es troba en ple desenvolupament i es preveu que comenci a funcionar al juny d'aquest any 2018, així que la següent informació és teòrica i extreta del article [4] publicat.

Aquest projecte és presenta amb la característica d'oferir un sistema on poder desenvolupar aplicacions descentralitzades sobre aquest blockchain. Les característiques que més destaquen són: us de base de dades, diferents permisos sobre un compte d'usuari, comunicació asíncrona i el processament de transaccions pot escalar fins a 1.000.000 de transaccions per segon.

En aquest cas, els usuaris les transaccions per interactuar amb les aplicacions descentralitzades no tenen comissió però si es vol desenvolupar una aplicació sobre aquest protocol cal disposar de *tokens* per tenir un percentatge de computació de tota la xarxa.

2.6. Taula comparativa dels protocols descrits

	Bitcoin	Ethereum	EOS.IO
Estat	Actiu	Actiu	En desenvolupament
Consens	Proof-of-Work	Proof-of-Work	Delegate Proof-of-Stake
Data del primer bloc	03/01/2009	30/07/2015	*01/06/2018*
Creació dels blocs	10 minuts	14 – 15 segons de mitja	500 mil·lisegons (0.5s)
Nombre de blocs	518.489	5.451.771	-
Mida del blockchain	164 Gb	65 Gb	-
Transaccions per segon	~7	~25	>100.000
Característica principal	Moneda digital descentralitzada	Desenvolupament de smart-contracts	Desenvolupament d'aplicacions descentralitzades
Llenguatge de programació	C++	Go, C++ i Rust	C++
Llenguatge dels smart-contracts		Solidity	Webassembly
Pàgina oficial	https://www.bitcoin.org	https://www.ethereum.org	https://www.eos.io

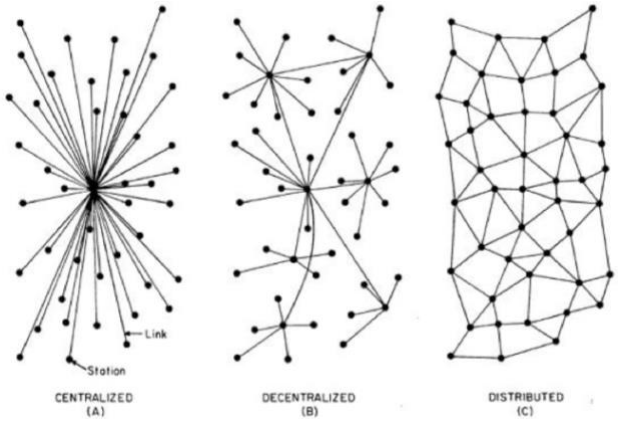
** Les dades referents al nombre de blocs i mida del blockchain s'han obtingut el dia 17/04/2018*

3. Introducció a les aplicacions descentralitzades

3.1. Introducció

Alhora de dissenyar una aplicació existeixen diferents models (Figura 5) com ara; el centralitzat, el descentralitzat i el distribuït. El centralitzat és caracteritza per concentrar tot el processament en un sol servidor, un model descentralitzat permet distribuir la carrega de treball en més d'un servidor i el distribuït treballa de manera coordinada amb els altres servidors.

Figura 5: Diferents models d'aplicacions



Font: https://www.safaribooksonline.com/library/view/decentralized-applications/9781491924532/

Les aplicacions descentralitzades cada cop tenen més ressò des de l'aparició de Bitcoin ja que aquest model ofereix forces avantatges com per exemple funcionar de manera autònoma i sense cap intermediari, també són difícilment censurables i ofereixen una alta disponibilitat.

En el sector del blockchain les aplicacions descentralitzades poden ser creades de diferents maneres; des de cero (p.ex. Bitcoin, Ethereum, EOS.IO, ...) , a partir d'un altre protocol de codi lliure *(fork)*, o bé utilitzant un protocol ja existent que ho permeti (p.ex. Ethereum o EOS.IO).

Com s'ha comentat al inici d'aquest treball, el desenvolupament de l'aplicació descentralitzada per aquest treball es centra en el protocol Ethereum pel desenvolupament del *back-end*[2] mitjançant un *smart-contract*.

3.2. Funcionament

Al desenvolupar un *smart-contract* per a que funcioni sobre el protocol Ethereum, només s'ha de crear la lògica del *smart-contract* utilitzant el llenguatge de programació Solidity i posteriorment publicar-ho a la xarxa.

[2] back-end: Lògica de l'aplicació situada en la part del servidor.

Un cop publicat el contracte, els nodes d'Ethereum es sincronitzen per disposar de l'últim bloc i així garantir l'accés al contracte des de qualsevol node. L'accés és pot realitzar mitjançant el client oficial de Ethereum (consola o aplicació), una interfície preparada per l'aplicació o bé fent us d'algun *block explorer* que ho permeti.

3.3. Avantatges i inconvenients

Els principals avantatges de les aplicacions descentralitzades és la seguretat que ofereix en garantir la integritat i autenticitat de les dades gràcies al us de les cadenes de blocs i les claus publiques / privades. També la transparència al disposar del codi font de l'aplicació i l'alta disponibilitat al estar replicada per diferents nodes.

Pel que fa els inconvenients, s'ha de tenir en compte que desenvolupar un *smart-contract* un cop s'ha publicat ja no podrà ser modificat. Això implica seguir unes pautes de bones practiques per evitar en el futur qualsevol vulnerabilitat en el codi. També, com s'ha comentar anteriorment per interactuar amb el protocol Ethereum implica un cost per cada transacció.

4. Introducció a l'aplicació a desenvolupar

4.1. Introducció

L'aplicació que es desenvoluparà és una proposta per a que entitats de formació (escoles, universitats, centres de formació, ...) puguin gestionar els certificats dels alumnes de manera segura i sense intermediaris, mostrant una possible alternativa de gestió mitjançant la tecnologia blockchain i el metode *proof-of-concept* els quals podrien validar aquesta idea.

En aquesta aplicació s'evitarà emmagatzemar qualsevol dada privada, i sempre s'utilitzarà claus publiques que estaran associades a cada tipologia d'usuari. Les tipologies establertes en aquesta aplicació són; organitzacions, alumnes i un administrador.

La funcionalitat de les entitats de formació es gestionar els certificats dels alumnes de manera que quedin associats a la seva clau publica. Els estudiants només podran consultar, descarregar i compartir els seus certificats. La tipologia de l'administrador es crea només per validar que les entitats donades d'alta són correctes, és a dir, en un possible funcionament real aquest administrador contactaria amb l'entitat per certificar que realment són ells.

4.2. Tecnologies

4.2.1. Ethereum

Ethereum (https://ethereum.org/) és el protocol blockchain que s'utilitzarà per crear el *back-end* mitjançant un *smart-contract*.

En el *smart-contract* -o contracte intel·ligent- es definiran les estructures de dades on s'emmagatzemaran part de la informació i els mètodes per gestionar-la. També es farà us de la clau publica de cada usuari per ser identificats.

Aquesta tecnologia podem dir que és la principal de l'aplicació ja que guardarà de manera segura i immutable tots els registre per saber qui, quan i quina operació a executat.

4.2.2. IPFS

IPFS (https://ipfs.io/) és un protocol que defineix un sistema de fitxer distribuït, aquest és de codi lliure i ofereix una alta disponibilitat d'accés als fitxers publicats mantenint aquests copiats per diferents nodes. Cada cop que un fitxer es pujat al servei, aquest retorna un *hash* únic que l'identifica.

L'ús que es realitzarà de la tecnologia IPFS és guardar la part del *front-end*, part de la informació de les organitzacions i l'alumne, i els certificats en PDF dels estudiants. Aquesta tecnologia ens oferirà l'accés i l'emmagatzematge dels fitxers i informació sense cap cost.

4.2.3. Reactjs

Reactjs (https://reactjs.org/) és una llibreria Javascript de codi lliure i desenvolupat per Facebook que permet crear interfícies d'usuari on el desenvolupador no ha de pensar en fer la gestió de les dades. Les interfícies poden ser generades des de la part del servidor o bé des del client, en aquest cas es generada des del client.

Amb aquesta llibreria es desenvoluparà la part del *front-end* amb la qual l'usuari interactuarà. També, es combinarà amb la llibreria CSS anomenada *Skeleton (http://getskeleton.com/)* per donar un aspecte més agradable a la interfície.

4.2.4. Web3.js

Web3.js (https://web3js.readthedocs.io/en/1.0/) és una llibreria Javascript que permet interactuar amb els node d'Ethereum.

Aquesta llibreria es situara a la part del *front-end* i permetrà la comunicació amb el contracte intel·ligent d'Ethereum.

4.2.5. Metamask

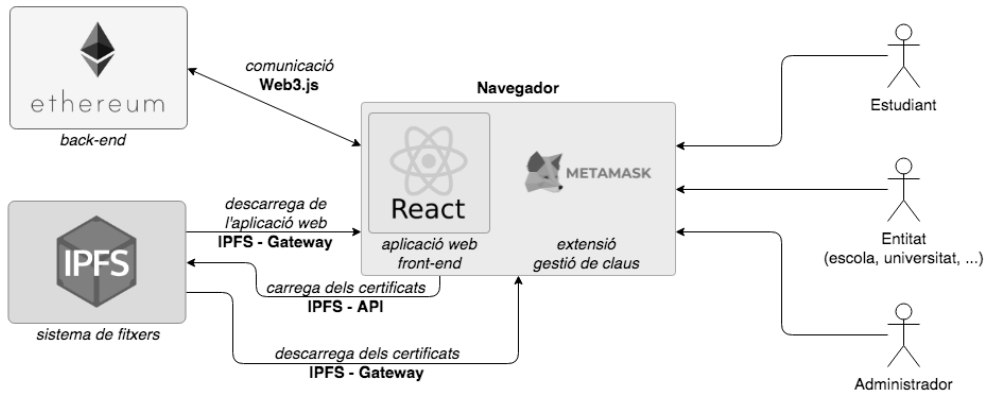
Metamask (https://metamask.io/) és una extensió de codi lliure pels navegadors Chrome i Firefox que permet gestionar les claus publiques d'Ethereum.

La utilització de la extensió en el treball, és disposar de diferents claus i així poder simular i canviar fàcilment entre les tipologies d'usuaris (administrador, estudiants i entitats).

4.3. Estructura de l'aplicació

La Figura 6 mostra de manera gràfica com s'estructurarà l'aplicació.

Figura 6: Estructura de l'aplicació



Font: Elaboració pròpia

5. Desenvolupament

5.1. Anàlisis i disseny

L'aplicació tindrà tres tipologies d'usuaris que disposaran dels següents permisos i funcions:

- **Administrador:** L'usuari que crea i publica el smart-contract a la xarxa d'Ethereum serà considerat com administrador de l'aplicació. Aquest només tindrà una funció disponible la qual li permetrà afegir noves claus publiques amb la tipologia d'organització (escola, universitat, ...)
- **Organitzacions:** Les organitzacions són aquells usuaris afegits per l'administrador i que disposen de les funcions; afegir nous alumnes, afegir nous certificats i actualitzar les seves pròpies dades com ara el nom, telèfon, pàgina web i el correu electrònic. Al afegir un nou estudiant es pot donar el cas que aquest ja existeixi perquè ha sigut donat d'alta per una altra organització, en aquest cas el sistema avisarà mitjançant un error.
- **Alumnes:** Per últim, la tipologia alumnes són aquells afegits per les organitzacions i en aquest cas només podran actualitzar les seves dades (nom i cognom) i visualitzar els seus certificats acadèmics.

Aquestes tipologies seran definides al contracte i relacionades amb cada clau d'usuari registrada, d'aquesta manera en la interficie web s'habilitaran unes funcions o unes altres depenent el tipus d'usuari. La decisió de que els usuaris sempre siguin donats d'alta per part de l'administrador (cas organitzacions) o per part de les organitzacions (cas alumnes) és per verificar l'existència de l'organització / alumne, i per evitar altes aleatòries i sense sentit per part d'usuaris / robots d'Internet.

Les estructures de dades de les tipologies *organitzacions* i *alumnes* són les següents:

- **Organitzacions:**
  - **Nom:** Cadena de text amb el nom de l'organització (ex.: Universitat Oberta de Catalunya)
  - **Telèfon:** Cadena de text amb el telèfon.
  - **Correu Electrònic:** Cadena de text amb el correu electrònic.
  - **Pàgina web:** Cadena de text amb la pàgina web.
  - **Nombre de certificats expedits:** Numero natural amb el nombre total de certificats expedits.
  - **Afegit per:** Clau publica de l'usuari que ha afegit l'organització.
  - **Afegit el dia:** Data en la que s'ha afegit l'organització.
  - **Última actualització:** Data de l'última actualització de les dades; nom, telèfon, correu electrònic i pàgina web.

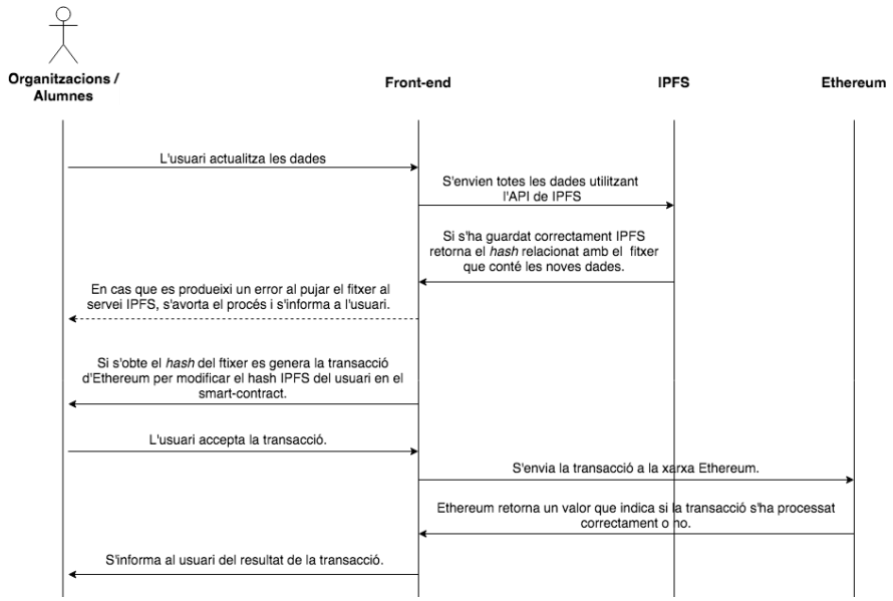
- **Alumnes:**
  - **Nom:** Cadena de text amb el nom.
  - **Cognoms:** Cadena de text amb el/s cognom/s
  - **Nombre de certificats:** Numero natural amb el nombre total de certificats.

- **Certificats:** Matriu amb tots els certificats de l'alumne. Per cada posició es guardarà la estructura de dades:
  - **Organització:**
    - **Nom:** Cadena de text amb el nom de l'organització que ha afegit el certificat.
    - **Clau publica:** Clau publica de l'organització que ha afegit el certificat.
  - **Estudis:** Cadena de text amb el nom del programa realitzat (ex. Grau d'Enginyeria Informàtica).
  - **Nota:** Cadena de text amb la nota final (ex. B, o 6)
  - **Fitxer:** Cadena de text amb el *hash* del certificat en PDF allotjat en el servei IPFS.
  - **Afegit el dia:** Data en la que s'ha afegit el certificat.
- **Afegit per:** Clau publica de l'organització que ha afegit l'alumne.
- **Afegit el dia:** Data en la que s'ha afegit l'alumne.
- **Última actualització:** Data de l'ultima actualització de les dades; nom i cognoms.

Aquestes estructures de dades seran guardades una part en el contracte d'Ethereum, i l'altre en el servei IPFS. He optat en no guardar totes les dades en el contracte ja que en una primera versió de l'aplicació, trobava que al actualitzar les dades de l'organització o de l'alumne el cost de la transacció era força elevat degut a al gran nombre de bytes que s'enviaven.

En el següent gràfic és pot veure un diagrama de com es guardaran les dades de les organitzacions i l'alumne.

Figura 7: Diagrama amb els procés d'actualització de les dades de les organitzacions i alumnes



Font: Elaboració pròpia

La lectura de les dades, es realitzaran de manera inversa de manera que primer es llegeix les dades en el contracte d'Ethereum i posteriorment es consulta el fitxer a IPFS mitjançant el *hash* del fitxer.

En els següents punts es troba totes les funcionalitats i estructures de dades descrites amb més detalls.

5.2. Entorn de treball

L'entorn de treball que s'ha preparat pel desenvolupament ha sigut de dos màquines virtuals amb el sistema operatiu Ubuntu 16.04.3, on una executarà un node d'Ethereum i en l'altra el servei IPFS. En el annex 8.1 és pot consultar la instal·lació del node d'Ethereum i en el 8.2 la del IPFS.

Per la interfície web com es executada a la part del client -es a dir el navegador- no he cregut necessari preparar cap màquina virtual amb un servidor ja que amb el mòdul SimpleHTTPServer que ofereix Python ens servirà per simular el servidor HTTP. Per activar aquest mòdul s'ha d'obrir la consola de comandaments, situar-se en el directori on estigui els fitxers de l'aplicació web i executar el següent comandament: `python -m SimpleHTTPServer 8000`. Ara quan s'accedeixi a l'adreça http://localhost:8000 el mòdul de Python ens retornarà com a resposta els fitxers de l'aplicació pel protocol HTTP.

Pel desenvolupament del *smart-contract* és farà ús de la següent eina web http://remix.ethereum.org que permet crear l'aplicació i simular les crides als mètodes tot des de la mateix eina, després tens la possibilitat d'enviar el contracte a la xarxa d'Ethereum (local o d'Internet) i interactuar amb aquest.

5.3. Back-end

En aquest apartat es comentarà les estructures de dades i els mètodes del contracte intel·ligent creat, que es pot trobar sencer en apartat 8.5 dels annexos.

En la següent taula es poden veure les estructures que guardaran les organitzacions, els alumnes i els certificats. L'estructura de les organitzacions i dels alumnes són força semblants, la diferència és que la de l'alumne necessita també guardar les estructures dels certificats.

```
struct Organization {
  string ipfs_hash;
  uint16 qt_certificates_issued;
  address added_by;
  uint256 added_timestamp;
  uint256 updated_timestamp;
}

struct Student {
  string ipfs_hash;
  mapping (uint8 => Certificate) certificates;
  uint8 qt_certificates;
  address added_by;
  uint256 added_timestamp;
  uint256 updated_timestamp;
}

struct Certificate {
  uint8 id;
  string ipfs_hash;
  address added_by;
  uint256 added_timestamp;
}
```

La majoria del tipus de dades que s'utilitzen són fàcilment identificables excepte el tipus mapping i address. El *mapping*, el que forma internament és una espècie de matriu que en aquest cas el índex és un nombre positiu de 8bits i el valor que trobarem és un objecte de tipus *certificate*. He trobat suficient definir l'índex com a 8 bits ja que un alumne no crec que disposi de més de $2^8 = 256$ certificats.

L'altre tipus *address*, es per guardar claus públiques de manera eficient i evitar utilitzar el tipus *string* que utilitzà més bits de memòria.

En el contracte també es pot veure algunes constants definides, com per exemple les que hi ha a continuació. Aquestes són les que defineixen i s'utilitzen per indicar les tipologies d'usuaris.

```
bytes1 public constant T_ADMIN = 0x01;
bytes1 public constant T_ORGANIZATION = 0x02;
bytes1 public constant T_STUDENT = 0x03;
```

Els següents *mappings* són els que emmagatzemen els usuaris de l'aplicació, aquestes han sigut definides de manera publiques per a que puguin ser consultades sense necessitar desenvolupar cap mètode (ex.: get_user() o get_organization()).

```
mapping (address => bytes1) public _users;
mapping (address => Organization) public _organizations;
mapping (address => Student) public _students;
```

Per últim, els mètodes definits són els següents.

```
constructor() public payable
addUser(address paddress) public payable
addCertificate(address pstudent, string pipfs_hash) public payable
setStudent(string pipfs_hash) public payable
setOrganization(string pipfs_hash) public payable
getCertificate(address pstudent, uint8 pcertificate) public constant returns (uint8, address, string, uint256)
```

Constructor: només s'executa quan el contracte es enviat a la xarxa. Aquest mètode obté l'adreça de qui envia el contracte i l'assigna com administrador.

addUser: Aquest mètode és per afegir nous usuaris. Si es executat per l'administrador el nou usuari serà definit com a organització, en canvi si l'executa una organització l'usuari afegit serà un alumne. En cas, que un alumne executi el mètode la transacció retornarà un error.

addCertificate: Aquest mètode només els usuaris de tipus organització el poden executar. La seva funció és assignar els certificats als alumnes.

setStudent: Mètode per actualitzar les dades del alumne, la seva execució només esta permesa per usuaris de tipus alumne.

setOrganization: Mètode per actualitzar les dades de l'organització, en aquest cas l'execució només esta permesa per usuaris tipus organització.

getCertificate: L'últim mètode es per consultar els certificats d'un alumne.

Per acabar, tots els mètodes estan definits com a públics i la majoria porten el terme *payable* que vol dir que per ser executat l'usuari haurà de pagar un petit import ja que modifiquen l'estat del contracte.

5.4. Front-end

La part del *front-end* es desenvolupada amb la llibreria Reactjs i el que s'ha creat són diferents components per fer la interfície de manera modular. Cada component conté el codi HTML i la lògica en Javascript que s'encarrega de consultar, modificar i mostrar les dades de cada usuari.

En la següent figura és pot veure com s'estructurarà.

Figura 8: Esquema gràfic de la interficie

Font: Elaboració pròpia

En la part superior de la pàgina -text dinàmic- es mostrarà la clau publica del usuari que esta accedint a la pàgina juntament amb la quantitat d'Ethereum que té disponible.

A continuació, si l'usuari està registrat a l'aplicació es mostrarà el component *userData* i un dels components *adminDashboard, organizationDashboard o studentDashboard* depenent de la tipologia del usuari. En cas que no estigui donat d'alta, es mostrarà un error.

El component *userData*, mostra les dades que hi ha en el contracte del usuari identificat. Aquestes dades poden variar depenent de si l'usuari és administrador, organització o alumne. A continuació és pot veure una captura de cada cas.

Figura 9: Captures del component *userData*. (1a Administrador, 2a Organització i 3a Alumne)

Font: Elaboració pròpia

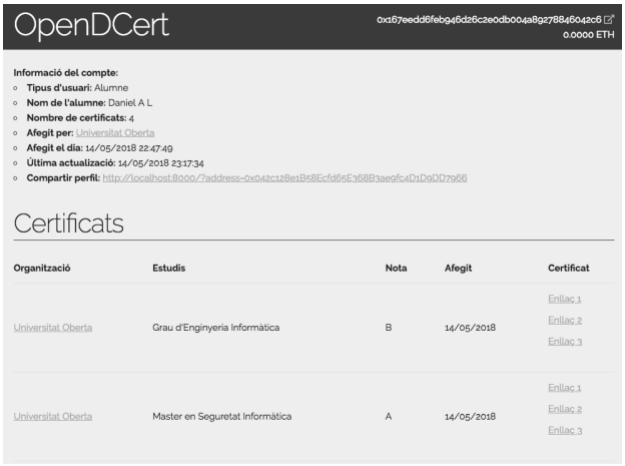
Els altres components *adminDashboard, organizationDashboard o studentDashboard* són els encarregats de mostrar els formularis per cada tipologia. En el *adminDashboard* mostra el formulari per afegir noves organitzacions. En el *organizationDashboard* permet actualitzar les dades de l'organització, afegir nous alumnes i certificats. I per últim, en el *studentDashboard* permet actualitzar les dades del alumne i visualitzar els seus certificats.

Figura 10: Captura parcial de la interfície d'un usuari tipus organització

Font: Elaboració pròpia

Tant les organitzacions com els alumnes poden compartir el seu perfil amb altres persones, en aquest cas no es mostra cap formulari i només es permet la visualització de les dades.

Figura 11: Captura parcial del perfil d'un alumne on es mostra la seva informació i els seus certificats



Font: Elaboració pròpia

El codi de la part del *front-end* es pot consultar en el fitxer comprimit adjunt, dins del directori ***dApp***.

5.5. Desplegament

5.5.1. Local

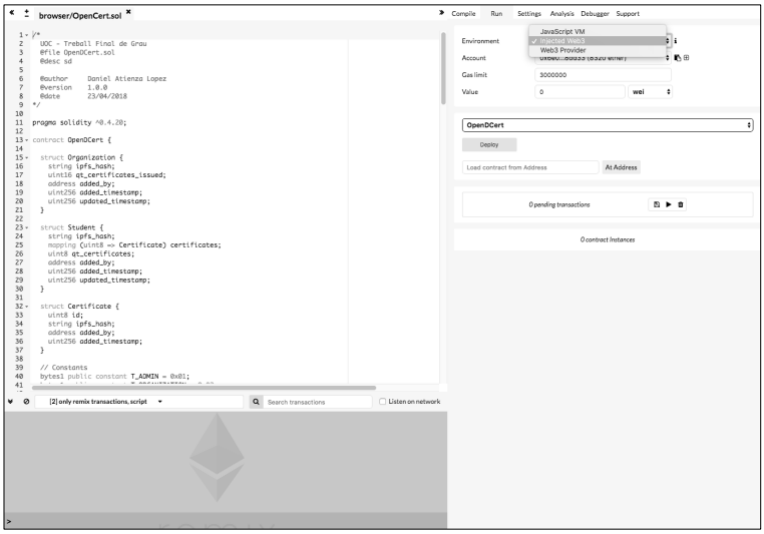
Pel desplegament en l'entorn local primer s'ha de comprovar que les dos màquines virtual (Ethereum i IPFS) estiguin amb el servei iniciat.

El següent pas es connectar l'extensió Metamask al nostre node i importar les claus que s'han generat prèviament, tots dos passos estan descrits en els annexos en els apartats 0 i 8.6 respectivament.

A continuació ja es pot crear el contracte intel·ligent en el nostre node, per fer aquest pas s'utilitzarà l'IDE https://remix.ethereum.org/ que permet publicar el contracte al node / xarxa que estigui connectada l'extensió Metamask. En el següent apartat l'utilitzarem per fer la creació a la xarxa de proves *Rinkeby*.

A la part esquera del IDE Remix carregarem el codi del contracte, i a la part dreta es molt important que en el apartat *run* es marqui en *Environment* l'opció *Injected Web* tal i com mostra la Figura 12. Aquesta opció seleccionada, vol dir que serà l'extensió Metamask el que generarà la transacció i la enviarà al node connectat.

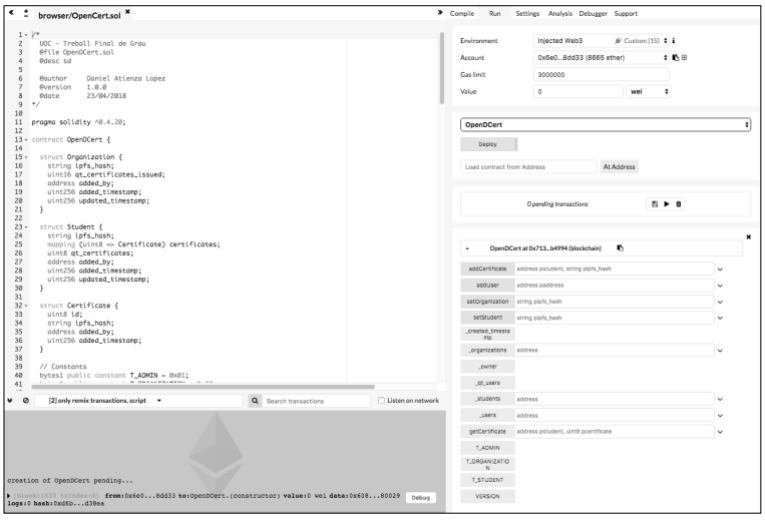
Figura 12: IDE Remix (https://remix.ethereum.org/) amb el codi del contracte



Font: Elaboració pròpia

Ara ja es pot fer clic en el boto que diu *Deploy* i acceptar la transacció que mostrarà l'extensió Metamask. Si tot va bé, a la part inferior dreta es mostrarà l'adreça on s'ha creat el contracte i també totes les variables i mètodes públics on des del mateix lloc s'hi podrà interactuar.

Figura 13: IDE Remix amb el contracte ja creat



Font: Elaboració pròpia

Per a que el *front-end* interactuï amb el contracte, s'ha d'indicar en el fitxer de configuració (8.8 Fitxer de configuració del front-end) l'adreça on es troba i tots els mètodes i variables que disposa amb l'estàndard ABI (Interficie Binaria d'Aplicacions). Aquesta última informació es pot trobar en el IDE Remix en la secció *Compile* i fent clic en el boto *Details*.

Per últim, en el fitxer de configuració del *front-end* només en faltarà introduir les dades del node local del IPFS (protocol, IP i ports) i indicar en el paràmetre *mode* que volem utilitzar l'entorn *local*. Així doncs, ara ja podem iniciar el mòdul python per a que creï el servidor web.

```
# Previament ens hem de situar en el directori on estan els fitxer del front-end
python -m SimpleHTTPServer 8000
```

Per comprovar que tot funciona correctament podem accedir a la direcció http://localhost:8000 seleccionant prèviament en l'extensió Metamask el compte que ha creat el contracte intel·ligent.

5.5.2. Internet

Alhora de fer el desplegament de manera global a Internet s'ha de crear el contracte a la xarxa de proves Rinkeby utilitzant l'IDE Remix, en aquest cas amb el Metamask connectat a la xarxa comentada. Abans de fer la creació del contracte s'ha d'importar unes noves claus que ja disposen d'Ethereums per aquesta xarxa, aquestes claus es poden trobar en un fitxer comprimit adjuntes a l'entrega de la memòria.

Un cop importades les claus ja es podrà crear el contracte seleccionant la clau de l'administrador. Aquest procés ja ha sigut realitzat i es pot trobar el contracte en la següent adreça: https://rinkeby.etherscan.io/address/0x83fd4676d4505ce912b3fbd1a52c2b38a27398ce .

Ara, en el fitxer de configuració del *front-end* afegirem les dades del contracte i canviarem el parametre *mode* a *production* per a que l'aplicació seleccioni aquestes dades. En aquest cas, també s'utilitzarà un servei d'Internet gratuït per accedir al IPFS. Per defecte tots els nodes públics d'IPFS no permeten guardar fitxers, excepte aquest servei gratuït (https://infura.io/) que permet pujar nous fitxers mitjançant l'API.

L'últim pas per deixar l'aplicació totalment descentralitzada i així que funcioni sense cap intermediari només ens queda pujar l'aplicació web al node IPFS que hem creat. Com que aquest node està connectat a la xarxa principal, un cop intentem accedir a la web o qualsevol fitxer a través d'un node públic aquest se'l descarregarà del node creat i guardarà una copia de les dades.

En aquest punt, ja tenim el contracte intel·ligent *(back-end)* funcionant a la xarxa d'Ethereum Rinkeby i tota l'aplicació web *(front-end)* a la xarxa d'IPFS. Això vol dir que per a que deixi de funcionar tant el *back-end* com el *front-end* haurien de desaparèixer tots els nodes de tots dos protocols, cosa poc probable.

Per pujar l'aplicació web al node IPFS que hem creat s'ha d'executar el següent comandament.

```
ipfs add -r <directori_de_l_aplicacio>/

# Resultat de l'execució
added QmS8E2wHVFvmjzBa91K2PW29zxnKnSMvUZb52TgDseES1L Web/config.js
added QmbTtwbDAjPqXr6xVRAJJZyqjxXWGCxi9XJkUzkkwRdXqu Web/config.sample.js
added QmZTU7gGa8n6GyCzGkpFwCgpkWFVLzTyrQt2RZS6pUyGZt Web/css/fontawesome-all.min.css
added QmQAPL7vsWyQSMcMzKBzALWXzFDQSRuwcRpdTXs6oi2yyZ Web/css/normalize.css
added QmRyrbwkktbe4SM4Q6ZYV1TENEwMKVfHBShnVbqyyKmovh Web/css/skeleton.css
added QmQShUMNp5fRavE1xEvzn7N9BchpzVCsUqvRL457i5zXKA Web/css/spinner.css
added QmX17vp61BeZxG578MLMPPF6GWUguAg9UmxnSjyezcTnDL Web/css/style.css
added Qmepw8NRsxbMZgreiK1pTFRa7ts3zdu5HWC25DNqCfj7qx Web/favicon.ico
added QmSLkxth92eT6tQb4nUAYLwJ6yZx9g5fdGkuBQPdAtnhkm Web/img/loader.gif
added QmbKHaMb8yN6Nd1tQXUMUQHWufTULU1xh7sAUPx37zNf6z Web/index.html
added QmWn12gJz646RUmdFn2FTAmzfDdMgKNQ53VuZrHGreEEzH Web/js/app.js
added Qmcd8LkrLRTn2nr3QP5oLnawnZRWjHaWouNVC3grubYKsv Web/js/babel.min.js
added QmPb2bYvLzhqJojb5uXvjoYFfHEtfUyk7HyFYeKBh6NrAw Web/js/components/adminDashboard.jsx
added QmS6dPGo98BknfBLiQTxCr6xCNuUQct8p7sbw9boCHwtC1 Web/js/components/organizationDashboard.jsx
```

```
added QmfSVLnh9FhMy1eBHL6bePuPXYphYK5bN96QrzACwWQxVQ Web/js/components/studentDashboard.jsx
added QmcDZtVvBc38EbqkcSRaoK9uLg3BZxGWAvHow3ryzz9p41 Web/js/components/userData.jsx
added Qmd4jTLELdCvJEjKt1stCML4pcRkzTmyu3AUJkHVgFMeiA Web/js/ipfs-api.min.js
added QmYqwsWC6D9BoNvWEndgJkXdPAFPtLLsYz3ynTwBgSH2ya Web/js/moment.min.js
added QmZMH7MgzgiYkN1KjY2FvhevERRNtTTq8Tw4fUAp9Agt2H Web/js/react-dom.production.min.js
added QmXnC9mp27N3fMwTen49HJaEoM7U3gxVdTjQ6h6nrCKipK Web/js/react.production.min.js
added QmZ9H6i5WNcNH5tM3q6NX3UzsQ1KKpjZ1e5cEXNEzZ2pEf Web/js/tools.js
added QmWVXNjSgHcSiczBaYa9xTwgyCmgMXvFsnheDfE1carF5G Web/webfonts/fa-brands-400.eot
added QmTyDQPGtGaaiJr1fnCKockSfiydm43SACnGGu6xAKLHi2 Web/webfonts/fa-brands-400.svg
added QmPrXHHDycjoJiJVA4GVXtGjv12Q8sFAYqiMaQRepb9CHe Web/webfonts/fa-brands-400.ttf
added QmdWtRgbsaNBFdRpe7uT2aUajy7KE6uLCSVPCKHzaQhrr9 Web/webfonts/fa-brands-400.woff
added QmfZxbx7dfvkHiBqBHNxWffqMHVGW67uoDKrwBgkyuQyQ6 Web/webfonts/fa-brands-400.woff2
added Qmdy7E94GsJRgF8roLBkQJcLrxAFZYH2NK6UHPHYEW97e6 Web/webfonts/fa-regular-400.eot
added QmUvetbezTq4PHWFDPGzV9LtQPd1DiBPkC9WMWQMSn71Hs Web/webfonts/fa-regular-400.svg
added QmVTDtUufHoy5PcyCJGmDKsW8tEsF4Dr4LLNaWuWeQsief Web/webfonts/fa-regular-400.ttf
added QmS9XLWFWYmQ17yY4gVyG3a9C8JKbD4kw6Va1kDvfVBX4R Web/webfonts/fa-regular-400.woff
added QmVHw6eBRYjH6RtsdiYWRn3e9isrDRpHgjWsPzxWixtK3w Web/webfonts/fa-regular-400.woff2
added QmWejgHgLEYBLiWTFV2nkNDPz97X1xo5GJFaAiaKt4MR5N Web/webfonts/fa-solid-900.eot
added QmYVThfqrKQ8bMgdSLK26GDMUHN7rX6zCp3LSWTy7Di3bA Web/webfonts/fa-solid-900.svg
added QmbErpcAdciRcZRbxiy6tYpN25VoC9yv9dVFsm2C9qtgWd Web/webfonts/fa-solid-900.ttf
added QmRzDNjQtCJMoNsC392itAgoBboHaZdDJ6qyjF6W7zfNYX Web/webfonts/fa-solid-900.woff
added Qmc7goJSpRWYbu1Sithn9WDxrKLGnn5mdFoXxHvK1nPcvw Web/webfonts/fa-solid-900.woff2
added QmaAgYVLoMcpa2jHDUidzuHgcWigyTcT685YYMcbm1uEhu Web/css
added QmR7kHXdgaLPKEcdEhYFkH1YYPawNpsp2hWrXpuTStzGMo Web/img
added QmX5uyARQVsLzt95HaMEAWzPaLSQiKpj52jmSjTT7XoGJ9 Web/js/components
added QmfKcFTCXQN2a1DFy5JopBXx2qs5KXknhxwDBHVwQEF9rV Web/js
added QmcEML9jp41KrueCK6bMHiQ9gj2EY5KRcZi7AKwhQZw7oF Web/webfonts
added Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa Web
```

En el resultat es pot veure tots els fitxer carregats amb el seu *hash*. Si agafem l'últim *Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa* i el cridem pel protocol HTTP veurem que es carrega l'aplicació.

Les següents adreces són nodes IPFS públics, on es pot accedir a l'aplicació web.

```
http://104.131.131.82:8080/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
http://104.236.179.241/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
http://128.199.219.111/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
http://104.236.76.40/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
http://178.62.158.247/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://siderus.io/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://ipfs.io/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://gateway.ipfs.io/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://ipfs.infura.io/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://www.eternum.io/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://hardbin.com/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
https://ipfs.renehsz.com/ipfs/Qmcb8842XtF4DSo1aDsVACLXfVKFT2H3E56dBstn8ZZNEa/
```

5.6. Avaluació del desplegament

El desplegament de l'aplicació en els dos entorns (local i Internet) s'han realitzat de manera ràpida i només m'he trobat amb un problema alhora d'utilitzar el protocol IPFS a Internet. Pel que fa l'ús de la xarxa d'Ethereum (local i Rinkeby) no hi hagut cap problema al publicar el contracte i interactuar amb aquest.

El problema obtingut amb el protocol IPFS és que els nodes públics d'Internet no permeten la publicació de cap fitxer i només s'hi pot accedir en mode lectura cosa que no hi comptava. Com ha solució, s'ha trobat el servei d'Infura (https://infura.io/) on permet de manera gratuïta fer us d'un node IPFS que si que permet la pujada de fitxers.

5.7. Valoració del resultat obtingut

El resultat obtingut és positiu ja que en aquest últims capítols amb el desenvolupament de l'aplicació, s'ha pogut veure de manera pràctica tota la part teòrica descrita a l'inici del treball.

Desenvolupant el contracte intel·ligent sobre Ethereum s'ha pogut comprovar la seguretat i fiabilitat que pot aportar la tecnologia *blockchain* a l'usuari final, ja que qualsevol execució dels mètodes que modifiquen l'estat del contracte quedarà registrat en la cadena de blocs.

Pel que fa el protocol IPFS es pot comprovar que sempre podrem trobar els certificats i l'aplicació en línia, i només podríem tenir problemes d'accés en el cas que tots els nodes que disposen dels nostres fitxers estiguin fora de connexió. Actualtment, els fitxers de l'aplicació web i els certificats estan replicats a un mínim de dotze nodes (es pot consultar la llista en l'apartat 5.5.2).

Un altre punt a comentar és que l'aplicació utilitza nodes públics per així reutilitzar les xarxes (Ethereum i IPFS) ja existent. D'aquesta manera el treball l'enfocat en descriure la tecnologia *blockchain* i les aplicacions descentralitzades. Si mai s'hagués de realitzar de manera real, crec que el més òptim i a més si volem introduir dades privades, és crear una xarxa privada dels protocols juntament amb altres organitzacions.

Per exemple, a nivell nacional s'ha creat un consorci anomenat Alastria (https://alastria.io/) on es vol crear un ecosistema de *blockchain*. En aquest projecte està participant empreses com ara Repsol, Everis, Banco Santander, entre altres... amb l'interès d'implementar la tecnologia en les seves empreses. Aquest consorci ofereix una xarxa i serveis de *blockchain* on també hi ha la possibilitat de que les empreses participants puguin incorporar nous nodes per fer la xarxa més segura.

6. Conclusions

Aquest treball m'ha permès obtenir uns coneixements actuals del que es la tecnologia *blockchain* i com s'està enfocant cara els pròxims anys, i també gràcies a la part pràctica he pogut adquirir nous coneixements de tecnologies que fins ara mai no havia treballat.

Al finalitzar el treball, es pot comprovar que s'han assolit tots els objectius proposats en el pla de treball i la planificació de les tasques ha sigut l'adient, excepte al inici del semestre que vaig haver d'ajustar una setmana per estar inactiu uns dies per malaltia.

El desenvolupament de l'aplicació descentralitzada m'ha aportat altres idees on la implementació de la tecnologia *blockchain* seria clau. Un exemple és, una plataforma per realitzar les entregues de les PACs, PRAs o exàmens virtuals de la UOC oferint als professors i alumnes un historial -impossible de modificar- on es podria fer el seguiment de les entregues o revisions realitzades.

També l'inconvenient que s'ha vist al fer ús del protocol d'Ethereum en la xarxa Rinkeby es el preu aleatori al interactuar amb el contracte que depèn de les transaccions pendents a processar. Aquest punt pot ser un problema cara al futur ja que si la xarxa està molt saturada el preu de les transaccions pot ser força alt i també el temps de confirmació.

Aquests problema es podria solucionar migrant la part del *back-end* al protocol d'EOS ja que en aquest cas les transaccions són gratuïtes pels usuaris i també ofereixen un gran volum de procés de transaccions. El protocol es preveu que estigui disponible a partir de juny al alliberar-se la primera versió.

A part de la migració a EOS, també es podria plantejar la creació d'una moneda criptogràfica per a que els alumnes puguin pagar la taxa dels certificats o bé que els centres acceptessin alguna de les monedes que hi ha actualment i així tenir també un seguiment dels pagaments realitzats.

7. Bibliografia

[1] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System,» [En línia]. Available: https://bitcoin.org/bitcoin.pdf. [Últim accés: 21 03 2018].

[2] S. Haber i W. S. Stornetta, «How to Time-Stamp a Digital Document,» [En línia]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.8740&rep=rep1&type=pdf. [Últim accés: 21 03 2018].

[3] P. Todd, «Bitcoin Improvement Proposal (BIP) 65,» 01 10 2014. [En línia]. Available: https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki. [Últim accés: 17 04 2018].

[4] E. «EOS.IO Technical White Paper,» [En línia]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. [Últim accés: 21 03 2018].

[5] J. Benet, «IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3),» [En línia]. Available: https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf. [Últim accés: 21 03 2018].

[6] D. Bayer, S. Haber i W. S. Stornetta, «Improving the Efficiency and Reliability of Digital Time-Stamping,» [En línia]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.4891&rep=rep1&type=pdf. [Últim accés: 21 03 2018].

[7] «Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,» [En línia]. Available: https://github.com/ethereum/wiki/wiki/White-Paper. [Últim accés: 21 03 2018].

[8] «Wikipedia - Blockchain,» [En línia]. Available: https://en.wikipedia.org/wiki/Blockchain. [Últim accés: 21 03 2018].

[9] «Wikipedia - Merkle tree,» [En línia]. Available: https://en.wikipedia.org/wiki/Merkle_tree. [Últim accés: 21 03 2018].

[10] «Wikipedia - Satoshi Nakamoto,» [En línia]. Available: https://en.wikipedia.org/wiki/Satoshi_Nakamoto. [Últim accés: 21 03 2018].

[11] «Wikipedia - Bitcoin,» [En línia]. Available: https://en.wikipedia.org/wiki/Bitcoin. [Últim accés: 21 03 2018].

[12] «Wikipedia - InterPlanetary File System,» [En línia]. Available: https://en.wikipedia.org/wiki/InterPlanetary_File_System. [Últim accés: 21 03 2018].

[13] «Wikipedia - Ethereum,» [En línia]. Available: https://en.wikipedia.org/wiki/Ethereum. [Últim accés: 21 03 2018].

[14] «Wikipedia - EOS.IO,» [En línia]. Available: https://en.wikipedia.org/wiki/EOS.IO. [Últim accés: 21 03 2018].

[15] «Wikipedia - ReactJS,» [En línia]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Últim accés: 21 03 2018].

[16] «Wikipedia - Hash Function,» [En línia]. Available: https://en.wikipedia.org/wiki/Hash_function. [Últim accés: 21 03 2018].

[17] «Wikipedia - Collision Resistance,» [En línia]. Available: https://en.wikipedia.org/wiki/Collision_resistance. [Últim accés: 21 03 2018].

[18] «Wikipedia - Cryptographic Hash Function,» [En línia]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function. [Últim accés: 21 03 2018].

[19] «Wikipedia - One-way function,» [En línia]. Available: https://en.wikipedia.org/wiki/One-way_function. [Últim accés: 21 03 2018].

[20] «ReactJS - Documentació,» [En línia]. Available: https://reactjs.org/docs/hello-world.html. [Últim accés: 21 03 2018].

[21] «Ethereum - Documentació,» [En línia]. Available: http://www.ethdocs.org/en/latest/. [Últim accés: 21 03 2018].

[22] «Solidity - Documentació,» [En línia]. Available: https://solidity.readthedocs.io. [Últim accés: 21 03 2018].

[23] «IPFS - Documentació,» [En línia]. Available: https://ipfs.io/docs/install/. [Últim accés: 21 03 2018].

[24] «Wikipedia - Timestamp,» [En línia]. Available: https://en.wikipedia.org/wiki/Timestamp. [Últim accés: 21 03 2018].

[25] W. Dai, «b-money,» 1998. [En línia]. Available: http://www.weidai.com/bmoney.txt. [Últim accés: 15 04 2018].

[26] EOS.IO, «Introduction to Blockchain: Daniel Larimer at Virginia Tech,» 15 04 2018. [En línia]. Available: https://www.youtube.com/watch?v=sYAktmG1NuA. [Últim accés: 16 04 2018].

8. Annexos

8.1. Instal·lació d'un node d'Ethereum en local

El node d'Ethereum que s'instal·larà en local no estarà connectat a la xarxa principal per evitar descarregar totes les dades de la cadena de blocs que hi ha fins ara, ~65Gb. Aquest node funcionarà de manera independent en una xarxa pròpia d'un node, i en la seva configuració inicial -bloc genesis- s'assignaran una sèrie de *tokens* a sis comptes que simularan; l'administrador, dos universitats i tres alumnes.

Els *tokens* assignats només són útils en la xarxa creada per fer us de l'aplicació, fora de la xarxa són inexistents ja que la cadena de blocs serà diferent.

En els següents comandaments instal·la el programari d'Ethereum fent us del repositori oficial.

```
sudo apt-get install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo add-apt-repository -y ppa:ethereum/ethereum-dev
sudo apt-get update
sudo apt-get install ethereum
```

Un cop instal·lat podem fer la creació dels comptes mitjançant el següent comandament, al ser executat ens demanarà una contrasenya que estarà associada amb el compte creat.

```
geth account new
```

Al crear un nou compte, per defecte es crearà un fitxer json en directori `~/.ethereum/keystore`. En aquest fitxer es pot trobar la clau publica, l'adreça del compte que són els últims 20 bytes de la clau publica, la clau privada encriptada amb la contrasenya introduïda, i altres detalls. En l'annex 8.3 és pot veure el fitxer d'un dels comptes creats, i per verificar el directori on estan situats els fitxers es pot executar la següent instrucció.

```
geth account list
```

El següent pas és crear el fitxer *genesis* amb la configuració per a que al executar el servei en les adreces generades disposin de *tokens*, també indicarem al paràmetre *chainID* un nombre que no coincideixi amb la de cap xarxa existent per així crear una pròpia. En l'annex 8.4 es mostra el fitxer que s'ha utilitzat, aquest es pot crear dins del directori creat pel servei d'Ethereum, `~/.ethereum`.

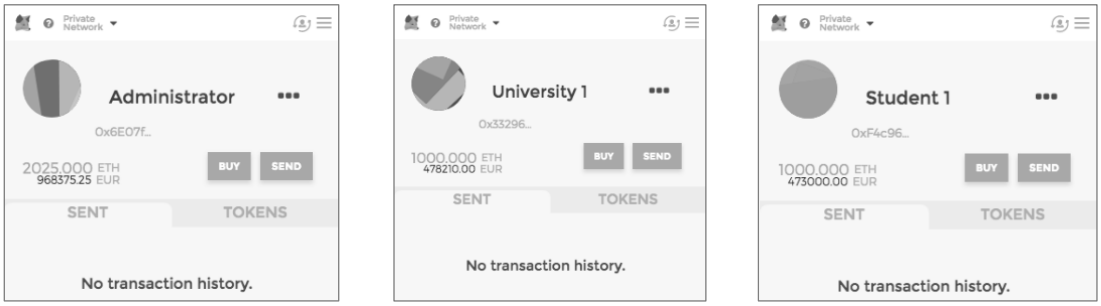
L'últim pas és executar les següents instruccions que el fan és, carregar el fitxer *genesis* per deixar el node preparat en el estat indicat i la següent fica en funcionament el node indicant paràmetres com per exemple el port per on realitzar les consultes al node.

```
geth init ~/.ethereum/genesis.json
geth --identity "UOC-TFG" --networkid 15 --nodiscover --rpc --rpcport "8545" --rpccorsdomain "*" --rpcaddr
"192.168.1.40" --mine --minerthreads=4
```

Si es vol verificar el funcionament del node i que l'assignació de *tokens* s'ha fet correctament, es pot importar a l'extensió Metamask els comptes creats. En la Figura 14 es pot veure els comptes de l'administrador, d'una de les universitats i d'un alumne. En el compte de l'administrador, el nombre de

tokens incrementa automàticament perquè és l'adreça que rep les recompenses quan un nou bloc es creat i afegit a la cadena de blocs.

Figura 14: Mostra d'alguns dels comptes importats a l'extensió Metamask



Font: Elaboració pròpia

8.2. Instal·lació d'un node IPFS

IPFS és un protocol que utilitza diferents tecnologies per mantenir fitxers distribuïts per la xarxa, algunes d'aquestes tecnologies són la de *bitTorrent* per compartir els fitxers, *git* per tenir un control de versions, i d'altres. El funcionament principal es que a partir de les peticions que rep el node, primer comprova si disposa del fitxer per retornar-lo al usuari i en cas contrari el node farà la petició a la xarxa i si algun altre el té li proporcionarà.

El node que s'instal·larà a continuació quedarà dins de la xarxa existent, d'aquesta manera si afegim un fitxer aquest el podrem sol·licitar des de qualsevol node de la xarxa. Per fer la instal·lació primer s'ha de descarregar el programari de la pàgina web oficial (https://dist.ipfs.io/#go-ipfs), recordar es fa sobre Ubuntu 16.04.3.

Un cop descarregat executem les següents instruccions.

```
tar xvfz go-ipfs_v0.4.14_darwin-amd64.tar.gz
cd go-ipfs/
sudo ./install.sh
```

Inicialitzem el node i modifiquem la configuració dels paràmetres `Addresses.API` i `Addresses.Gateway` per a que s'hi pugui accedir mitjançant el protocol TCP des de qualsevol IP de la xarxa local.

```
ipfs init
ipfs config Addresses.API "/ip4/0.0.0.0/tcp/5001"
ipfs config Addresses.Gateway "/ip4/0.0.0.0/tcp/8080"
ipfs config --json API.HTTPHeaders.Access-Control-Allow-Origin '["*"]'
ipfs config --json API.HTTPHeaders.Access-Control-Allow-Methods '["PUT", "GET", "POST", "OPTIONS"]'
```

Ara, amb la següent instrucció ja podem iniciar el servei IPFS.

```
ipfs daemon --writable
```

Per defecte, al inicialitzar el primer cop et recomana consultar un fitxer per comprovar que el servei s'ha instal·lat correctament. Amb les següents instrucció es pot verificar mitjançant la terminal o bé pel protocol HTTP, en tots dos casos hauríem de poder visualitzar el mateix fitxer. L'última adreça HTTP és per verificar que l'API funciona correctament.

```
# Comprovació des del terminal
ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
```

```
# Comprovació per HTTP
http://192.168.1.41:8080/ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
http://192.168.1.41:5001/api/v0/version
```

Les següents instruccions és per afegir un nou fitxer des de la consola i fer la comprovació d'aquest mitjançant el terminal, el servei web en local i d'un de públic d'Internet.

```
echo "Fitxer afegit el dia" `date -u` "pel treball final de grau (TFG)." > fitxer-TFG.txt
ipfs add fitxer-TFG.txt

# Verificació des del terminal
ipfs cat QmNPHAsNgXGCHUmLGPhhMh7GAV9RSb7H28sW1qJSBtjwg4

# Verificació per HTTP (node local)
http://192.168.1.41:8080/ipfs/QmNPHAsNgXGCHUmLGPhhMh7GAV9RSb7H28sW1qJSBtjwg4

# Verificació per HTTP (node públic d'Internet)
http://104.236.179.241/ipfs/QmNPHAsNgXGCHUmLGPhhMh7GAV9RSb7H28sW1qJSBtjwg4
```

En el següent cas el fitxer s'afegeix utilitzant l'API de IPFS.

```
echo "Fitxer de prova del treball final de grau (TFG) afegit el dia" `date -u` "mitjançant l'API de IPFS." >
fitxer-TFG-API.txt
curl -F file=@./fitxer-TFG-API.txt "http://192.168.1.41:5001/api/v0/add"

# Verificació des del terminal
ipfs cat QmVXHNxoE5nq4FRqA4tVYzgZnmA6Fa6JeHsj1aJU2MkiX5

# Verificació per HTTP (node local)
http://192.168.1.41:8080/ipfs/QmVXHNxoE5nq4FRqA4tVYzgZnmA6Fa6JeHsj1aJU2MkiX5

# Verificació per HTTP (node públic d'Internet)
http://128.199.219.111/ipfs/QmVXHNxoE5nq4FRqA4tVYzgZnmA6Fa6JeHsj1aJU2MkiX5
```

8.3. Fitxer d'un dels comptes creats al node d'Ethereum

```
{
   "address":"f4c962b2e406caa33be58562fdb98dde48170ef7",
   "crypto":{
      "cipher":"aes-128-ctr",
      "ciphertext":"2344a4aa318db45037b49595a2226f4c461ffd9105857b2336088b4d493a4647",
      "cipherparams":{
         "iv":"59c04f5660b76c52e31aaa32b020ed22"
      },
      "kdf":"scrypt",
      "kdfparams":{
         "dklen":32,
         "n":262144,
         "p":1,
         "r":8,
         "salt":"9c6aa650d81c7f784b68808466738c26b8fcf3416021b63b8ecfe828d3dbdec0"
      },
      "mac":"7576fb0de496d428906124637420a63e0be0274c0d6d60ae54f7401081fdb651"
   },
   "id":"f40d9cb1-81cb-4cb1-891e-321418a330a2",
   "version":3
}
```

8.4. Fitxer genesis utilitzat pel node d'Ethereum

```
{
    "config": {
        "chainId": 15,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0,
        "byzantiumBlock": 0
    },
    "difficulty": "0x400",
    "gasLimit": "0x2100000",
    "alloc": {
        "a68ab04da203589a1ea6dd2041591ea2970ec481": {"balance": "100000000000000000000"},
        "4998ec59f43448412a6eafed0bf6a69a81203c4c": {"balance": "100000000000000000000"},
        "49fb86b91b4534a65f535e1249738161da82be99": {"balance": "100000000000000000000"},
        "ec10639df3f7b1ec6ee46abd5ce4791bd4512f2f": {"balance": "100000000000000000000"},
        "87ee68af33d74e6130d96d90132fc5e30621d274": {"balance": "100000000000000000000"},
        "2c6229fa8e5f46f51043971fca0c08da58377718": {"balance": "100000000000000000000"},
        "738ba0dc29b8d6e473dc981d9e03dda05dd8831b": {"balance": "100000000000000000000"}
    }
}
```

8.5. Codi del smart-contract (back-end)

```
/*
  UOC - Treball Final de Grau
  @file OpenDCert.sol
  @desc Smart-contract per l'aplicació descentralitzada OpenDCert.

  @author     Daniel Atienza Lopez
  @version    1.0.0
  @date       23/04/2018
*/

pragma solidity ^0.4.20;

contract OpenDCert {

  struct Organization {
    string ipfs_hash;
    uint16 qt_certificates_issued;
    address added_by;
    uint256 added_timestamp;
    uint256 updated_timestamp;
  }

  struct Student {
    string ipfs_hash;
    mapping (uint8 => Certificate) certificates;
    uint8 qt_certificates;
    address added_by;
    uint256 added_timestamp;
    uint256 updated_timestamp;
  }

  struct Certificate {
    uint8 id;
    string ipfs_hash;
    address added_by;
```

```
    uint256 added_timestamp;
  }

  // Constants
  bytes1 public constant T_ADMIN = 0x01;
  bytes1 public constant T_ORGANIZATION = 0x02;
  bytes1 public constant T_STUDENT = 0x03;
  bytes10 public constant VERSION = "1.0.0";

  address public _owner;
  uint256 public _created_timestamp;
  uint256 public _qt_users = 0;

  // Users mapped to the user type 1 - Admin, 2 - Organizations (Schools, Universities, ...), 3 - Student
  mapping (address => bytes1) public _users;
  mapping (address => Organization) public _organizations;
  mapping (address => Student) public _students;

  /*
    constructor
  */
  constructor() public payable
  {
    _owner = msg.sender;
    _created_timestamp = block.timestamp;

    _users[msg.sender] = T_ADMIN;
    _qt_users++;
  }

  /*
    addUser:
    Method to insert a new organization or a new student.
    The administrator only can add organizations, and the organizations only
    can add students.

    @param address  paddress  Clau publica del usuari
  */
  function addUser(address paddress) public payable
  {
    require((_users[msg.sender] == T_ADMIN || _users[msg.sender] == T_ORGANIZATION) && _users[paddress] ==
0x00);

    bytes1 user_type;

    if (_users[msg.sender] == T_ADMIN) {

      user_type = T_ORGANIZATION;

      Organization memory objOrganization;

      objOrganization.ipfs_hash = "";
      objOrganization.qt_certificates_issued = 0;
      objOrganization.added_by = msg.sender;
      objOrganization.added_timestamp = block.timestamp;

      _organizations[paddress] = objOrganization;
    }
    else if (_users[msg.sender] == T_ORGANIZATION) {

      user_type = T_STUDENT;
```

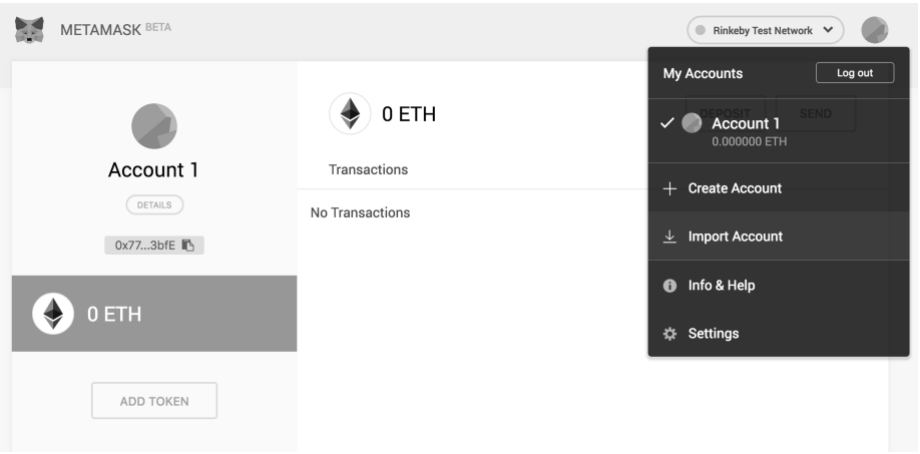
```
      Student memory objStudent;

      objStudent.ipfs_hash = "";
      objStudent.qt_certificates = 0;
      objStudent.added_by = msg.sender;
      objStudent.added_timestamp = block.timestamp;

      _students[paddress] = objStudent;
    }

    _users[paddress] = user_type;
    _qt_users += 1;
  }

  /*
    addCertificate:
    Method to insert a new certificate of a student.
    Only organizations can add certificates.

    @param address  pstudent    Clau publica de l'alumne
    @param string   pipfs_hash  Hash del fitxer en el sistema IPFS
  */
  function addCertificate(address pstudent, string pipfs_hash) public payable
  {
    require(_users[msg.sender] == T_ORGANIZATION && _users[pstudent] == T_STUDENT);

    Student storage objStudent = _students[pstudent];
    Certificate memory objCertificate;

    objCertificate.id = objStudent.qt_certificates + 1;
    objCertificate.ipfs_hash = pipfs_hash;
    objCertificate.added_by = msg.sender;
    objCertificate.added_timestamp = block.timestamp;

    objStudent.certificates[objStudent.qt_certificates] = objCertificate;
    objStudent.qt_certificates += 1;

    Organization storage objOrganization = _organizations[msg.sender];
    objOrganization.qt_certificates_issued += 1;

    _students[pstudent] = objStudent;
    _organizations[msg.sender] = objOrganization;
  }

  /*
    setStudent / setOrganization:
    Method to update the information of a organization or a student.
    Only the organization or the student can update their details.

    @param string pipfs_hash IPFS hash with the details of the student
  */
  function setStudent(string pipfs_hash) public payable
  {
    require(_users[msg.sender] == T_STUDENT);

    Student memory objStudent = _students[msg.sender];

    objStudent.ipfs_hash = pipfs_hash;
    objStudent.updated_timestamp = block.timestamp;

    _students[msg.sender] = objStudent;
  }
```

```
  function setOrganization(string pipfs_hash) public payable
  {
    require(_users[msg.sender] == T_ORGANIZATION);

    Organization memory objOrganization = _organizations[msg.sender];

    objOrganization.ipfs_hash = pipfs_hash;
    objOrganization.updated_timestamp = block.timestamp;

    _organizations[msg.sender] = objOrganization;
  }

  /*
    getCertificate:
    Method to get a specific certificate of a user.

    @param address pstudent Clau publica de l'alumne

    @return address Organization address.
    @return string  IPFS hash with the details of the certificate.
    @return uint256 Timestamp when the certificate was added.
  */
  function getCertificate(address pstudent, uint8 pcertificate) public constant returns (uint8, address,
string, uint256)
  {
    require(_users[pstudent] == T_STUDENT);
    Certificate memory objCertificate = _students[pstudent].certificates[pcertificate];
    return          (objCertificate.id,          objCertificate.added_by,          objCertificate.ipfs_hash,
objCertificate.added_timestamp);
  }
}
```

8.6. Importar claus a l'extensió Metamask

Per importar les claus a l'extensió Metamask primer s'ha de desbloquejar i posteriorment fer clic a la següent icona que obrirà una pàgina web on es gestionen les claus. Ara, en aquesta pàgina despleguem el menú fent clic al cercle situat a la part superior dreta i seleccionem l'opció "Import Account".

Figura 15: Pàgina web local de l'extensió Metamask

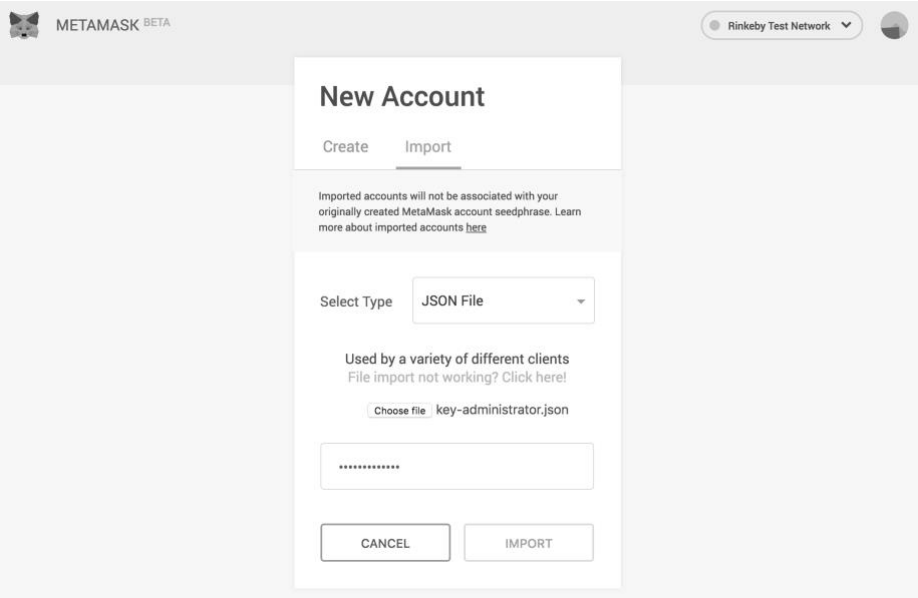


Font: Elaboració pròpia

Ja només falta indicar que el tipus de clau és un fitxer JSON, seleccionar el fitxer i introduir la contrasenya. Si tot és correcte l'extensió mostrarà el nou compte.

Recordar que les claus generades en el node local, els fitxers JSON es troben en el directori `~/.ethereum/keystore`.

Figura 16: Importació d'una clau a l'extensió Metamask



Font: Elaboració pròpia

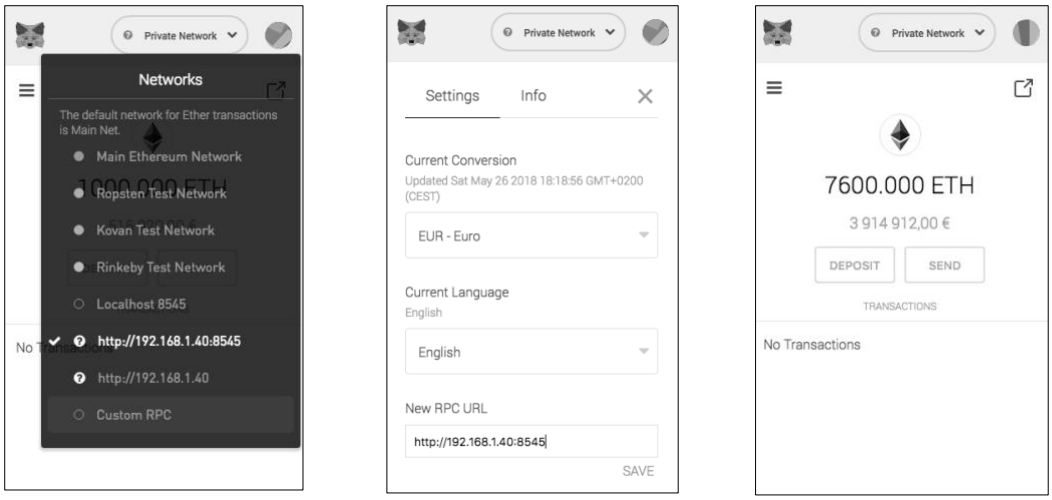
8.7. Connectar l'extensió Metamask a una xarxa Ethereum privada

Per connectar l'extensió Metamask a una altra xarxa d'Ethereum, es pot fer des del menú superior. Aquest menú per defecte ja incorpora diferents xarxes, des de la principal *(main net)* fins a altres de prova *(test net)* com ara Ropsten, Kovan i Rinkeby.

L'extensió també permet afegir una xarxa nova introduint l'adreça IP d'un node. Per afegir aquest nou node s'ha de fer clic a "Custom RPC", i en la pantalla carregada afegir l'IP en el camp "New RPC URL". Si l'extensió no s'hi pot connectar informarà a l'usuari mitjançant un error.

Una comprovació que es pot realitzar després de connectar-se al nostre node, és importar una clau generada i que s'ha afegit al bloc genesis i si tot es correcte veurem la quantitat d'Ethereum que disposa l'adreça.

Figura 17: Captures amb els passos per connectar l'extensió Metamask a una xarxa privada



Font: Elaboració pròpia

8.8. Fitxer de configuració del front-end

```
/**
 * @file config.js
 * @autor Daniel Atienza Lopez
 * @date 03-05-2018
 */

const config = {
	'version': '1.0.0',
	'mode': 'test', // production or test
	'production' : {
		'ethereum' : {
			'network_id' : 4, // Rinkeby test-net
			'network_name': 'Rinkeby',
			'etherscan': 'https://rinkeby.etherscan.io/{{type}}/{{id}}',
			'contract': {
				'address': '0x83fd4676d4505ce912b3fbd1a52c2b38a27398ce',
				'ABI':
[{"constant":true,"inputs":[],"name":"_qt_users","outputs":[{"name":"","type":"uint256"}],"payable":false,"
stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"pipfs_hash","type":"string"
}],"name":"setOrganization","outputs":[],"payable":true,"stateMutability":"payable","type":"function"},{"co
nstant":true,"inputs":[{"name":"","type":"address"}],"name":"_users","outputs":[{"name":"","type":"bytes1"}
],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"paddress
","type":"address"}],"name":"addUser","outputs":[],"payable":true,"stateMutability":"payable","type":"funct
ion"},{"constant":true,"inputs":[],"name":"_created_timestamp","outputs":[{"name":"","type":"uint256"}],"pa
yable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"pstudent","ty
pe":"address"},{"name":"pipfs_hash","type":"string"}],"name":"addCertificate","outputs":[],"payable":true,"
stateMutability":"payable","type":"function"},{"constant":true,"inputs":[],"name":"T_STUDENT","outputs":[{"
name":"","type":"bytes1"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"i
nputs":[{"name":"pipfs_hash","type":"string"}],"name":"setStudent","outputs":[],"payable":true,"stateMutabi
lity":"payable","type":"function"},{"constant":true,"inputs":[{"name":"","type":"address"}],"name":"_organi
zations","outputs":[{"name":"ipfs_hash","type":"string"},{"name":"qt_certificates_issued","type":"uint16"},
{"name":"added_by","type":"address"},{"name":"added_timestamp","type":"uint256"},{"name":"updated_timestamp
","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":
[],"name":"T_ADMIN","outputs":[{"name":"","type":"bytes1"}],"payable":false,"stateMutability":"view","type"
:"function"},{"constant":true,"inputs":[],"name":"T_ORGANIZATION","outputs":[{"name":"","type":"bytes1"}],"
payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"","type":"ad
dress"}],"name":"_students","outputs":[{"name":"ipfs_hash","type":"string"},{"name":"qt_certificates","type
":"uint8"},{"name":"added_by","type":"address"},{"name":"added_timestamp","type":"uint256"},{"name":"update
d_timestamp","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":tru
e,"inputs":[{"name":"pstudent","type":"address"},{"name":"pcertificate","type":"uint8"}],"name":"getCertifi
cate","outputs":[{"name":"","type":"uint8"},{"name":"","type":"address"},{"name":"","type":"string"},{"name
":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"input
s":[],"name":"_owner","outputs":[{"name":"","type":"address"}],"payable":false,"stateMutability":"view","ty
pe":"function"},{"constant":true,"inputs":[],"name":"VERSION","outputs":[{"name":"","type":"bytes10"}],"pay
able":false,"stateMutability":"view","type":"function"},{"inputs":[],"payable":true,"stateMutability":"paya
ble","type":"constructor"}],
			}
		},
		'ipfs' : {
			'protocol': 'https',
			'host': 'ipfs.infura.io',
			'port': 5001
		}
	},
	'test' : {
		'ethereum' : {
			'network_id' : 15, // Local node
			'network_name': 'Local',
			'contract': {
				'address': '0x6b623b003271bfe455b6f23b9abbf92a6e7968a4',
				'ABI':
[{"constant":true,"inputs":[],"name":"_qt_users","outputs":[{"name":"","type":"uint256"}],"payable":false,"
stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"pipfs_hash","type":"string"
}],"name":"setOrganization","outputs":[],"payable":true,"stateMutability":"payable","type":"function"},{"co
nstant":true,"inputs":[{"name":"","type":"address"}],"name":"_users","outputs":[{"name":"","type":"bytes1"}
],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"paddress
```

```
","type":"address"}],"name":"addUser","outputs":[],"payable":true,"stateMutability":"payable","type":"funct
ion"},{"constant":true,"inputs":[],"name":"_created_timestamp","outputs":[{"name":"","type":"uint256"}],"pa
yable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"pstudent","ty
pe":"address"},{"name":"pipfs_hash","type":"string"}],"name":"addCertificate","outputs":[],"payable":true,"
stateMutability":"payable","type":"function"},{"constant":true,"inputs":[],"name":"T_STUDENT","outputs":[{"
name":"","type":"bytes1"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"i
nputs":[{"name":"pipfs_hash","type":"string"}],"name":"setStudent","outputs":[],"payable":true,"stateMutabi
lity":"payable","type":"function"},{"constant":true,"inputs":[{"name":"","type":"address"}],"name":"_organi
zations","outputs":[{"name":"ipfs_hash","type":"string"},{"name":"qt_certificates_issued","type":"uint16"},
{"name":"added_by","type":"address"},{"name":"added_timestamp","type":"uint256"},{"name":"updated_timestamp
","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":
[],"name":"T_ADMIN","outputs":[{"name":"","type":"bytes1"}],"payable":false,"stateMutability":"view","type"
:"function"},{"constant":true,"inputs":[],"name":"T_ORGANIZATION","outputs":[{"name":"","type":"bytes1"}],"
payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"","type":"ad
dress"}],"name":"_students","outputs":[{"name":"ipfs_hash","type":"string"},{"name":"qt_certificates","type
":"uint8"},{"name":"added_by","type":"address"},{"name":"added_timestamp","type":"uint256"},{"name":"update
d_timestamp","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":tru
e,"inputs":[{"name":"pstudent","type":"address"},{"name":"pcertificate","type":"uint8"}],"name":"getCertifi
cate","outputs":[{"name":"","type":"uint8"},{"name":"","type":"address"},{"name":"","type":"string"},{"name
":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"input
s":[],"name":"_owner","outputs":[{"name":"","type":"address"}],"payable":false,"stateMutability":"view","ty
pe":"function"},{"constant":true,"inputs":[],"name":"VERSION","outputs":[{"name":"","type":"bytes10"}],"pay
able":false,"stateMutability":"view","type":"function"},{"inputs":[],"payable":true,"stateMutability":"paya
ble","type":"constructor"}],
                        }
                },
                'ipfs' : {
                        'protocol': 'http',
                        'host': '192.168.1.41',
                        'port': 5001,
                        'port_web': 8080
                }
        }
};
```