Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Implementació en Ethereum d'un programa de negociació de compensació de factures entre empreses

Document: Memòria

Alumne: Enrique Sambola Giménez

Tutor: Josep Lluís de la Rosa Esteve
Departament: EEEA
Àrea: ESA

Convocatòria (mes/any): Setembre 2022

Universitat de Girona
Escola Politècnica Superior

PROJECTE FI DE GRAU

Implementació en Ethereum d'un programa de negociació de compensació de factures entre empreses

Autor:
Enrique SAMBOLA GIMÉNEZ

Setembre 2022

Grau en Enginyeria Informàtica

Tutors:
Josep Lluís DE LA ROSA ESTEVE

Agraïments

Per començar vull agrair molt especialment al meu tutor del projecte, Pep Lluís, pel suport que m'ha donat sempre que he tingut problemes. També a la meva família i amics pel suport donat en els moments més complicats...

Índex

CAPÍTOL 1

Introducció, motivacions, propòsit i objectius del projecte

1.1 Introducció

El món està experimentant una revolució tecnològica mai vista: des d'enviar un missatge a una altra zona del món fins a poder diagnosticar una malaltia a un pacient en qüestió de segons o canviar el bec i la pala per una màquina molt més productiva. Això abans era impensable fa uns segles. Aquí doncs, apareix la tecnologia per solucionar problemes i millorar la vida de les persones. Aquest últim punt és clau per entendre aquesta nova era.

Un d'aquests problemes és la falta de liquiditat. Mitjans de comunicació afirmen que hi ha empreses que han hagut d'endarrerir pagaments a proveïdors [1] i que el 30% de les empreses tenien problemes de liquiditat abans de la pandèmia [2]. A més, segons un estudi del Banc d'Espanya [3], entre un 67% i 69% d'empreses espanyoles tindrian problemes de liquiditat durant aquesta. Una de les possibles causes és perquè tenen moltes despeses que venceran abans que hi arribin ingressos, per la situació econòmica global (pandèmia i inflació), despeses extraordinàries, etc..

Per tal de pal·liar això es poden liquidar les factures pendents (mig termini) de compradors amb l'ajuda d'un tercer. Això es coneix com a facturatge o *factoring* en anglès. Actualment hi ha companyies que fan *crowdfunding* de factures com [4] que ja aporten una solució tradicional. El propòsit d'aquest Projecte de Fi de Carrera és aquest, el de crear un programa descentralitzat de negociació de compensació entre empreses que sigui una solució nova o alternativa a aquest problema.

El projecte es pot entendre des de dos perspectives o blocs: una econòmica i altra tecnològica. La primera es podria considerar des del punt de vista d'usuari o d'inversor. En canvi, la segona és la més important, la informàtica, que és la que s'explicarà i detallarà més.

1.2 Motivacions

En aquesta secció explicaré perquè he decidit fer aquesta temàtica. Per això, enumeraré i explicaré les raons una a una:

Temes
Una de les meves principals inquietuds és ser un ignorant en certs tòpics. Sempre he tingut curiositat i m'he fet preguntes de com s'implementen i desenvolupen els sistemes que fan servir Blockchain. També entendre certs aspectes econòmics i el polèmic món dels criptoactius (criptomonedes, NFT, ...).

Nous coneixements
A la carrera no es profunditza molt en els temes que he comentat abans, potser algun treball o comentari puntual però no una assignatura com a tal. Entendre el funcionament de Blockchain i la programació i posada en marxa de contractes intel·ligents el veig una experiència molt enriquidora personal.

Inversió a futur
Com es detallarà en el treball, hi ha moltíssima llibertat i formes de fer en aquesta tecnologia. És molt complicat trobar la solució òptima. A més, tenir una petita base de Blockchain i saber fer una aplicació descentralitzada el veig com una bona inversió personal de cara a nous projectes.

Per concloure, he decidit escollir aquesta temàtica perquè en conjunt la veig molt profunda i enriquidora. A més, considero que agrupa molts tòpics que personalment són molt interessants: sistemes descentralitzats, economia, programació, contractes, criptomonedes, ...

1.3 Objectius

Els objectius principals d'aquest projecte són els següents:

- Desenvolupar un sistema de factoring amb l'ús de la tecnologia Blockchain.
- Comprendre i estudiar el sistema descentralitzat de Blockchain i els contractes intel·ligents.
- Desenvolupar, aplicar i desplegar contractes intel·ligents a una cadena de blocs.
- Tokenitzar actius amb Liquidity pools i/o protocols.
- Crear una petita interfície semblant a una aplicació descentralitzada.

CAPÍTOL 2

Viabilitat

Tot projecte perquè es pugui dur terme s'ha de fer una anàlisi de la seva viabilitat tant econòmica com tecnològica i dels seus recursos. En aquesta part, explicaré tot allò que he fet servir per poder fer el projecte.

2.1 Econòmica

La meva filosofia que m'he plantejat a l'inici del projecte era que no és necessari fer una despesa de diners gran en un treball centrat en cripto. Reitero que no he gastat ni un cèntim de la meva butxaca i que és possible programar i fer codi de contractes intel·ligents sense cap inversió inicial.

Qualsevol persona, independentment de la seva capacitat econòmica, pot fer el mateix que he fet jo en aquest projecte només disposant del material necessari que comentaré a la part tecnològica [2.2]. També, cal recalcar que part del projecte s'ha fet a un centre de recerca. Aquest es diu TECNIO Centre EASY i compte amb material de sobra i suficient perquè qualsevol persona pugui treballar perfectament.

2.2 Tecnològica

De cara a desenvolupar el projecte, sí que cal definir uns requisits mínims i recomanats. He fet una petita taula explicant cada element i un cost aproximat per persona. En la següent taula s'especifica el hardware mínim necessari:

Element	**Comentaris**	**Cost**
Ordinador	Fonamental per poder crear el codi i documentar el projecte, recomanable que tingui bona memòria en general	Variable: 300-1200€
Internet	Necessàri per poder conectar-nos a una Cadena de blocs i documentar el projecte, recomanable conexió per cable	30€/mes
Electricitat	Indispensable perquè el hardware pogui funcionar amb el temps	<10€/mes [**?**]

Taula 2.1: Elements indispensables per a fer el projecte

Com es pot veure només amb un ordinador ja n'és suficient. És recomanable que tingui bona memòria per ser augmentar la productivitat. Cal comentar que hi ha centres de recerca, com per exemple el TECNIO Centre EASY al Parc Científic de la UdG que tenen a la seva disposició d'això i molt més.

CAPÍTOL 3

Metodologia

La metodologia emprada en el projecte ha sigut Scrum de tipus Àgil. Aquesta consisteix a dividir el projecte en diferents períodes o també coneguts com a Sprints. Cadascun d'aquests també té les seves fases: anàlisi, planificació, revisió, disseny i posada en marxa.

Figura 3.1: Esquema del funcionament de la metodologia Agile, Scrum

Anàlisi

Durant aquesta fase es feia un estudi sobre el tema a desenvolupar i implementar a l'sprint basat en els objectius. Per exemple, em formava de com funcionava Blockchain, Solidity i les Liquidity Pool. Un cop que ja es tenia el concepte més clar, procedia a programar el codi. Durant el procés, m'ajudava d'eines amb Excel i per fer simulacions de com hauria de funcionar.

Al principi de tot el projecte, era una fase bastant llarga perquè és un tema completament nou per mi i no tenia cap experiència, però la resta de sprints un cop els conceptes ja estan més assimilats i entesos, eren més ràpids.

Planificació

En aquesta etapa, es detallaven quins requeriments i algorismes s'havien d'implementar i incorporar al codi. També la temporalització que es detallarà més al seu capítol 4

Per exemple, en incorporar la taxa d'interès al programari havia de comprovar i testejar que la resta del codi funcionés correctament perquè en un procediment s'havia afegit i podia quedar alterat.

Disseny i posada en marxa

Consistia en el desenvolupament i testeig del codi. Un cop acabat, es feia el desplegament del sprint. Tot seguit, parlava amb el tutor per parlar sobre dubtes i per començar la fase d'anàlisi del següent solucionant dubtes.

Revisió

Aquesta fase consistia simplement a tornar a comprovar el bon funcionament del codi en incorporar els nous canvis fets, només si modifica el seu comportament. També es feien alguns algorismes òptims.

CAPÍTOL 4

Planificació

Com s'ha comentat prèviament a Metodologia, el projecte s'ha anat desenvolupant durant 3 sprint. Aquí detallaré els continguts i la temporalització de cadascun. Voldria recalcar que les dates no són 100% precises i que es poden haver superposat algunes etapes.

Primer de tot, es va decidir el tema a finals de gener/principis de febrer de l'any 2022.

Sprint 1

El primer sprint va comprendre entre principis de febrer i mitjans d'abril. Concretament entre el 4/2/2022 fins al 19/4/2022.

Aquest va consistir en una gran fase d'anàlisi i de formació tant en Blockchain [5] [6] [7], finances descentralitzades [8], facturatge [9], Solidity [10] [11] i criptoactius [12] [13] [14]. La fase de disseny va concloure amb la creació de la moneda (més correctament, token) i entendre l'AMM que em va proporcionar el tutor a partir d'una sèrie d'exercicis i simulacions amb Excel.

Etapa	**Inici**	**Fi**
Anàlisi	4 de febrer	1 d'abril
Planificació	2 d'abril	3 d'abril
Disseny i posada en marxa	4 d'abril	18 d'abril
Revisió	[No feta]	[No feta]

Taula 4.1: Resum Sprint 1

Sprint 2

El segon sprint va comprendre entre mitjans d'abril i mitjans de juny. Concretament entre el 19/4/2022 fins al 10/6/2022.

Primerament, a l'etapa d'anàlisi em vaig centrar a fer la Liquidity Pool i aplicacions descentralitzades [15] [16]. La fase de disseny va concloure amb un verificador de documents i un AMM incorporat al sistema.

Etapa	**Inici**	**Fi**
Anàlisi	19 d'abril	1 de maig
Planificació	2 de maig	4 de maig
Disseny i posada en marxa	5 de maig	10 de juny
Revisió	8 de juny	10 de juny

Taula 4.2: Resum Sprint 2

Sprint 3

El tercer i últim sprint va comprendre entre mitjans d'juny i l'agost. Concretament entre el 10/6/2022 fins al 5/8/2022.

Aquest cop la fase d'anàlisi va ser més curta i vaig planificar adaptació a Truffle i Ganache, incorporar la interfície gràfica amb React.js, afegir l'interès i altres comissions i correcció d'errors.

Etapa	**Inici**	**Fi**
Anàlisi	10 de juny	17 de juny
Planificació	18 de juny	20 de juny
Disseny i posada en marxa	21 de juny	5 d'agost
Revisió	5 d'agost	20 d'agost

Taula 4.3: Resum Sprint 3

CAPÍTOL 5

Marc de treball i conceptes previ

Per tal d'entendre la totalitat del projecte, he decidit dividir aquest capítol en dos blocs. Un econòmic que explica conceptes bàsics i el més important, el tècnic, que explica el marc teòric de l'estructura del projecte.

5.1 Bloc 1: Econòmic

5.1.1 Conceptes

Abans de començar a parlar d'aspectes tècnics del projecte, crec que cal explicar alguns conceptes per persones que no estiguin tan basades en termes econòmics que són importants.

En aquest bloc detallaré tot allò relacionat amb l'estructura econòmica del projecte: l'actiu i passiu, la liquiditat, l'oferta i demanda i la inflació.

Actiu i passiu

Un **actiu**, en economia, es tracta d'un element que té un cert valor i que una persona o grup de persones n'és propietària. Aquests solen generar un rendiment o benefici (no cal que sigui econòmic) futur.

N'hi ha dues distincions comunes d'actius: el corrent i el no corrent. Es diferencien en el quan o no generen rendiment. El primer és a curt termini, alguns exemples d'aquests són l'efectiu, inversions, clients i d'altres. El segon és a llarg termini i es complicat de convertir-lo en diners. Alguns exemples són: maquinària, immobles, inversions a llarg termini i més.

Per altra banda, el **passiu**, és la contrapart de l'actiu. Es tracta de tota l'estructura financera que sosté els actius. Està formada pel patrimoni i els deutes i obligacions.

En termes més senzills i entenedors, podríem dir que **els actius són fons d'ingressos** (més diners) mentre que **els passius són fons de deute** (menys diners).

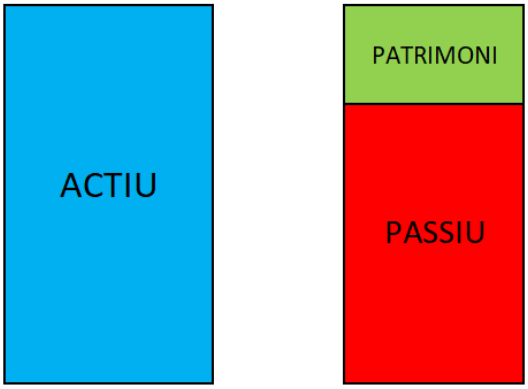


Figura 5.1: L'actiu i el passiu més patrimoni formen el balanç

Liquiditat

La liquiditat és el procés de convertir un actiu en diner efectiu de forma immediata sense que aquest pateixi una pèrdua de valor. És una analogia de passar d'estat sòlid a líquid, d'aquí ve el terme.

Cal dir que existeix un grau de liquiditat. El nivell màxim o absolut el té els diners en efectiu seguits pel de caixa. Per exemple, és fàcil canviar una moneda de 1 € per una altra igual. Per altra banda, els immobles o la maquinària, actius no corrents (Actiu i passiu), és més difícil liquidar-los i que no baixi el seu valor.

Llei de l'oferta i demanda

És un principi bàsic que consisteix que el preu d'un producte està relacionat amb l'oferta i la demanda d'aquest.

Així, segons el preu al mercat d'un producte o servei, els venedors estan disposats a fabricar una quantitat determinada d'aquest. Com també els clients estan disposats a comprar una quantitat determinada, segons el preu.

La intersecció d'aquestes dues quantitats s'anomena punt d'equilibri. Això vol dir que els clients volen comprar la mateixa quantitat que els venedors posen a disposició per un preu.

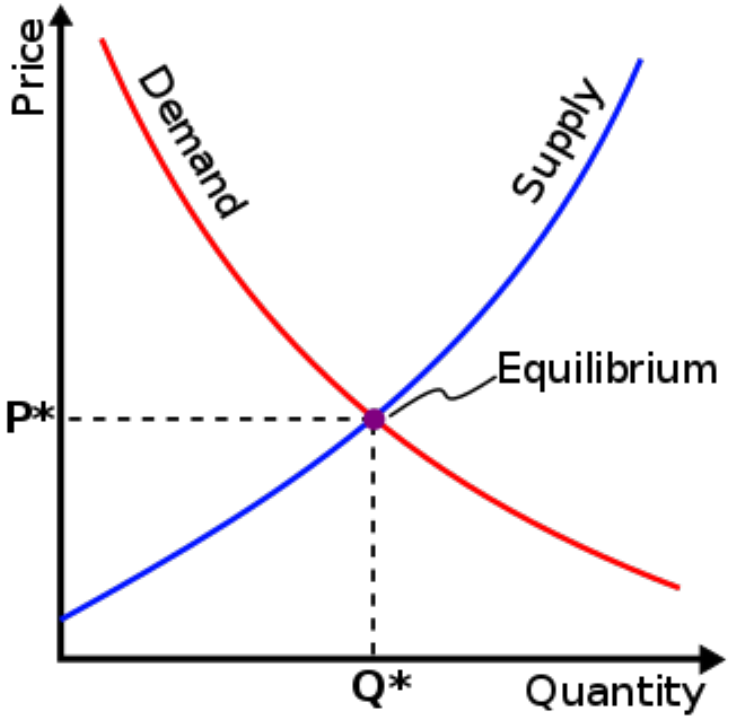


Figura 5.2: Gràfic il·lustratiu de la llei d'oferta i la demanda

Segons la teoria i com es pot observar al gràfic, la llei de la demanda diu que si tot es manté constant (no influeix cap altre variable), la quantitat demandada disminueix quan el preu del producte augmenta. En canvi, la llei de l'oferta indica que, sota les mateixes condicions d'abans, l'estoc augmenta quan el preu puja.

Inflació

La inflació és un fenomen monetari que consisteix en un augment generalitzat dels preus. A causa d'això, el poder adquisitiu de la moneda en qüestió disminueix. Les causes que la generen estan íntimament lligades a l'oferta i la demanda de la divisa.

Per tant, podem tenir els següents casos:

1. Augment de l'oferta i la demanda, el valor romandrà constant.
2. Augment de l'oferta però no de la demanda, el valor de la divisa disminuirà.
3. Disminució de l'oferta i la demanda, el valor serà constant.
4. Disminució de l'oferta però no de la demanda, el valor serà constant.

Un dels principals problemes que té la inflació és que pot entrar en un cicle viciós difícil d'aturar si no es combat en el curt-mig termini:

Quan es produeix un augment de preus per causes externes, els consumidors, en notar la pèrdua de poder adquisitiu, demanen als seus llocs de treball un augment de sou. Un cop aquí, l'empresa per finançar l'augment, pot decidir pujar els preus dels productes/serveis que ofereix. En tornar a pujar els preus, els consumidors tornen a demanar l'augment i així successivament.

Cal recalcar que si existeix l'augment generalitzat dels preus, també la seva contrapart, la deflació. A més, aquesta pot ser igual de problemàtica que la inflació.

5.1.2 Facturatge

El facturatge, conegut en anglès com a factoring, és un mecanisme alternatiu disponible per a les empreses amb el qual liquidar documents de cobrament com factures, rebuts, lletres, pagarés, garanties, etcètera.

Tradicionalment, hi existeixen tres parts: l'empresa emissora, la de factoring i la deutora. La segona compra els documents a l'emissora a canvi d'una comissió. La tercera és la que ha de fer efectiu el pagament a l'emissora en un temps.

L'esquema és el següent:

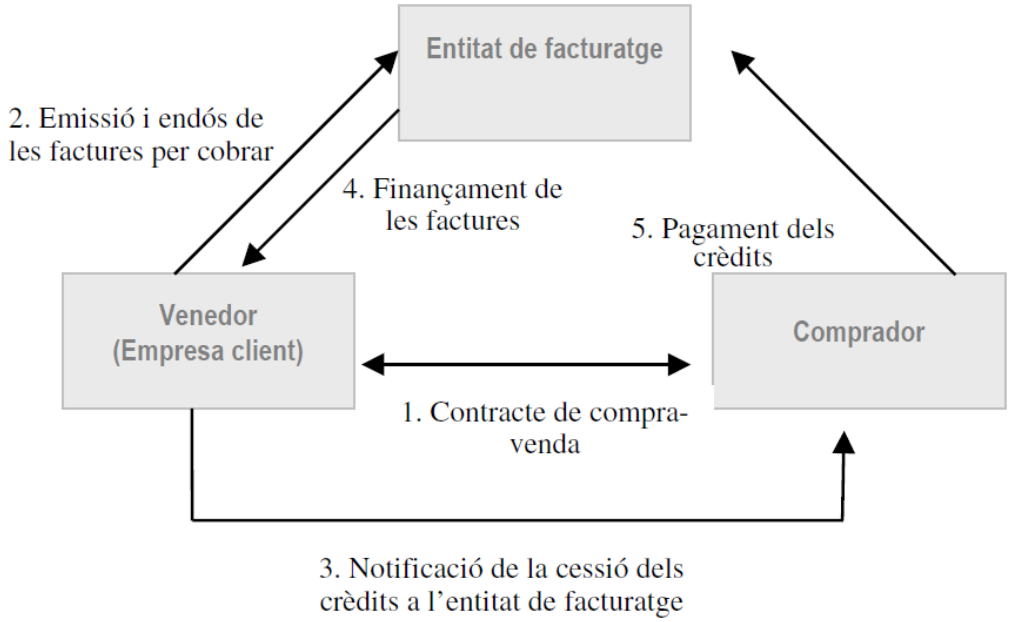


Figura 5.3: Funcionament del procés de facturatge tradicional

Un detall a tenir en compte és qui es fa càrrec del risc de l'operació. L'empresa de facturatge pot assumir el risc, això pot implicar un encariment de la comissió, o directament l'emissora. En aquest cas, la resolució del conflicte estaria en mans de la justícia.

Evidentment, aquest mecanisme té avantatges i inconvenients. Proporciona una ràpida liquiditat a l'empresa emissora així permetent la viabilitat i continuïtat del negoci. Per altra banda, l'empresa de factoring pot demanar una comissió elevada per a l'assumpció del risc o directament només acceptar documents d'empreses solvents.

La idea del treball és la de desenvolupar una aplicació descentralitzada que faci servir aquest mecanisme però eliminant la confiança dipositada en un tercer, l'empresa de factoring, per la fiabilitat i seguretat de Blockchain. És a dir, aplicar una solució no centralitzada.

5.1.3 DeFi

De l'acrònim Decentraliced Finances, com el seu nom indica es tracta de finances a partir d'aplicacions descentralitzades. És un nou protocol monetari que utilitza Blockchain per a fer tan inversions tradicionals com novedoses amb actius.

Els casos més habituals de DeFis són donar i sol·licitar préstecs, la més coneguda és AAVE; intercanvi de divises (exchanges o dexes) com fa Coinbase i la que més ens interessa proporcionar liquiditat amb Liquidity Pools.

Liquidity pools

També conegudes en català com a fons de liquiditat, és una de les aplicacions més populars en DeFi. Al cap i la fi, són fons d'intercanvis de divises, però molt més pràctiques, ja que no hi ha el broker, un intermediari.

En les finances descentralitzades, són fons de liquiditat de dues divises en el que es pot dipositar o treure d'aquestes dues. El valor per intercanviar una moneda o altra variarà en funció de la seva oferta i demanda. Això està regit per una fórmula matemàtica en la qual el producte de les quantitats de les dues divises ha de romandre constant. Els contractes intel·ligents que implementen això s'anonemen Automated Market Maker (AMM).

Normalment, la fórmula sol ser senzilla i exchanges coneguts com Uniswap fan servir aquesta:

$$x \cdot y = k \tag{5.1}$$

Però a partir d'aquí hi ha moltes variants tant de més complexes com de simples que implementen altres AMM. Habitualment sol haver-hi una ràtio de 50:50 entre els dos actius. Per exemple: inicialment si en una pool dòlars-ether vols retirar 500$, hauràs de fer un dipòsit equivalent per mantenir la ràtio. A més a més, per cada nova comanda anirà augmentant el preu. Més en detall:

Es pot veure que a mesura que traiem dòlars, el cost de l'altre actiu va augmentant. Per evitar que la relació $\frac{x}{y}$ es distorsioni (no confondre amb $x \cdot y$ que

Actiu X ($)	Actiu Y (ETH)	K	Tret de X ($)	Cost en Y (ETH)	X/Y
500	250,00	125000	0	0	2,00
499	250,50	125000	1	0,50	1,99
498	251,00	125000	1	1,00	1,98
497	251,51	125000	1	1,51	1,98
496	252,02	125000	1	2,02	1,97
495	252,53	125000	1	2,53	1,96
494	253,04	125000	1	3,04	1,95

Figura 5.4: Evolució dels preus en una Liquidity Pool

és la constant k), es necessiten uns àrbitres que tinguin la responsabilitat de mantenir aquesta relació. Aquests simplement haurien de fer l'acció oposada: si una persona fa un intercanvi d'X per Y, l'àrbitre farà un intercanvi d'Y per X. Òbviament aquestes persones no fan això caritativament, sinó que se sol fer un incentiu. Aquest sol ser una fee per cada transacció.

Tokenització

És el procés de segmentar o dividir un actiu en tokens que la suma total d'aquests tenen el mateix valor. En concret, és la creació de tokens digitals que representen la propietat d'actius. També s'anomena Security Token Offering. És molt similar a les accions d'una empresa a la borsa, però se centra més en un actiu que no pas en l'empresa en conjunt. Això té moltíssimes aplicacions.

Per exemple, si volem vendre un edifici per 500.000 € només un grup reduït d'inversors tenen la capacitat de pagar aquesta quantitat. Amb la tokenització, l'immoble es digitalitza i es divideix en petits blocs en forma de tokens. L'edifici es pot dividir en 10.000 blocs. Gràcies a això, tens 10.000 tokens que tenen un valor de 50 € cadascun i qualsevol pot invertir. Quan es vengui o llogui, els inversors obtindran beneficis corresponent al bloc.

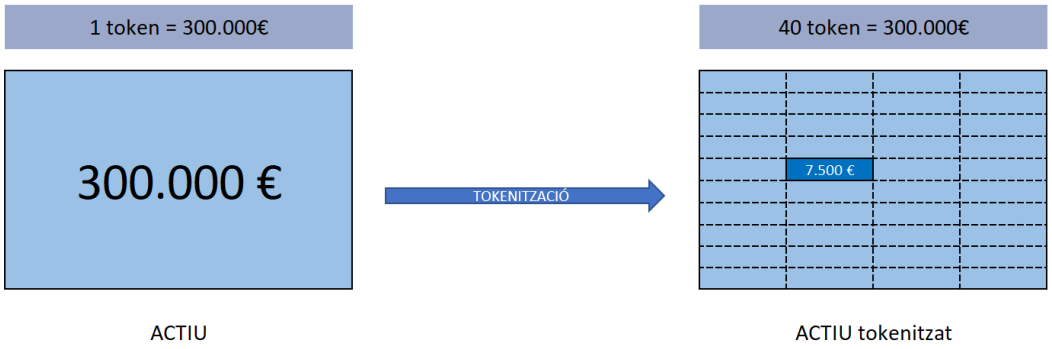


Figura 5.5: Exemple il·lustratiu de tokenitzar un actiu

Col·lateralització

És el procés de donar un actiu en cas de garantia que proporcioni confiança en una futura transacció. Aquest pagament, es diu en anglés *col·lateral*. És important perquè mitiga el risk en cas que no es faci efecti el pagament. Igualment, hi ha diversos graus en el que es pot reduir l'impacte. Per exemple, el terme "altament col·lateralitzat"vol dir que a darrera hi ha un actiu com salvaguarda que compensa les pèrdues, però no totes. Per altra banda, hi ha els "sobrecol·lateralitzats"que proporcionen un valor de més del estipulat en la transacció donant molta confiança. Per exemple, donar un col·lateral de 110.000€ a un prèstec de 100.000€. És una bona pràctica per millorar l'historial creditici. El projecte té mecanismes que incentiven aquesta pràctica.

Tokenomics

Els tokenomics és un anglicisme que junta les paraules *token* i *economics*, això ja ens diu que es tracta de l'economia dels tokens. Més ben dit, de l'estructura econòmica d'una criptomoneda. És fonamental per entendre la viabilitat d'una criptomoneda. Podríem dir que és l'aplicació de la funció empresarial, és a dir, quin impacte té i el seu valor. També, és molt semblant a les polítiques monetàries que fa un banc central d'increment o reducció d'oferta, promoure la despesa o l'estalvi, etc.. A diferència dels bancs, tot això s'implementa en codi fent-lo més transparent, predictible i difícil de canviar.

Aquests són els factors que més ajuden a definir l'estructura econòmica d'una moneda:

- Oferta: com ja hem dit al 5.1.1 és decisiu per saber l'impacte en el preu. Cal diferenciar entre oferta total i en circulació. La primera pot tenir un màxim o ser indefinit. La segona indica quantes monedes hi ha disponibles actualment. Ha de respondre a aquestes preguntes: quantes monedes n'hi ha? Quantes s'afegeixen o es treuen?
- Valor: es tracta de la funció empresarial o els casos d'ús. Quina utilitat té la moneda? Per què serveix? Què fa exactament?
- Distribució: un cop que tenim els tokens creats, qui s'encarrega d'oferir-les al públic i com ho fa?
- Demanda: és important que hi hagi demanda de la moneda en el temps per això es defineixen uns mecanismes que ho incentivin.

Al capítol de requisits del sistema [6] estan definits els tokenomics del projecte.

5.2 Bloc 2: Tecnològic

5.2.1 Blockchain

Què és?

La tecnologia Blockchain, en català cadena de blocs, va sorgir l'any 2009 creat per una persona (o grup) amb el pseudònim de Satoshi Nakamoto [17]. Resumidament, aquesta consisteix en una estructura de dades conformada per blocs encadenats uns amb els altres pel seu hash i és pràcticament immutable.

És semblant a un enorme llibre comptable on queden enregistrades totes les transaccions realitzades en tota la xarxa. És a dir, per a garantir la seguretat i veracitat dels pagaments, aquestes transaccions queden guardades i a l'abast de tothom. Així, Blockchain funciona com un registre públic inalterable consensuat pels seus usuaris. És el coneixement consensuat de les transaccions, transparents per tothom, qui li donarà validesa.

Per què Blockchain?

Un dels principals eixos que mou la tecnologia Blockchain és la no-intervenció de terceres parts en les transaccions entre dos individus o entitats. Els sistemes actuals bancaris i burocràtics es basen en la confiança que tenen les persones a les seves institucions, ja que en cas que hi hagi algun problema o inconvenient, un tercer intervindrà per solucionar-ho. La gran innovació de Blockchain és que és el mateix sistema basat en la seva seguretat i immutabilitat el que permet que no siguin necessaris intermediaris.

La tecnologia aconsegueix assolir aquesta seguretat gràcies a la combinació de dos mecanismes: el sistema descentralitzat i un mecanisme de consens. El primer consisteix a proporcionar una còpia a tothom dins la xarxa i que aquesta es mantingui actualitzada cada cop que es fa una transacció.

El principal risc a les Blockchain, són els atacs del 51%. Personalment, jo l'associo amb la política quan un partit tirànic té la majoria absoluta a un parlament o un dictador totalitari. Aquests eliminen adversaris, canvien les lleis i fan tot allò que faci falta per establir-se al poder. L'analogia és igual: quan un node o un conjunt d'aquests tenen control total de més del 50% (en aquest cas de potència de la xarxa), tenen tot el poder de decisió i no hi ha consens possible. És a dir, tenen control de la majoria de la potència de hashes. Això implica que un atacant té la capacitat de modificar, revertir i denegar transaccions del

registre maliciosament. També pot evitar que els miners rebin la recompensa al minar un bloc obtenint així el monopoli. Disposar d'aquest 50% es considera un atac a la integritat de la cadena de blocs, la qual es basa en la confiança en el seu sistema.

Consens

Prova de treball

Conegut també com a Proof of Work (PoW). Es tracta d'un trencaclosques criptogràfic que té associat un cost computacional per cada node participant, amb això si algú pretén suplantar un terminal, haurà d'assumir aquest cost.

Els puzles es basen en els hashes, funcions que per cada entrada donen un únic resultat i no a la inversa. Notar que una mínima modificació a l'entrada torna un resultat completament diferent. La més comuna i més utilitzada és SHA-256.

Aleshores, a cada bloc i un número se li aplica una funció de hash. Per tal de resoldre el trencaclosques, el resultat ha de tenir una seqüència de zeros a l'inici. Sembla un problema senzill, però gràcies als canvis abruptes amb lleugeres modificacions a l'input del hash, es deu provar per força bruta fins a trobar la solució. Això requereix un temps mitjà i de recursos molt elevat. Qui ho hagi trobat primer tindrà el dret d'afegir el bloc al registre general i compartirà el resultat amb la resta de la xarxa. En això consisteix la mineria de blocs.

Prova de participació

També s'anomena Proof of Stake (PoS). És un algorisme de consens basat en el fet que cada node per tal de poder participar, ha de demostrar que té un actiu econòmic. Els blocs es minen, però en aquesta prova es sol dir que el bloc està "forjat". Cada node té una garantia econòmica associada per cada bloc minat, és a dir, hi ha un actiu que el sosté. Aquesta garantia també es coneix com a col·lateral. En la prova, es diu stake.

Els usuaris per poder participar en el procés han de deixar bloquejada una certa quantitat de l'actiu. Com més quantitat s'aporti, més probable és que el node sigui validador. Per evitar que els més rics s'apropiïn de la xarxa, existeixen dos mètodes: aleatori i per edat. En el primer són seleccionats els nodes amb un hash baix i un stake elevat. En canvi, en el segon hi ha un càlcul de l'antiguitat de l'actiu en la xarxa basat en el temps i la quantitat de monedes, quan arriba a zero es torna a inicialitzar el temps.

A l'hora de validar un bloc, es comproven si les transaccions són correctes. Un cop fet, se signa i s'afegeix a la Blockchain. Per aquesta feina, el node rep una comissió per transacció com a recompensa com per exemple monedes. Les Blockchains més conegudes que utilitzen aquest mecanisme són Solana, Avalanche i Polkadot. Es preveu també que Ethereum 2.0 segueixi el model de Proof of Stake.

Diferències

Actualment, el mercat està apostant fortament pel Proof of Stake enlloc de la Proof of Work. Principalment, hi ha una diferència en el cost. Mentre que PoS només s'ha de demostrar la possessió d'un actiu, PoW ha de demostrar la capacitat computacional i això implica un consum energètic molt gran. En canvi, a la prova de participació s'ha de disposar d'una inversió inicial mentre que a la prova de treball la inversió és en maquinària.

Aquesta taula treta de Binance [18] explica molt bé les diferències entre les dues proves de consens més conegudes:

	Proof of Work (PoW)	Proof of Stake (PoS)
Equipo necesario	Equipo de minería	Cantidad mínima o ninguna
Consumo de energía	Elevado	Bajo
Tendencia hacia	Centralización	Descentralización
Método de validación	Prueba informática	Staking de monedas

Figura 5.6: Prova de treball vs Prova de participació

Respecte a la disposició d'una còpia del registre a cada node i la competència dels miners donen seguretat impedint que les persones amb intencions malicioses assumeixin el cost-benefici del seu acte, ja que aquest serà injust i completament inassumible. Si una persona fa una modificació al registre hauria de competir contra tota la xarxa de miners per tenir-lo actualitzat, ja que es té en compte la cadena més llarga (a partir de 6 ja es considera segur) entre tots els nodes i aquest, al llarg termini quedaria obsolet. Aquests mecanismes també resolen problemes de sincronització entre nodes.

Tipus

Hem estat parlant sobre Blockchain, però no totes tenen les mateixes característiques. Això és perquè n'hi ha de tres tipus: pública, privada i consorciades. És una tipologia en la qual es mesura el grau d'accés i control sobre les dades a la cadena de blocs. No confondre amb el finançament.

Públiques
És la Blockchain que tothom fa referència sempre. Destaca per: la immutabilitat de la cadena, ningú té el control absolut, qualsevol pot accedir al contingut, assegurança de la protecció de les dades i tots els nodes tenen el mateix poder de decisió. Les més conegudes són Bitcoin, Ethereum, Solana, Ardano, etc..

Privades

Són cadenes de blocs que es fan servir per protegir dades confidencials. Usualment, hi ha una tercera persona que s'encarrega d'administrar la Blockchain afegint nous nodes o denegar accés a aquesta. Tècnicament, no són descentralitzades i actua més com una base de dades. És habitual que tinguin aquest tipus empreses privades com per exemple HyperLedger.

Híbrides

Són una barreja de les anteriors. Existeix una part de la xarxa en la qual els nodes són privats i una altra en la que són públiques. El seu funcionament és amb permisos. El grau d'accés es basarà en els permisos que tingui l'usuari. N'hi ha un administrador que proporciona els nivells de seguretat. La més coneguda és Ripple.

PÚBLIQUES	**PRIVADES**	**HÍBRIDES**
Cap control	Control per un administrador	Cap control nodes públics
Immutabilitat de la cadena	Cadena modificable	Immutabilitat de la cadena
Qualsevol pot accedir al contigut	Només si té permís	Només si té permís
Qualsevol pot participar	Només per invitació	Qualsevol pot participar
Mateix poder de decisió	Depèn de l'administrador	Depèn dels permisos
Bitcoin	HyperLedger	Ripple

Taula 5.1: Taula comparativa dels tipus

5.2.2 Contractes intel·ligents

Què són?

Els contractes intel·ligents, també coneguts com a Smart Contracts, és codi desplegat a la Blockchain que estipula i executa un acord entre parts. La majoria estan escrits a la xarxa d'Ethereum en el llenguatge Solidity. També n'hi ha altres com Solana codificats amb Rust.

Per definició, els Smart Contracts són immutables, és a dir, no poden ser modificats. Això es deu que són minats a la Blockchain i a aquesta no es poden fer alteracions en els blocs. Habitualment es crea un de nou i s'adreça als usuaris a utilitzar-lo i no a l'antic.

Són distribuïts i no poden haver discrepàncies en allò estipulat al contracte. Gràcies al fet que el codi està descentralitzat, l'acord és públic i tothom el pot consultar a la xarxa. És a dir, estan dissenyats per evitar errors i conflictes entre persones.

Per tal de poder saber i verificar les condicions, es fan servir oracles. És una tercera part de confiança que proporciona informació. Actua com a un intermediari entre Blockchain i el món real. Perquè siguin fiables, han de ser descentralitzats, ja que una única font no ho és, per tant, els oracles també són Blockchains. El més usat és Chainlink.

Les aplicacions són diverses i infinites: préstecs flashs, assegurances, intercanvi de tokens, comprar propietats, loteries, apostes, valor d'accions, DAOs, etc..

Solidity

És el llenguatge de programació específicament creat per programar contractes intel·ligents desenvolupat per Ethereum. És fortament tipat, orientat a objectes i està clarament influenciat en llenguatges com JavaScript, C++ o Python. Els fitxers en Solidity tenen la terminació .sol.

Per Solidity, un contracte no és res més que un conjunt de codi (funcions) i dades (estat) que s'ubiquen a una direcció a la Blockchain d'Ethereum. Es pot pensar com un trosset d'una base de dades a la qual es poden fer consultes i modificacions mitjançant funcions que permeten gestionar la base de dades.

El codi s'executa en màquines virtuals d'Ethereum, EVM, que funciona sobre la seva Blockchain. Aquest es pot programar i compilar localment des de qualsevol dispositiu i després es desplega (deploy) per la xarxa d'Ethereum de forma descentralitzada. És molt habitual utilitzar l'entorn online Remix IDE on es poden desplegar contractes amb comptes de prova.

Sempre es comença indicant la llicència a la qual està el codi. Això s'indica amb dues barres diagonals a la primera línia de codi (//). Normalment, són llicències GPL d'ús lliure. Tot seguit s'especifica la versió de Solidity amb la que s'està programant. S'indica amb "pragma solidity"més la versió o entre dos (>= antiga < nova). Finalment, va la resta amb la definició del contracte amb els seus mètodes (funcions) i atributs (estat).

Ethereum Virtual Machine
L'entorn d'execució, EVM, està completament aïllat, no té accés a altres recursos com si tenen altres llenguatges com per exemple el sistema de fitxers, la xarxa, etc.. Són tan restringits que fins i tot dos contractes poden tenir accés limitat entre ells.

A Ethereum, hi existeixen dos tipus de comptes que comparteixen la mateixa direcció: les controlades per persones, els comptes externs, i els comptes interns o del contracte controlats directament pel codi. L'adreça d'un compte extern prové de la clau pública de l'usuari mentre que la interna depèn d'altres variables com el moment de creació del contracte, nombre de transaccions, etc.. Els dos tipus són tractats d'igual manera per l'EVM.

Cada compte té un diccionari on la clau i valor són diferents paraules de 256 bits. Aquest s'anomena storage. A més, tenen un saldo en Ethers (més concretament en una magnitud més molt més petita anomenada Wei) que es modifica al fer transaccions des del compte. Les transaccions són missatges entre dos contractes que inclouen una quantitat d'Ethers i unes possibles dades.

Des de la seva creació, cada transacció té associat un cost que s'anomena gas i només el paga qui ha originat la transacció. Les úniques funcions que no tenen gas són aquelles per consultar l'estat del contracte. Aquestes tenen el terme view. Es fa servir aquest mecanisme per donar un incentiu econòmic i compensar a les persones que permeten executar a les seves màquines l'EVM.

Per poder emmagatzemar les dades, l'EVM té tres àrees concretes: el ja mencionat storage, memòria i una pila. Es diferencien en el temps de consulta, límit d'accés, expansió, mida de paraules i més.

5.2.3 Criptoactius

Definirem com a criptoactiu a tota aquella propietat digital que es pot obtenir a partir d'un sistema descentralitzat. Això inclou les criptomonedes, tokens fungibles i tokens no fungibles. Per definició, un element fungible és aquell que una persona obté, consumeix i desapareix i a més és fàcilment reemplaçable per un altre nou. En canvi, els no fungibles són únics i no són consumibles.

Criptomonedes

Són actius digitals líquids que es fan servir com a pagament descentralitzat de productes i serveis en el món virtual. També actuen com a reserva de valor i estalvi. Les monedes més conegudes són Bitcoin, Ethereum, Dogecoin, etc..

Aquestes monedes tenen un principal problema i és que per ser pròpiament una divisa aquesta ha de tenir un valor estable amb el temps. Aquí un exemple per contrastar com fluctua el valor d'Ether, la criptomoneda d'Ethereum, comparat amb el Dòlar en només un dia [19]:

Figura 5.7: Relació de valor entre Dòlar USD amb ETH

Es pot veure que en els últims 3 mesos, el valor màxim ha estat 2145,71 el 15/5/2022 mentre que el mínim, en canvi, ha estat de 993,64 el 18/6/2022. Si es mira més enllà, es pot veure que no és un cas aïllat. N'hi ha hagut bastants canvis abruptes.

Per altra banda, si ho comparem amb l'Euro, es pot observar que el valor oscil·la entre centèsimes, gairebé amb un valor constant d'1.

Figura 5.8: Relació de valor entre Dòlar USD amb l'Euro

Aquesta volatilitat de les cryptocurrencies és deu a problemes vinculats a l'oferta i demanda de la moneda. Per una banda, no és àmpliament utilitzada per l'adquisició de béns o serveis comparada amb les monedes fiat i només 2 països la tenen com a moneda de curs legal.

Això vol dir que no té la confiança dels governs centrals, ja que probablement va en contra dels seus interessos. A diferència d'aquestes, tenen un banc central que controla la política monetària. Bitcoin, per exemple, només te planificat un màxim de 21 milions d'unitats en circulació. A més, hi ha influencers i wallets amb molt de poder (whales) que especulen i mouen el mercat en funció del sentiment i el curtplacisme.

Davant d'aquest problema, el mercat ja ha trobat una solució: les stablecoins. Són monedes que tenen un propòsit clar d'estabilitzar el seu valor respecte a un actiu de referència. Normalment, aquest sol ser una moneda fiduciària com per exemple el USD Dòlar ($), el Euro (€) o el Yuan (¥). És a dir, l'objectiu primordial de les monedes estables és que una unitat de valor d'una stablecoin vinculada al dòlar cotitzi al mercat amb un preu estable d'una unitat de dòlar.

Tanmateix, si observem la fluctuació del valor de Tether, una stablecoin, amb el dòlar es veu que no és volàtil:

Figura 5.9: Relació de valor entre Dòlar USD amb USD Tether

Per tant, les stablecoin permeten actuar com a actius refugi en cas que hi hagi canvi sobtats en el valor d'una criptomoneda. Són un instrument per l'entrada i sortida entre criptoactius sense haver de liquidar el valor directament al món real, ja que el cost de transacció per passar entre Blockchain al model tradicional és elevat i lent.

Les stablecoin que estan recolzades per un actiu en el món real, s'anomenen stablecoin col·lateralitzades. Prové de l'anglès, col·lateral, que significa garantia. Això vol dir que hi ha un actiu que actua com a garantia del seu valor. Aquest terme està molt present en el món crypto.

Per altra banda, també existeixen stablecoin que estan sostingudes pel funcionament d'un algorisme que controla l'oferta monetària d'aquestes. Són monedes estables algorítmiques o *algorishmic stablecoin*. Bàsicament, afegeixen o treuen massa monetària en funció de l'oferta i demanda per evitar casos d'inflació o deflació.

És important saber i ser conscient que tota inversió té el seu risc. Les monedes algorítmiques tenen un problema crític i és que no estan col·lateralitzades. Això vol dir que depenen exclusivament de la fiabilitat del funcionament del seu algorisme i del mercat. Un dels casos més coneguts és la caiguda de la stablecoin TERRA USD i la criptomoneda LUNA al maig del 2022. Aquesta es considerava una de les criptomonedes més famoses i va tenir caigudes de 44% per part de TERRA USD i del 99% per LUNA. Per més informació recomano aquest article [20].

Tokens fungibles

Són actius digitals descentralitzades que emet una entitat privada. Aquestes donen la funcionalitat i el valor que considerin. L'exemple més comú és compararlos amb les fitxes del casino; un cop intercanviat els diners per les fitxes, pots accedir, jugar i fer apostes per tot l'establiment allà on vulguis com al pòquer, ruleta, escurabutxaques. Altres analogies serien els tiquets a les fires o vals regal a una botiga. És a dir, van més enllà que una divisa. L'estàndard de token més conegut és l'ERC-20 d'Ethereum.

Quina diferencia real hi ha entre un token i una moneda? Una moneda fa servir la seva pròpia xarxa de Blockchain per estar al corrent de la informació. En canvi, un token fa servir la Blockchain d'una moneda d'un altre com a infraestructura. Es pot comparar amb un lloguer i una compra: amb el lloguer no som propietaris, però no ens preocupem de despeses de manteniment i altres, mentre que una compra sí som ple propietaris.

Entre les diverses aplicacions més comunes dels tokens fungibles, hi ha els següents:

- Platform Tokens: intercanviar tokens d'una mateixa Blockchain. Un exemple seria Uniswap.
- Security Tokens: representen el valor d'un actiu mitjançant un col·lateral. Actuen com a reserva de valor. USD Coin o USD Tether són un exemple.
- Transactional Tokens: forma ràpida de fer transferències de diners, com per exemple xDAI.
- Governance Tokens: són semblants a un vot, que aquest et permet participar en la presa de decisions en aplicacions descentralitzades.
- Utility Tokens: fan de moneda de canvi per un servei dins l'empresa que fa l'emissió. Tenen una intenció comercial.

ERC-20

És un estàndard de token que implementa una API per a tokens fungibles dins de contractes intel·ligents. Podríem dir que és la plantilla en la qual es basen la gran majoria de tokens fungibles. És àmpliament utilitzat i es pot comprovar quants hi existeixen amb Etherscan.

Aquest estàndard indica les diferents funcionalitats que ha de tenir un token ERC-20 perquè es pugui considerar com a tal, com per exemple: veure el saldo actual, subministrament total disponible o donar permís perquè un tercer pugui administrar una certa quantitat de tokens d'un compte.

Més detalladament, es pot considerar com a token ERC-20 a tot aquell contracte que:

1. Tingui un nom identificador i un símbol associat perquè es puguin diferenciar a la Blockchain d'Ethereum de la resta.
2. Tingui la capacitat de controlar la seva emissió com per exemple subministre total o la precisió en els decimals.
3. Té una interfície amb la qual es pot consultar el balanç total dels fons dels propietaris.
4. Pot controlar el sistema de transferències
5. Aprovisionar fons prèviament amb permís del propietari

Tot això en llenguatge *Solidity* és el següent :

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256
balance)
function transfer(address _to, uint256 _value) public returns
(bool success)
function transferFrom(address _from, address _to, uint256
_value) public returns (bool success)
function approve(address _spender, uint256 _value) public
returns (bool success)
function allowance(address _owner, address _spender) public
view returns (uint256 remaining)
```

Figura 5.10: Estàndar ERC-20 en *Solidity* [12]

Amb això qualsevol persona pot crear, programar i implementar la seva pròpia criptomoneda. Tècnicament, no són monedes pròpies sinó que es fa servir la Blockchain d'Ethereum. Com ja hem comentat prèviament, per ser estrictament monedes han de tenir la seva infraestructura amb la seva corresponent cadena de blocs.

Tokens no fungibles

Són actius digitals únics que es generen a partir d'un altre element com per exemple art, imatges, vídeos, etc.. En essència, són un trosset de dades que apunten a un servidor on està l'element original el qual una persona n'és propietària. Aquest trosset és simplement l'aplicació d'una funció de hash (SHA-256) a l'element digital de referència.

Els NFT representen un actiu únic i pot ser tan digital com versions tokenitzades del món real. A causa del seu caràcter no-fungible, està íntimament relacionat amb el col·leccionisme, però també n'hi ha altres aplicacions com l'autenticitat i propietat a l'àmbit digital. Cal recalcar que l'adquisició només és nominal, és a dir, no té cap efecte pràctic. L'ERC-721 és el més popular i també és d'Ethereum.

ERC-721

Igual que l'ERC-20, és un estàndard amb el qual poder implementar tokens no fungibles. A diferència de l'ERC-20, els NFT són únics i poden tenir valor diferent dos tokens en el mateix contracte. És a dir, és equivalent intercanviar dos tokens ERC-20, però no és el mateix en el cas de l'ERC-721 (a no ser que per casualitat tinguin el mateix valor).

La manera de poder fer això és afegint un identificador únic a un token. D'aquesta forma, li permetria ser diferent dels altres, és a dir, no fungible. En essència, un NFT és això.

Per tal de poder considerar a un contracte intel·ligent com a NFT, ha de complir els següents requisits:

1. Cada token ha de tenir un nom i un símbol identificatiu.
2. Poder consultar el balanç de tokens d'un compte i el suministre total.
3. Portar un camp i funcions relatives al propietari.
4. Tenir un camp d'aprovació.
5. Portar un camp de transferència.
6. Determinar el propietari d'un token a partir del seu index.
7. Poder consultar les metadades del NFT.

5.2.4 Aplicacions descentralitzades

Què són?

Tothom coneix les aplicacions tradicionals que fem servir en el nostre dia a dia: WhatsApp, Twitter, Youtube, Instagram, Facebook, etc.. Aquestes són aplicacions centralitzades, això vol dir que la seva estructura està basada en el model Client-Servidor. Per exemple si jo vull consultar el meu perfil de Facebook, el meu dispositiu (client) farà una petició al servidor de Facebook sol·licitant les meves dades corresponent al meu perfil. Hi ha una interacció entre l'aplicació i els servidors de la companyia. Per altra banda, les aplicacions descentralitzades, conegudes com a dApps, tenen un funcionament semblant, però en lloc de parlar amb els servidors de l'empresa la comunicació és amb Blockchain.

Primer de tot, cal recalcar que només es poden crear dApps a partir de tecnologies que fan servir contractes intel·ligents. Per això, Bitcoin i d'altres no poden utilitzar dApps, ja que el disseny d'aquestes només permet l'intercanvi del seu actiu. En canvi, Etherum o Solana que sí que ho disposen sí que poden crear dApps amb Solidity i Rust respectivament.

Les aplicacions descentralitzades aporten molts beneficis, entre els quals és la transparència en el codi en ser aquest lliure d'accés i consulta, és a dir Open Source. Això dona més confiança que no pas els algorismes secrets de les companyies. Es passa de confiar en un tercer a un codi executat. Són lliures de censura, ja que per poder privar als usuaris del contingut haurien de desconnectar tots els nodes (o codificar-lo si es vol) que componen la xarxa Blockchain en lloc de només un en els centralitzats. Amb el qual, si un node falla o té algun problema no cau tota l'aplicació. Això vol dir una alta disponibilitat.

Les dApps a més tenen utilitats molt diverses: finances, jocs, apostes, traçabilitat, mercats, ensenyament, dominis, votacions, etc.. En aquest treball ens centrarem en el financer; les DeFi.

Estructura

Com sempre en el món Blockchain, hi ha moltes maneres d'implementar una dApp. Totes són vàlides i es regeixen per un cert grau de centralització. El Frontend, la lògica d'interfície d'usuari, no sol canviar i podríem dir que és com una peça que es pot intercanviar per un altre perfectament. En canvi, on més diferències hi ha és a la part de Backend, la lògica de l'aplicació. En una aplicació tradicional tindríem tota la part de comunicació amb una base de dades mentre que en una dApp seria amb una cadena de blocs.

Aquesta imatge ho explica molt bé:

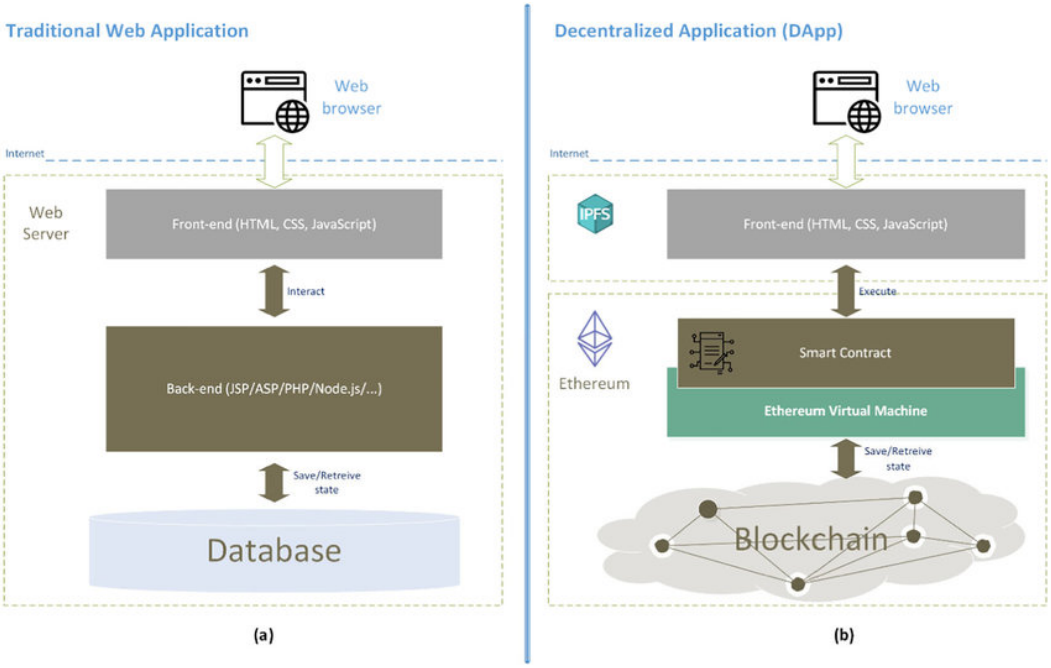


Figura 5.11: Web tradicional vs Web3 [15]

Web3.js és una llibreria de JavaScript que ens permet connectar el nostre Frontend amb la Blockchain d'Ethereum. És semblant a una API, però no intercanvia informació com per exemple el protocol AJAX, sinó que utilitza JSON-RPC. Aquesta consisteix en la crida de procediments remots (Remote Procedure Calls).

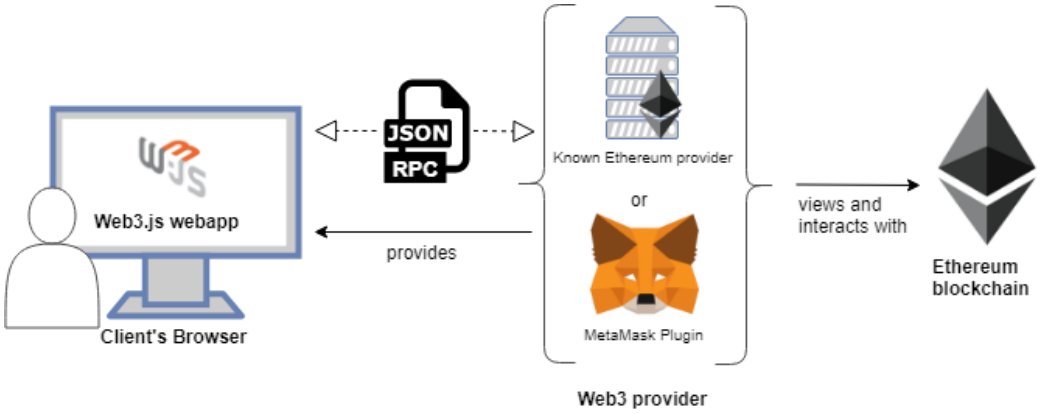


Figura 5.12: Estructura del funcionament d'una app descentralitzada

Respecte al grau de centralització, podem tenir perfectament una dApp amb l'habitual arquitectura Client-Servidor: podem tenir un servidor executat la nostra dApp amb el seu domini o es pot penjar a una Blockchain com Swarm que descentralitza tota la nostra aplicació web. El projecte se centrarà en el primer cas. Per consultar més informació al respecte, recomano molt aquest article: [21]

CAPÍTOL 6

Requisits del sistema

En aquesta secció explicaré cadascun dels requeriments que té el projecte, tan funcionals com no. També la seva estructura econòmica.

Requeriments funcionals:

- Com a propietari, vull poder donar informació sobre el document i poder donar seguretat i fiabilitat al sistema.
- Com a propietari, un cop proporcionada la informació del document, vull poder tokenitzar l'actiu i incorporar-lo a un fons de liquiditat per poder rebre finançament.
- Com a propietari, vull poder arbitrar en el procés.
- Com a propietari, vull poder rebre el finançament.
- Com a usuari, vull poder verificar l'existència d'un document i donar-li validesa.
- Com a usuari, vull poder denegar un document i advertint que no és fiable.
- Com a usuari, vull poder consultar la liquiditat del fons.
- Com a usuari, vull poder consultar la taxa d'interès.
- Com a inversor, un cop creat un fons, vull poder invertir-hi.
- Com a inversor, vull poder rebre el benefici.
- Com a inversor, vull poder consultar quants tokens de l'actiu disposo.

Requeriments no funcionals:

- El sistema crearà fons de liquiditat de dos actius: Ether i Tokens de l'actiu. És a dir, només es pot finançar amb aquests dos.

- El sistema farà el procés de tokenització generant tokens a partir de l'estàndard ERC-20.

- El sistema tindrà una taxa d'interès inicial del 15%, variable segons la fiabilitat que proporcioni l'usuari: col·lateral, informació del document i verificació.
- El sistema considerarà que un document està verificat si hi ha una relació 5:1 respecte aprovacions i denegacions.
- El sistema només acceptarà ethers en la seva unitat de valor més petita, els weis, equivalent a 10^{18} ethers.

Al ser un projecte crypto i perquè cal definir bé l'estructura econòmica d'aquest per la seva viabilitat en el temps, definiré els tokenomics:

Taula 6.1: Tokenomics del projecte

Factor	**Preguntes**	**Resposta**
Oferta	Quina és l'oferta total? I en circulació?	La total és indefinida: oferta inicial del propietari més arbitratges. L'altre és definida pel propietari en la creació.
Valor	Quina utilitat té i en què consisteix?	És un token que representa una unitat de valor d'una factura. És semblant a una acció a una empresa però dedicat a una factura.
Distribució	Qui s'encarrega i com ho fa?	El propietari mitjançant el fons de liquiditat
Demanda	Quins mecanismes incentius té?	Té una taxa d'interès variable a partir d'informació i confiança donada pel propietari i els inversors.

CAPÍTOL 7

Estudi i decisions

Aquí explicaré tots els elements tant de maquinari com de programari que he fet servir per dur a terme. Vull resaltar que perquè jo els hagi utilitzat no significa que siguin els millors; molt probablement n'hi ha més bons i que cadascú triï allò el que consideri oportú.

7.1 Hardware

Respecte al maquinari, al capítol 2 ja n'he detallat que només cal un ordinador, connexió a Internet i electricitat. Aquí mostro el meu:

Model	Asus ROG Zephyrus G14
CPU	AMD Ryzen 7 5800HS Radeon Graphics 3.20 GHz
RAM	16GB
Targeta gràfica	NVIDIA GeForce RTX 3050

Taula 7.1: Característiques de l'ordinador utilitzat

7.2 Software

Aquí entraré en detall sobre el programari que jo he utilitzat per fer el projecte. Cal dir que el que he emprat no ha de ser necessàriament el mateix i n'hi ha moltes possibilitats que són perfectament vàlides.

Solidity

És el llenguatge principal per desenvolupar contractes intel·ligents. He fet servir principalment la versió 0.8.15. Per consultar més informació a l'apartat de Solidity.

Figura 7.1: Logo de Solidity

Remix IDE

És l'entorn principal online per crear, programar, testejar i desplegar contractes intel·ligents fàcilment. L'he utilitzat molt per la seva senzillesa i al ser pràctic. Disposa de comptes de prova de fins a 100 ETH per testejar el codi. Per projectes més grans no és recomanable perquè l'IDE té un límit de processament de memòria.

Figura 7.2: Logo de Remix

OpenZeppelin

Es tracta d'un framework per construir contractes intel·ligents. És una de les principals llibreries que ja tenen implementats els estàndards de tokens com ERC-20, ERC-721 i més. Es pot consultar el repositori amb els contractes aquí [22]

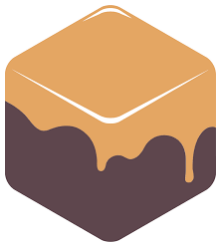
Figura 7.3: Logo d'OpenZeppelin

Truffle i Ganache

És un framework per crear, programar, compilar i testejar contractes intel·ligents fàcilment. Conjuntament amb la part servidora de Node.js, permet fer el mateix que Remix però més ràpid i pràctic.

L'únic problema que té és que s'ha d'especificar una cadena de blocs perquè es pugui connectar. Per això hi és Ganache, que permet engegar una cadena de blocs d'Ethereum localment i on podem desplegar els contractes. També podem indicar-li directament un mainnet o testnet amb l'ajuda d'un proveïdor.

(a) Logo de Truffle



(b) Logo de Ganache

Node.js

És un entorn d'execució de JavaScript per servidor que permet programar Backend. S'ha fet servir la versió 16.16.0.

Figura 7.5: Logo de Node.js

React.js

És un framework de frontend per crear interfícies d'usuari. S'ha fet servir la versió 18.0.

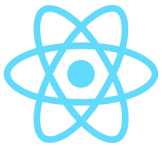


Figura 7.6: Logo de Node.js

Bootstrap

És un framework de frontend que ajuda en el disseny d'elements a la interfície d'usuari amb biblioteques. He fet servir la darrera versió, la 5.0.

Figura 7.7: Logo de Bootstrap

Github

És una aplicació web molt coneguda pel control de versions del codi font conjuntament amb Git. Tots els canvis i les versions estan guardades a un repositori de Github que es pot consultar aquí [23].

Figura 7.8: Logo de Git i Github

Visual Studio Code

És el principal entorn de treball o IDE que he utilitzat perquè permet incorporar amb facilitat programari com Truffle, Ganache, Github i altres mitjaçant extensions.

Figura 7.9: Logo de Visual Studio Code

Excel

Aplicació de Microsoft que, mitjançant fulls i taules, permet fer càlculs, gràfics, programar macros, etc.. És una eina molt útil per fer simulacions, analitzar i comparar resultats.

Figura 7.10: Logo d'Excel

CAPÍTOL 8

Anàlisi i disseny del sistema

L'objectiu clau és crear un sistema basat en Blockchain que ajudi a finançar empreses que disposen de poca liquiditat en el curt termini mitjançant el facturatge. Els usuaris donaran informació de la factura i li donaran fiabilitat amb un actiu de garantia i una verificació que indiqui l'existència del document.

Aleshores, el sistema està conformat pels següents actors:

- Propietari: és qui vol obtenir el finançament i dona informació del document. També s'encarrega de l'arbitratge.
- Inversor: és qui proporciona finançament i dona veracitat als documents.
- Contracte: és l'encarregat d'administrar, mostrar informació i interactuar amb els usuaris i la cadena de blocs.

Per tant, el disseny proposat és una aplicació descentralitzada que està conformada pel següent:

- Contracte intel·ligent que tindrà implementat la lògica de negoci.
- Aplicació web on l'usuari pot interactuar amb aquest contracte.
- Estructura de Cadena de blocs on quedaran registrades les transaccions.

Amb la part de Backend, tota la lògica dels contractes intel·ligents, es pot fer una analogia entre els contractes i classes o mòduls. Això es deu al fet que el llenguatge *Solidity* és fortament Orientat a Objectes i semblant a altres llenguatges que fan servir aquest paradigma com ja s'ha comentat aquí Solidity.

Llavors, es pot dividir el contracte intel·ligent en diferents mòduls:

- L'Automated Market Maker: és la implementació d'un fons de liquiditat de dos actius, ether i tokens.
- Tokenització: serà el contracte encarregat de la gestió dels tokens del document. Està basat en l'estàndard ERC-20 d'Ethereum [12].
- Verificador: proporciona veracitat al document.
- Facturatge: és qui gestiona i connecta tots els mòduls. És el principal amb qui l'usuari pot interactuar.
- Interès: controla el valor de la taxa d'interès.

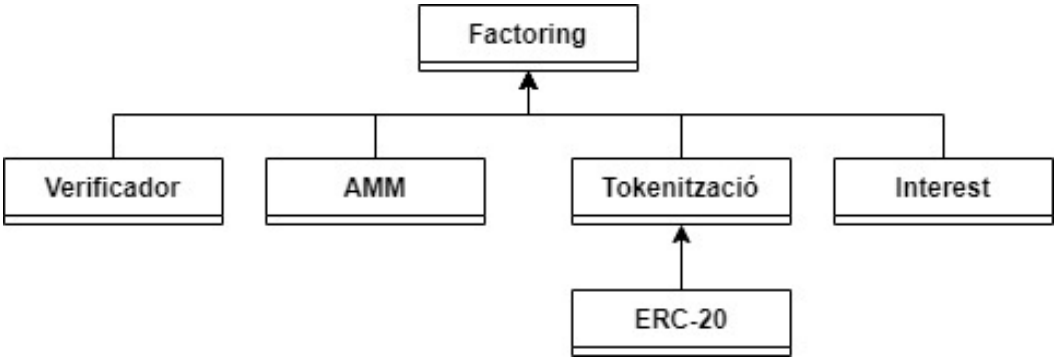


Figura 8.1: Diagrama de clases UML simplificat

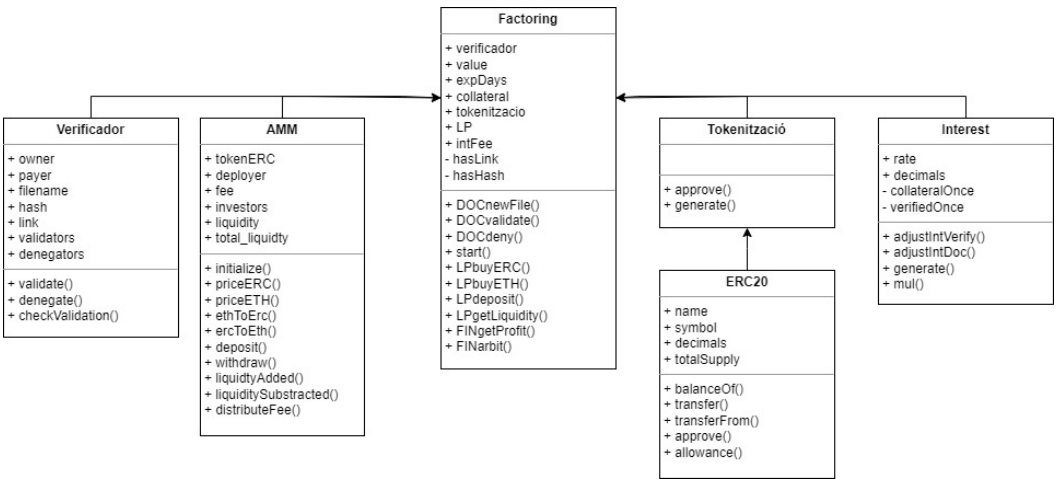


Figura 8.2: Diagrama de clases UML

Per altra banda, tenim l'aplicació web que aquesta està formada per un conjunt de formularis on l'usuari pot demanar i rebre informació i manipular l'estat del contracte. S'ha seguit el patró de disseny Model-Vista-Controlador.

Principalment, està composta per:

- Components: són elements que conformen la interfície web. La gran majoria són formularis per interactuar amb el contracte. Es tracta del Model.
- Frontend: la lògica que està al darrere de la pàgina i que parla amb el contracte intel·ligent. És el Controlador.
- Pàgina principal que aglutina els dos. És la Vista.

No s'ha indagat i aprofundit en l'estètica i el disseny d'aquest, ja que no és l'objectiu principal del projecte. En qualsevol cas, es pot fer servir el mateix Remix IDE com a interfície. Només cal compilar i desplegar el codi per poder interactuar directament amb el contracte sense fer cap UI.

Seguidament, la cadena de blocs. Aquí cal distingir dos tipus de Blockchains on es pot desplegar el contracte:

- Testnet: xarxa de proves on hi ha usuaris d'exemple amb diners ficticis per poder comprovar el bon funcionament dels contractes.
- Mainnet: xarxa principal on s'executen els contractes i on els usuaris interactuen amb els seus diners reals.

Voldria recalcar que si el contracte funciona correctament al testnet, sí o sí funcionarà també al mainnet. En el nostre cas el testnet serà Teranyina i el mainnet Alastria. Per proves locals també s'ha fet servir Ganache, un testnet al nostre ordinador [7.2].

Finalment, aquí un esquema general de l'estructura del projecte:

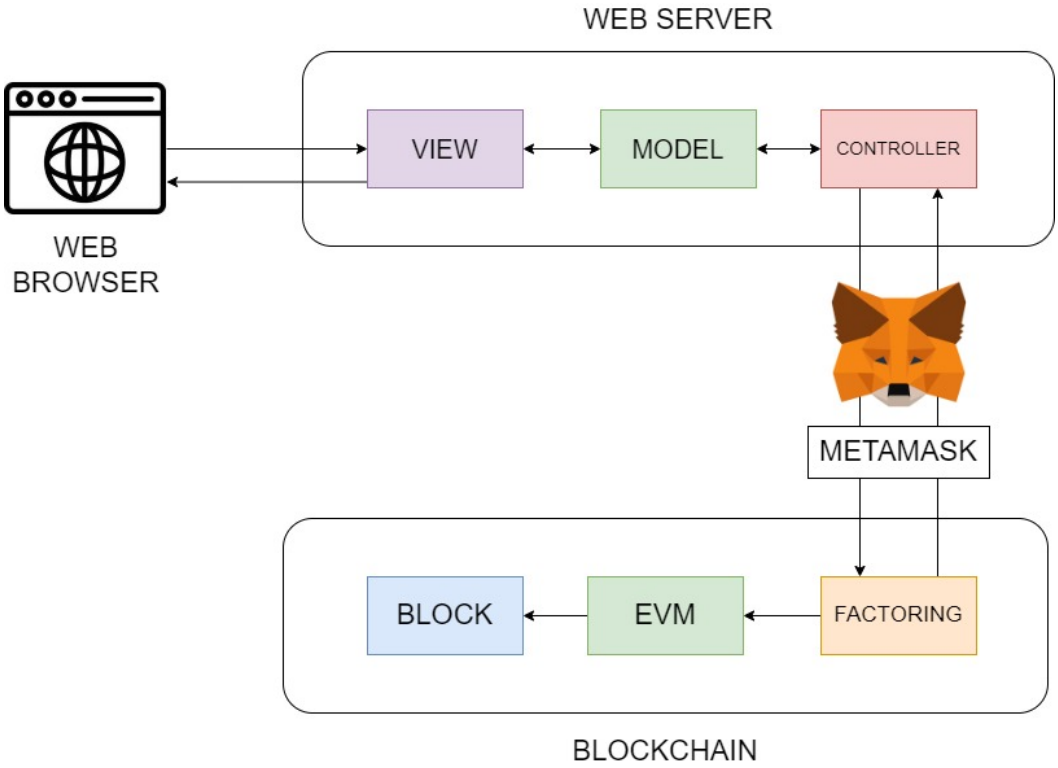


Figura 8.3: Esquema general del projecte

CAPÍTOL 9

Implementació i proves

La implementació del projecte es pot fer de dues maneres. Una és online mitjançant l'editor Remix d'Ethereum o localment a través, per exemple, de Visual Studio Code amb les extensions ja comentades anteriorment al Software. Jo he optat per les dues opcions, encara que és preferible la segona, ja que dona més llibertat a l'hora de poder implantar el projecte i a més és molt més ràpid. Remix està orientat també per persones que estan començant a desenvolupament *smart contracts*.

9.1 Implementació

La implementació i les proves s'han fet en l'entorn de desenvolupament de Visual Studio Code i testos amb Remix (es poden fer perfectament a VSC també). El testnet que s'ha fet servir és Ganache.

Primer de tot, hem de donar informació del document al sistema. Per això serà necessari el nom del document, quin és el seu valor (en ETH) i per últim la data d'expiració. Aquesta data són els dies que queden perquè el deutor faci efectiu el pagament al propietari de la factura. Perquè tot això sigui més fiable, he decidit incorporar noves variables que el propietari pot decidir o no proporcionar. Es tracten d'un hash del document, com per exemple MD5, i un enllaç a un emmagatzematge al núvol per poder consultar el document, com per exemple Google Drive o IPFS. Per fomentar aquest procés, he decidit que si es pot tenir accés a aquesta informació, el tipus d'interès baixi.

Tot seguit, el propietari ja pot començar a crear un fons de liquiditat. Per poder-lo fer, necessita indicar en quants tokens vol dividir la factura i donar un col·lateral. El contracte intel·ligent iniciarà el procés de tokenització de l'actiu, la forma en la qual ho fa és generant els tokens ERC-20 que ha indicat el propietari. Aquests tenen com a nom Tokfa (Tokenització de factures) i el símbol TFA (Token Factoring Asset). A continuació, se li dona l'aprovació a la transacció d'afegir els ethers al liquidity pool i es crea amb la quantitat proporcionada dels dos actius: l'ETH i TFA. El col·lateral seran els ETH i els tokens indicats els TFA. Després, es fa efectiva la transferència. Ja està inicialitzada la pool amb

la liquiditat que ha donat el propietari. En fer-lo el valor de la taxa d'interès s'actualitza en funció de la relació entre el col·lateral i el valor.

Mentrestant, només els inversors poden validar l'existència del document o indicar que aquest és fraudulent. No es contempla, i no té sentit, que una persona validi i denegi un document alhora. A més, només es pot fer un cop així que s'ha d'avaluar i examinar bé el document. Un cop fet, es comprova si està finalment verificat o no. Jo he considerat que amb una ràtio de 5 validacions per 1 denegació ja es pot considerar com a verídic el document. Finalment, la taxa d'interès s'actualitza.

Ara un cop creat el fons, els inversors ja poden començar a comprar tokens TFA. Aquests tokens podríem considerar-los com les accions a una empresa. L'usuari indica la quantitat que vol comprar, es donen permisos al fons i es fa la transacció. El fons rep això i fa l'intercanvi d'ETH a tokens. Aquest sap quants tokens li pertoquen a l'inversor gràcies a la fórmula $x \cdot y = k$ com ja hem comentat a Liquidity pools.

Un cop que es fa una compra, el propietari ha d'intervenir per fer l'arbitratge (5.1.3). Aquest consisteix a generar nous tokens que intercanviarà pels ethers que ha dipositat l'inversor. Com ja es va comentar a la part de la llei de l'oferta i la demanda, al ser sota demanda (la compra de l'inversor) es puja l'oferta de TFAs per equilibrar el valor entre TFA i ETH.

Cal comentar que per cada transacció que es fa de compra, es genera una fee com a benefici per la persona que ha desplegat el contracte a la Blockchain.

Aquest procés de compra i arbitratge es fa fins a la data d'expiració. Un cop arribat aquest moment, es presenten 2 escenaris possibles: que el propietari rebi el pagament per part del proveïdor o que aquest no es produeixi. En el primer cas, el propietari ha de fer un dipòsit amb el valor de la factura més els interessos. Un cop fet l'ingrés, es distribueixen els beneficis i les fees. Els inversors poden obtenir el seu benefici traient la seva quantitat corresponent del fons. Per altra banda, en cas que no es rebi el pagament, s'activa el col·lateral i cobreix una part i tot el risc de la inversió.

Finalment, quan tothom ha rebut la seva part es dona el facturatge com acabat.

Hem comentat que s'ajusta la taxa d'interès depenent de la fiabilitat, però quina és la funció implementada que ho fa? Primer de tot, hem de definir que és "fiabilitat". En aquest projecte s'ha considerat la fiabilitat amb 3 variables:

- col·lateral: l'impacte que té el valor del col·lateral és definida per la següent funció:

$$f(x,y,t,r) = \begin{cases} g(x,y,t,r) & \text{if } \frac{y}{x} \geq r \\ 100\%, & g(x,y,t,r) > 100\% \\ 3\%, & h(x,y,t,r) < 3\% \\ h(x,y,t,r) & \text{altrament} \end{cases}$$

on:

$$g(x,y,t,r) = t + (\frac{x}{y} - r) \tag{9.1}$$

$$h(x,y,t,r) = t \cdot (2 - \frac{r \cdot x}{y}) \tag{9.2}$$

on:

  - x és el col·lateral, i el seu domini és: $x > 0$ i $x \leq y$.
  - y és el valor del document, amb domini $y > 0$.
  - t és la taxa actual, i el seu domini és: $t > 0$
  - r és el punt de referència, amb domini $r > 0$ i $r \leq y$

  L'objectiu de la nostra funció és que a mesura que s'aporti més garantia o col·lateral, la taxa d'interès ha de reduir-se i al contrari, quan menys garantia s'incrementi bastant. La garantia i el preu el dona el propietari mentre que la taxa i el punt el proporciona al *smart contract*.

  Com es pot apreciar, la funció f conté dues funcions més; h i g. La primera és quan s'està per sota del punt de referència i l'altra per damunt.

  Per altra banda, el punt de referència vol dir a quin punt ens situem com a punt de sortida. Això vol dir el valor de col·lateral on el preu del document sigui el de la taxa d'interès inicial. Per exemple, si tenim com a taxa d'interès inicial un 15%, valor de factura com a 500 i punt de referència 2, això vol dir que per un col·lateral de 250 (500 dividit per 2) la taxa d'interès serà del 15%. També es pot entendre amb el seu valor invers com a

percentatge: la taxa d'interès inicial se situarà en el punt en el qual el preu del document sigui el 50% del seu valor, és a dir 250 (500 per 0,5). Generalment, ens interessa que estigui entre l'1,25 i 1,67 o (60% i el 80%), ja que quan el col·lateral té aquests valors es considera que està altament col·lateralitzat. Jo he considerat aquest valor com a constant amb valor $r = 1,25$.

També es pot observar que hi apareix un límit quan la funció h sobrepassa el 100%. Això es deu al fet que per un col·lateral molt petit i un preu molt elevat, el valor es distorsiona molt. Per altra banda, si l'actiu està sobrecol·lateralitzat, llavors la taxa d'interès se situarà com a mínim al 3%.

Respecte a la variable t, aquesta es tracta de la taxa d'interès inicial. Aquest valor el podem considerar gairebé com a constant. Com més gran sigui, més beneficis tindran els inversors i més haurà d'aportar el propietari. En aquest projecte he considerat donar una taxa d'interès inicial de 15%, és a dir, que $t = 15\%$. Per tant, la funció quedaria així:

$$f(x,y) = \begin{cases} g(x,y) & \text{if } \dfrac{y}{x} \geq r \\ 100\%, & g(x,y) > 100\% \\ 3\%, & h(x,y) < 3\% \\ h(x,y) & \text{altrament} \end{cases}$$

on:

$$g(x,y) = 15 + (\frac{x}{y} - 1,25) \tag{9.3}$$

$$h(x,y) = 15 \cdot (2 - \frac{1,25 \cdot x}{y}) \tag{9.4}$$

Si fixem com a constants amb els següents valors $y = 150$, $t = 15$ i $r = 1,25$, es pot veure el comportament de la funció en el següent gràfic:

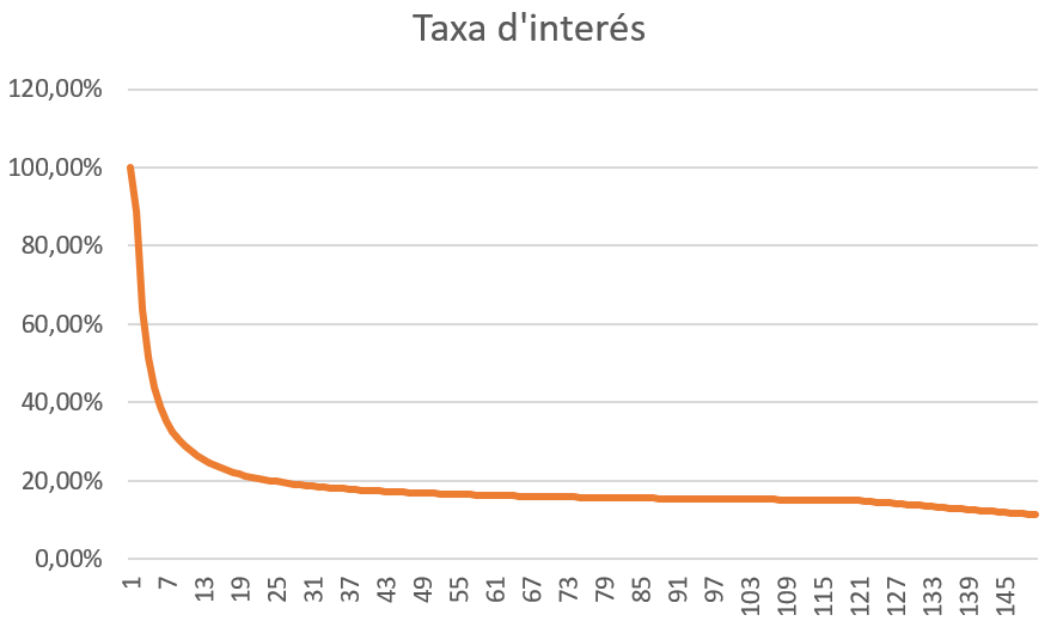


Figura 9.1: Comportament de la la funció que modifica la taxa d'interès segons el col·lateral.

- Verificació:

  Com ja s'ha comentat prèviament en aquesta secció, la ràtio per considerar que un document està verificat és de 5:1. Això vol dir que han d'haver-hi 5 vegades més verificacions que denegacions. Si es dona el cas, llavors es reduirà el valor de la taxa d'interès set vuitens ($\frac{7}{8}$). Es fa això per promoure el procés de verificació. No es fa una funció com al col·lateral, ja que els inversors poden no estar incentivats a verificar documents, ja que ells volen que la taxa sigui alta i verificant poden contribuir a reduir-la.

- Documentació:

  Consisteix a proporcionar informació respecte a la veracitat del document. Aquesta ha de ser el hash o checksum del document i un enllaç per poder veure l'arxiu. Com l'anterior cas, si es compleix. Llavors es farà un decrement de set vuitens de la taxa. El control sobre la veracitat dels enllaços i el hash es deixa a càrrec del Frontend i no es donarà èmfasi en aquest projecte.

Llavors, fent càlculs es pot reduir fins a aproximadament més d'un 40% el valor de la taxa d'interès si es compleixen tots els requisits que s'han comentat. Aquesta dada depèn del punt de referència també; com més gran sigui més es podrà reduir la taxa.

9.2 Algorismes

En aquesta secció explicaré amb detall els algorismes que considero més rellevants i fonamentals per entendre el funcionament general del projecte. Comentaré només els que estan presents a contractes intel·ligents, les de frontend s'obviaran. Els algorismes estan implementats en llenguatge *5.2.2*.

Inicialització del fons

Figura 9.2: Funció start, present a Factoring.sol

```
event StartLP(uint256 assetX, uint256 assetY);
function start(uint256 quants_tokens) public payable {
    collateral = msg.value;

    //1.
    tokenitzacio = new Tokfa(quants_tokens * 4);
    uint256 token_amount = quants_tokens * 10**uint(tokenitzacio.decimals());
    //2.
    tokenitzacio.approve(tx.origin, address(LP), tokenitzacio.totalSupply());
    //3.
    LP.initialize{value:msg.value}(address(tokenitzacio), token_amount);
    //4.
    intFee.adjustIntCol(value, collateral);

    emit StartLP(token_amount, msg.value);
}
```

Aquest algorisme es podria considerar el botó d'encendre del fons de liquiditat. Es pot dividir en 4 passos:

1. Tokenització de l'actiu:

   Genera els tokens que ha indicat l'usuari. Tot seguit es guarden els tokens amb els decimals.

2. Permisos:
   Se li dona la possibilitat al fons de liquiditat que pugui fer transferències en nom del propietari amb la quantitat de tokens que ha indicat.

3. S'inicialitza el fons:

   Amb el col·lateral i els tokens indicats per l'usuari, es canvia l'estat del fons per un amb els dos actius amb valor donant així una liquiditat inicial.

El fons farà la transferència de tokens al contracte gràcies als permisos d'abans.

4. Taxa d'interès:

   S'ajusta la taxa d'interès en funció del col·lateral. Per veure com ho ajusta veure aquesta secció prèvia 9.1.

Intercanvi d'actius

Figura 9.3: Funcions per intercanviar ETH per TFA

```
event BuyERC(uint256 payed, uint256 tokens_received)
function LPbuyERC() public payable {
    LP.ethToErc{value:msg.value}();

    emit BuyERC(msg.value, LP.tokens_received());
}
```

```
function ethToErc() external payable {
        uint256 tokens_swapped = calculatePrice(msg.value, (ad

        require(tokenERC.transfer(tx.origin, tokens_swapped));

        fee += msg.value;
        investors[tx.origin] += msg.value;

        tokens_received = tokens_swapped;

        emit TokensSwapped(tokens_swapped);

}
```

Aquestes dues funcions consisteixen en el procés de comprar tokens TFA amb ETH. Ja que la funció és de tipus *payable*, guarda la quantitat d'ethers al camp *msg.value*. Seguidament, crida la funció del contracte LP, que és el fons de liquiditat, passant l'ethers. A la funció del fons, es calculen els tokens que rebria

l'inversor a través de la fórmula de $x \cdot y = k$. No es veu a la imatge, però fa el càlcul a partir del valor en eth de l'inversor i de les dues quantitats que hi ha dels dos actius al fons. A continuació, el fons fa la transferència de tokens a l'inversor amb la quantitat calculada i guarda la quantitat d'eth d'aquest. Al ser una transacció, es guarden les fees. Un cop acabat, s'emet un event informant de l'intercanvi.

Figura 9.4: Funcions per intercanviar TFA per ETH

```
event BuyETH(uint256 payed, uint256 tokens_received);
function LPbuyETH(uint256 tokens) public {

    tokenitzacio.approve(tx.origin, address(LP), tokens);
    LP.ercToEth(tokens);
    emit BuyETH(tokens, LP.eth_received());
}
```

```
function ercToEth(uint256 token_amount) external {

        uint256 eth_swapped = calculatePrice(token_amount, tokenERC.balanceOf(a

        (bool success, ) = tx.origin.call{value: eth_swapped}("");
        require(success, "Transfer failed.");

        require(tokenERC.transferFrom(tx.origin, address(this), token_amount));

        fee += eth_swapped;
        emit ETHSwapped(eth_swapped);
        eth_received = eth_swapped;
}
```

Les dues funcions de les imatges són les que implementen l'intercanvi entre tokens TFA i ETH. El procés és similar a l'altre intercanvi comentat, però hi ha diferències. La primera és que cal donar permisos al contracte que implementa el fons de liquiditat perquè aquest pugui transferir en nom de l'usuari els tokens al pool de TFAs. Es fa la transferència d'ETH a l'usuari i es guarden les fees.

Afegir liquiditat

Figura 9.5: Funcions per afegir liquiditat al fons

```
event DepositLP(uint256 liquidity);
function LPdeposit() public payable {
    uint256 tokens_added = tokenAmount(msg.value);
    if (tokens_added > TFAmybalance()){
        tokenitzacio.generate(tokens_added);
    }

    tokenitzacio.approve(tx.origin, address(LP), tokenAmount(msg.value));
    LP.deposit{value:msg.value}();
    emit DepositLP(LP.liq_added());
}
```

```
function deposit() external payable {
        require(msg.value > 0);
        uint256 token_amount = ((msg.value * tokenERC.balanceOf(address(this))
        uint256 liquidity_added = (msg.value * total_liquidity) / (address(thi

        liquidity[tx.origin] += liquidity_added;

        total_liquidity += liquidity_added;
        require(tokenERC.transferFrom(tx.origin, address(this), token_amount))

        emit Deposit(liquidity_added);
        liq_added = liquidity_added;
}
```

Per tal de poder afegir liquiditat, l'usuari ha d'indicar la quantitat d'ethers que vol en què s'incrementi. Assumim que la liquiditat total és l'ether. Es pot veure que hi ha una estructura de control sobre els tokens, això es deu al fet que pot passar que la persona no disposi de saldo. En el nostre cas, hem considerat que qualsevol pot afegir liquiditat si disposa ethers, llavors si no té prou tokens se'n generaran de nous. Ara, a partir de la relació de valor entre les dues monedes, és a dir, els tokens i ethers ($\frac{x}{y}$) es calcula el valor tant d'ETH com dels tokens que s'ingressaran. Com abans, es necessita l'aprovació abans amb la quantitat a afegir. Seguidament, es fan les transferències dels dos actius.

Retirar liquiditat

Figura 9.6: Funcions per treure liquiditat del fons

```
event WithdrawLP(uint256 eth_received, uint256 tokens_received);
function LPgetLiquidity(uint256 liquid) public {
    LP.withdraw(liquid);

    emit WithdrawLP(LP.eth_received(), LP.tokens_received());
}
```

```
function withdraw(uint256 value) external {
        uint256 eth_value = (value * address(this).balance) / total_liquidity;
        uint256 token_value = (value * tokenERC.balanceOf(address(this))) / to

        total_liquidity -= eth_value;

        (bool success, ) = tx.origin.call{value: eth_value}("");
        require(success, "Transfer failed.");

        require(tokenERC.transfer(tx.origin, token_value));

        emit Withdraw(eth_value, token_value);
        eth_received = eth_value;
        tokens_received = token_value;
}
```

Per treure liquiditat, l'usuari indica quanta vol treure de la pool. Assumim que la liquiditat total és l'ether. Llavors, a partir de la relació de valor entre les dues monedes, és a dir, els tokens i ethers ($\frac{x}{y}$) es calcula el valor tant d'ETH com dels tokens. Com abans, es necessita l'aprovació abans amb la quantitat que es traurà. Seguidament, es fan les transferències dels dos actius.

Obtenció de beneficis

Figura 9.7: Funcions obtenir els beneficis

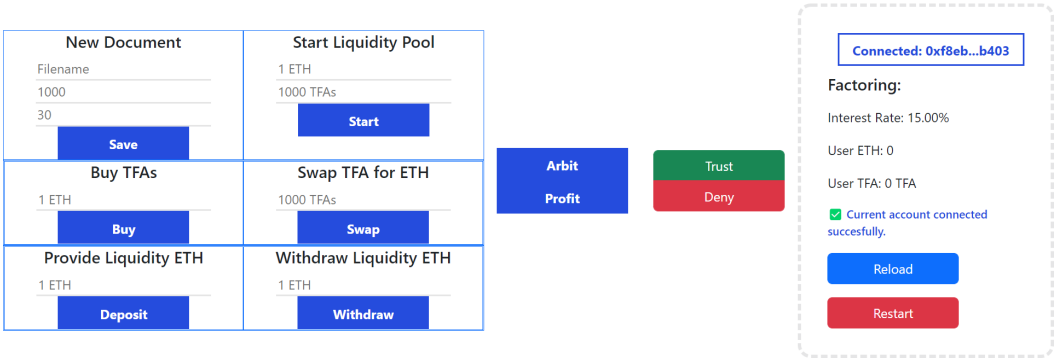
```
function FINgetProfit() public {
    LPgetLiquidity(intFee.mul(LP.invested(tx.origin)));
}
```

```
function invested(address who) external view returns (uint256){
        return investors[who];
}
```

Aquesta funció permet als inversors obtenir el benefici directament un cop l'usuari ha fet el dipòsit. Primer preguntem al fons quant a invertit l'usuari. Un cop que es té, es multiplica per la taxa d'interès actual. Finalment, es crida la funció de treure liquiditat amb el valor més els interessos.

Interfície gràfica

La interfície gràfica s'ha fet amb l'entorn de treball Visual Studio Code amb el framework de React. Es tracta d'un Single Page Application molt bàsic per interactuar amb el contracte, Podríem dir que és un fork o el seu punt de partida és aquesta guia feta per part de Web3 Academy [24]. La interfície es veu així:



Aquesta està dividida en tres seccions. La primera, situada a l'esquerra de tot, conté una sèrie de formularis per crear el fons, comprar tokens, donar informació del document, etc.. La segona part, al mig, està formada per un conjunt de botons. Aquests són les transaccions simples que no requereixen informació per part de l'usuari. Són l'arbitratge, l'obtenció dels beneficis i la verificació i denegació del document. La tercera, a la dreta de tot, podríem dir que és la consola o la sortida on veiem els resultats i les dades que ens proporciona el contracte. També té un botó per recarregar la informació del fons i un altre per reiniciar tot.

Comunicació entre el frontend i el contracte

A continuació comentaré com es comunica el frontend amb el contracte intel·ligent. Abans de tot explicaré qué és l'ABI, de les sigles *Application Binary Interface*. Es podria considerar com l'esquelet del contracte, que defineix tots els seus mètodes i esdeveniments presents. És semblant a les API. És necessari per saber com comunicar-se amb el contracte intel·ligent i cridar al mètode correctament. Està en format JSON.

Quan es comunica amb el contracte, hem de diferenciar dues transaccions: les de lectura i escriptura. La primera només és per consultar l'estat actual del contracte i no es consumeix gas. Per fer aquesta operació, s'ha de fer servir la crida *.call()*. Per altra banda, la segona modifica l'estat del contracte i, per tant, s'ha de pagar el gas. Aquestes necessiten uns paràmetres i llavors, mitjançant una sol·licitud a Metamask, es fa la transacció. Com per exemple:

```
const transactionParameters = {
    to: contractAddress,
    from: address,
    data: factoringContract.methods.start(numberTokens).encodeABI(),
    value: ether,
};
```

Figura 9.8: Definició dels paràmetres d'una transacció

Es pot veure que es necessita saber l'adreça del contracte, la de l'usuari, la funció encodificada en ABI i, si la funció és payable, una quantitat d'ethers. Un cop això, podem fer la crida de la transacció gràcies a l'API que proporciona MetaMask.

```
const txHash = await window.ethereum.request({
    method: "eth_sendTransaction",
    params: [transactionParameters],
});
```

Figura 9.9: Petició de transacció, o *call* amb els paràmetres definits

A més, a l'hora d'implementar les transaccions que modifiquen l'estat diferenciarem entre les que reben una quantitat d'ethers o *payables*, i les que no ho fan. En lloc de codificar una funció una a una, he fet una abstracció d'aquestes dues. L'única diferència seria que una té el camp *value* amb els ethers i l'altre no. Un exemple d'això:

```
export const txBuyTFA = async (ether) => {
    return transactionPayable(ether, factoringContract.methods.LPbuyERC().encodeABI());
};

export const swapETH = async (tokens) => {
    const tokensTFA = web3.utils.toWei(tokens, 'ether');
    return transaction(factoringContract.methods.LPbuyETH(tokensTFA).encodeABI());
};
```

Figura 9.10: Exemple del tipus de transaccions

9.3 Proves

En aquesta secció mostraré els diferents testos unitaris i complets del sistema que he realitzat. Per això faré servir un plugin que té l'IDE de Remix.

9.3.1 Testos unitaris

Inicialització del fons

Primer comprovarem el funcionament de *start* creant diverses pools de liquiditat i que aquestes tinguin la quantitat que els hi pertoca. També s'han afegit proves per a casos com que hi hagi 0 de quantitat dels dos assets al fons o nombres petits. En tot cas, aquí estan els casos i el resultat esperat:

1. Quantitats abundats de dos actius: el resultat ha de ser de 12 ETH i 10^9 tokens.
2. Quantitats mitjanes de dos actius: el resultat ha de ser de 0,28 ETH i 13250 tokens.
3. Zero quantitat d'ETH aportat: el resultat ha de ser error.
4. Zero quantitat de TFAs aportat: el resultat ha de ser error.
5. No s'especifica valor: el resultat ha de ser error.
6. Quantitat petita dels dos actius: el resultat ha de 1000 wei i 4 tokens.
7. Quantitat molt petita dels dos actius: el resultat ha de 10 wei i 1 tokens.

En resum, la programació d'aquests testos es pot resumir amb les següents imatges:

```
/// #sender: account-1
/// #value: 12000000000000000000
function checkStartFactoring() public payable {
    uint256 valor = msg.value;
    uint256 ntokens = 1000000000;
    factoring.start{value: msg.value}(ntokens);

    Assert.equal(factoring.LPether(), valor, "error value");
    Assert.equal(factoring.LPtokens(), ntokens*10**18, "error value");
}
```

Figura 9.11: Test bàsic de la funció *start*

```
/// #sender: account-3
/// #value: 0
function checkStartFactoring3() public payable {
    uint256 valor = msg.value;
    uint256 ntokens = 10;
    try factoring3.start{value: msg.value}(ntokens) {
        Assert.ok(false, "Must be error");
    }
    catch Error(string memory reason) {
        Assert.ok(true, "Must be true");
    }
    catch (bytes memory reason) {
        Assert.ok(true, "Must be true");
    }
}
```

Figura 9.12: Test d'errors de la funció *start*

Primer de tot, els triples comentaris indiquen amb quin compte d'usuari es fa servir i quina quantitat. És a dir, sender és el compte que signa la transacció i value el valor en ethers que diposita. Remix ofereix 10 comptes (des de *account-1* fins *account-10*) amb 100 ETH cadascuna al seu wallet.

Es pot veure que els testos són simples; indiquem els valors a la funció *start* tant d'ETH com de tokens i comprovem que el fons tingui correctament els valors. Per altra banda, la segona imatge té un control d'errors. Si es produeix un error en fer la transacció, llavors considerem el resultat com a correcte, ja que hauria de produir-se un error.

Un cop tot definit, executem els testos i comprovem el resultat:

```
Result for tests/Factoring1_test.sol
Passed: 7
Failed: 0
Time Taken: 3.26s
```

Figura 9.13: Resultat dels testos unitaris de la funció *start*

Intercanvi d'actius

El següent testeig serà a la funció d'intercanvi d'ethers per tokens, *LPbuyERC*, i la seva contrària, *LPbuyETH*. Suposem que tenim una pool formada per aquesta quantitat de cada actiu: 12 ETH i 10^9 TFA. Llavors, els casos a tractar seran els següents:

1. Primer intercanvi d'ETH per TFAs: el resultat ha de ser un increment de 0 a una quantitat de tokens superior.
2. Arbitratge; intercanvi de TFAs per ETH per part del propietari: el resultat ha de ser el mateix o semblant al crear la pool (12 ETH i 10^9 TFA).
3. Més intercanvis d'ETH per TFAs: el resultat ha de ser un increment de la quantitat prèvia a una quantitat de tokens superior.
4. Segon arbitratge: el resultat ha de ser 12 ETH i 10^9 TFA.
5. Intercanvi amb altre compte: el resultat ha de ser un increment de 0 a una quantitat de tokens superior.
6. Zero com a valor d'intercanvi: error com a resultat.
7. Valor d'intercanvi no donat: error com a resultat.

El valor dels ethers aportats pels usuaris s'ha obviat en aquest cas, només es veuen els increments o decrements de quantitats en els actius dels comptes. La implementació ha estat la següent:

```
/// #sender: account-2
/// #value: 1490000000000000
function checkETH() public payable {
    Assert.ok(factoring.TFAmybalance()==0, "Should have 0 TFAs");
    factoring.LPbuyERC{value: msg.value}();
    Assert.ok(factoring.TFAmybalance()>0, "Should have more than 0 TFAs");
    Assert.ok(ethers < factoring.LPether(), "Should have more ethers");
    Assert.ok(tokens > factoring.LPtokens(), "Should have more tokens");
}
```

Figura 9.14: Test de la funció *LPbuyERC*

```
/// #sender: account-1
function checkArbit() public {
    factoring.FINarbit(tokens-factoring.LPtokens());
    factoring.LPbuyETH(tokens-factoring.LPtokens());
    Assert.ok(tokens/10==factoring.LPtokens()/10, "Sl
    Assert.ok(ethers/10==factoring.LPether()/10, "Sh
}
```

Figura 9.15: Test de la funció *LPbuyETH*

Primer de tot, es pot observar que en la primera funció comprovem que el saldo inicial del compte sigui 0 (ja que és el primer intercanvi). Un cop feta la transacció, es compara l'estat actual amb el previ de la transacció. Es comprova que el compte ha tingut un ingrés de tokens, l'increment d'eth i el decrement de tokens. Per altra banda, la segona funció fa el procés d'arbitratge. Genera els tokens i compra els ETH amb la quantitat necessària perquè quedi compensat el fons. Es comprova que l'estat sigui el mateix que al principi de tot. Finalment, es divideix per 10 per obviar els petits errors.

Vet aquí el resultat del test:

```
Result for tests/Factoring1_test.sol
Passed: 7
Failed: 0
Time Taken: 3.26s
```

Figura 9.16: Resultat dels testos unitaris de intercanvi d'actius

Dipositar i retirar liquiditat

A continuació, farem el testeig sobre dipositar i treure liquiditat del fons, és a dir, les funcions *LPdeposit* i *LPgetLiquidity*. L'escenari plantejat per fer els testos és el d'una pool formada pels següents per 4,7 ETH i 33.000.000 TFA. Seguidament, els escenaris plantejats són aquests:

1. Dipòsit de liquiditat: ha de veure's incrementat la liquiditat i la quantitat d'actiu ETH com de TFA.

2. Obtenció de liquiditat: pas contrari a l'anterior. El resultat ha de ser l'estat previ en fer el dipòsit.
3. Dipòsit amb altre compte: mateix que el primer però amb un altre compte.
4. Retirar liquiditar amb un altre compte: mateix cas que l'anterior.
5. Intercanvi amb altre compte: el resultat ha de ser un increment de 0 a una quantitat de tokens superior.
6. Zero com a valor de dipòsit: error com a resultat.
7. No dipòsit: error com a resultat.

```
/// #sender: account-1
/// #value: 580000000000000000
function checkDeposit() public payable {
    factoring.LPdeposit{value: msg.value}();
    Assert.ok(ethers < factoring.LPether(), "Should have more ethers");
    Assert.ok(tokens < factoring.LPtokens(), "Should have more tokens");
}

/// #sender: account-1
function checkWithdraw() public payable {
    uint256 myBalance = msg.sender.balance;
    factoring.LPgetLiquidity(580000000000000000);
    Assert.ok(ethers==factoring.LPether(), "Should have same ethers");
    Assert.ok(tokens==factoring.LPtokens(), "Should have same tokens");
    Assert.ok(msg.sender.balance>myBalance, "Should have more ethers");
}
```

Figura 9.17: Test de les funcions per treure i retirar liquiditat

Com es pot apreciar, aquestes funcions són semblants a les d'intercanvi, però hi ha un canvi en la comprovació, ja que en fer-ne un ingrés la quantitat de tokens i eth puja. A més, quan es retira liquiditat es comprova que el compte ha incrementat el seu balanç d'ethers. S'ha decidit que no cal que l'usuari tingui tokens a la seva disposició per fer l'ingrés, es generaran de nous.

Taxa d'interès

A continuació es mostraran diverses proves fetes per comprovar el bon funcionament de la implementació de la funció.

$$f(100, 86, 15\%, 1.25) = 13.875\%$$
$$f(1.2 * 10^{20}, 2.4 * 10^{19}, 15\%, 1, 25) = 15, 75\%$$
$$f(2.7 * 10^{18}, 10^{17}, 12\%, 1, 25) = 37, 75\%$$
$$f(10^{18}, 4 * 10^{17}, 35\%, 1, 25) = 20, 1\%$$
$$f(10^{17}, 10^{17}, 6\%, 1, 25) = 4, 5\%$$
$$f(10^{17}, 10^{15}, 6\%, 1, 25) = 3\%$$
$$f(10^{17}, 10^{18}, 6\%, 1, 25) = 100\%$$

Llavors, simplement és comprovar que els valors es corresponen amb els que ha de retornar la funció. Els casos que s'han estudiat són:

1. col·lateral superior a 1,25
2. col·lateral inferior a 1,25
3. col·lateral baix comparat amb el valor
4. col·lateral, verificació i documentació activades
5. col·lateral igual que valor
6. Superi el límit per abaix
7. Superi el límit per amunt

Si es fa el test, el resultat és positiu:

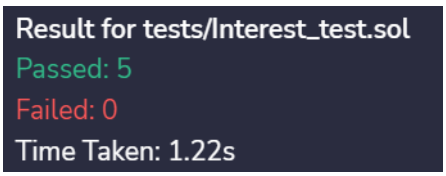


Figura 9.18: Resultat dels testos unitaris de la taxa d'interès

Verificar i denegar

Per concloure el testeig unitari, es comprovarà la verificació i denegació. Recordar que es basa en la ràtio 5:1, si és igual o superior donarà el document com a verificat. Si no hi ha cap denegació, es tindrà en compte que hi hagi més de 5 verificacions. El cas d'estudi és simple, arribar a superar la xifra de 5 verificacions i al denegar, veure si ha denegat el document.

El codi és el següent:

```
/// #sender: account-6
function verifyAcc6() public {
    verifier.validate();
    Assert.equal(verifier.checkValidation(), true, "Should be true");
}

/// #sender: account-7
function denyAcc7() public {
    verifier.denegate();
    Assert.equal(verifier.checkValidation(), false, "Should be false");
}
```

Figura 9.19: Test de les funcions per verificar

Si fem el test veiem que el resultat és també positiu:

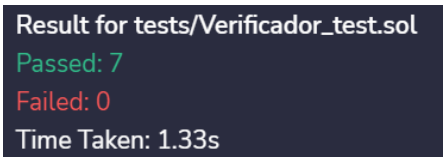


Figura 9.20: Resultat dels testos unitaris del procés de verificació

9.3.2 Test del sistema

A continuació faré tres testos del sistema. Aquests són simulacions del que ha de ser el funcionament del sistema. El primer és sense que la taxa d'interès canviï, el segon sí canvia i el tercer és amb canvi i amb algun usuari fent més 1 un intercanvi

El primer consisteix en una creació d'un fons normal per part del propietari i tot seguit una seqüència d'intercanvis i arbitratges. Per facilitar i entendre bé el procés he creat un full en Excel on es mostren els passos que es fan i els resultats que s'espera obtenir. El context amb el qual es mou és d'una taxa d'interès inicial de 15% i un valor del document de 0,1 ETH.

ACTORS		INPUTS		ESTAT INICIAL				ESTAT SEGÜENT			
Usuari	Acció	ETH	TFA	ETH User	TFA User	Pool X	Pool Y	ETH User	TFA User	Pool X	Pool Y
Account-1	Inicialitzar	0,08000	1000,00	100,00	1000,00	0,00	0,00	99,920	0,00	0,08000	1000,00
Account-2	Buy ERC	0,02580	0,00	100,00	0,00	0,08	1000,00	99,974	243,86	0,10580	756,14
Account-1	Arbit	0,00000	243,86	99,92	0,00	0,11	756,14	99,946	0,00	0,08000	1000,00
Account-3	Buy ERC	0,03400	0,00	100,00	0,00	0,08	1000,00	99,966	298,25	0,11400	701,75
Account-1	Arbit	0,00000	298,25	99,95	0,00	0,11	701,75	99,980	0,00	0,08000	1000,00
Account-4	Buy ERC	0,01370	0,00	100,00	0,00	0,08	1000,00	99,986	146,21	0,09370	853,79
Account-1	Arbit	0,00000	146,21	99,98	0,00	0,09	853,79	99,994	0,00	0,08000	1000,00
Account-5	Buy ERC	0,02000	0,00	100,00	0,00	0,08	1000,00	99,980	200,00	0,10000	800,00
Account-1	Arbit	0,00000	200,00	99,99	0,00	0,10	800,00	100,014	0,00	0,08000	1000,00
Account-6	Buy ERC	0,01270	0,00	100,00	0,00	0,08	1000,00	99,987	137,00	0,09270	863,00
Account-1	Arbit	0,00000	137,00	100,01	0,00	0,09	863,00	100,026	0,00	0,08000	1000,00
Account-7	Buy ERC	0,05380	0,00	100,00	0,00	0,08	1000,00	99,946	402,09	0,13380	597,91
Account-1	Arbit	0,00000	402,09	100,03	0,00	0,13	597,91	100,080	0,00	0,08000	1000,00
Account-8	Buy ERC	0,03110	0,00	100,00	0,00	0,08	1000,00	99,969	279,93	0,11110	720,07
Account-1	Arbit	0,00000	279,93	100,08	0,00	0,11	720,07	100,111	0,00	0,08000	1000,00
Account-1	Deposit	0,21977	0,00	100,33	0,00	0,08	1000,00	100,111	0,00	0,29977	3747,06
Account-2	Profit	0,02967	0,00	99,97	243,86	0,30	3747,06	100,004	243,86	0,27010	3376,19
Account-3	Profit	0,03910	0,00	99,97	298,25	0,27	3376,19	100,005	298,25	0,23100	2887,44
Account-4	Profit	0,01576	0,00	99,99	146,21	0,23	2887,44	100,002	146,21	0,21524	2690,50
Account-5	Profit	0,02300	0,00	99,98	200,00	0,22	2690,50	100,003	200,00	0,19224	2403,00
Account-6	Profit	0,01461	0,00	99,99	137,00	0,19	2403,00	100,002	137,00	0,17764	2220,44
Account-7	Profit	0,06187	0,00	99,95	402,09	0,18	2220,44	100,008	402,09	0,11577	1447,06
Account-8	Profit	0,03577	0,00	99,97	279,93	0,12	1447,06	100,005	279,93	0,08000	1000,00

Figura 9.21: Cas 1, simulació d'un procés sencer

El test codificat és una combinació de tots els unitaris mencionats anteriorment i es pot observar que l'estat inicial i el final de la pool ha de ser el mateix perquè el procés estigui correcte. El resultat ha de ser el mateix que es mostra aquí. S'ha afegit una comprovació final sobre els imports que han de tenir al final del test. Com que es pot arrossegar un petit valor d'error, es té en compte l'interval en el qual es troba:

```
function checkState() external {
    Assert.ok(factoring.LPtokens()>99900000000000000000, "Should have more TFAs");
    Assert.ok(factoring.LPtokens()<100100000000000000000, "Should have less TFAs");
    Assert.ok(factoring.LPether()>7990000000000000, "Should have more ETHs");
    Assert.ok(factoring.LPether()<8010000000000000, "Should have less ETHs");
}
```

Figura 9.22: Comprovació de l'estat final

En sumar totes les accions més l'últim, ens donen 24 comprovacions en total. Si executem el test podem veure que el resultat és positiu:

```
Result for tests/Factoring_test.sol
Passed: 24
Failed: 0
Time Taken: 8.63s
```

Figura 9.23: Resultat del primer test del sistema

Seguidament, el segon test és semblant a l'anterior, però es canvia la taxa d'interès durant el procés i les dades són diferents. Aquesta és la taula amb les accions fetes i el resultat esperat:

ACTORS		INPUTS		ESTAT INICIAL				ESTAT SEGÜENT			
Usuari	Acció	ETH	TFA	ETH User	TFA User	Pool X	Pool Y	ETH User	TFA User	Pool X	Pool Y
Account-1	Inicialitzar	9,12800	1350,00	100,00	1000,00	0,00	0,00	90,872	0,00	9,12800	1350,00
Account-2	Buy ERC	3,22000	0,00	100,00	0,00	9,13	1350,00	96,780	352,04	12,34800	997,96
Account-1	Arbit	0,00000	352,04	90,87	0,00	12,35	997,96	103,140	0,00	9,12800	1350,00
Account-3	Buy ERC	1,99400	0,00	100,00	0,00	9,13	1350,00	98,006	242,03	11,12200	1107,97
Account-1	Arbit	0,00000	242,03	103,14	0,00	11,12	1107,97	105,134	0,00	9,12800	1350,00
Account-4	Buy ERC	2,18400	0,00	100,00	0,00	9,13	1350,00	97,816	260,64	11,31200	1089,36
Account-1	Arbit	0,00000	260,64	105,13	0,00	11,31	1089,36	107,318	0,00	9,12800	1350,00
Account-5	Buy ERC	3,01500	0,00	100,00	0,00	9,13	1350,00	96,985	335,19	12,14300	1014,81
Account-1	Arbit	0,00000	335,19	107,32	0,00	12,14	1014,81	110,333	0,00	9,12800	1350,00
Account-1	Documentation	0,00000	0,00	110,33	0,00	9,13	1350,00	110,333	0,00	9,12800	1350,000
Account-1	Deposit	11,58700	0,00	121,92	0,00	9,13	1350,00	110,333	0,00	20,71500	3063,68
Account-2	Profit	3,58303	0,00	96,78	352,04	20,72	3063,68	100,363	352,04	17,13197	2533,76
Account-3	Profit	2,21881	0,00	98,01	242,03	17,13	2533,76	100,225	242,03	14,91315	2205,60
Account-4	Profit	2,43023	0,00	97,82	260,64	14,91	2205,60	100,246	260,64	12,48292	1846,18
Account-5	Profit	3,35492	0,00	98,01	242,03	12,48	1846,18	100,080	0,00	9,12800	1350,00

Figura 9.24: Cas 2, amb la taxa d'interès canviada

I a continuació el resultat del test:

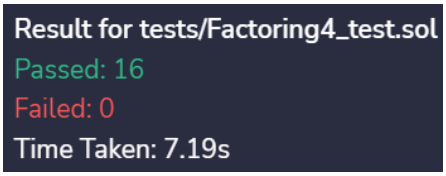


Figura 9.25: Resultat del segon test del sistema

Per acabar, l'últim test és semblant a l'anterior, però es canviïn els usuaris. En lloc de fer servir quatre comptes, es fan servir dues que faran les accions. Vet aquí la taula i el resultat:

ACTORS		INPUTS		ESTAT INICIAL				ESTAT SEGÜENT			
Usuari	Acció	ETH	TFA	ETH User	TFA User	Pool X	Pool Y	ETH User	TFA User	Pool X	Pool Y
Account-1	Inicialitzar	9,12800	1350,00	100,00	1000,00	0,00	0,00	90,872	0,00	9,12800	1350,00
Account-2	Buy ERC	3,22000	0,00	100,00	0,00	9,13	1350,00	96,780	352,04	12,34800	997,96
Account-1	Arbit	0,00000	352,04	90,87	0,00	12,35	997,96	103,140	0,00	9,12800	1350,00
Account-3	Buy ERC	1,99400	0,00	100,00	0,00	9,13	1350,00	98,006	242,03	11,12200	1107,97
Account-1	Arbit	0,00000	242,03	103,14	0,00	11,12	1107,97	105,134	0,00	9,12800	1350,00
Account-2	Buy ERC	2,18400	0,00	96,78	352,04	9,13	1350,00	94,596	260,64	11,31200	1089,36
Account-1	Arbit	0,00000	260,64	105,13	0,00	11,31	1089,36	107,318	0,00	9,12800	1350,00
Account-3	Buy ERC	3,01500	0,00	98,01	260,64	9,13	1350,00	94,991	335,19	12,14300	1014,81
Account-1	Arbit	0,00000	335,19	107,32	0,00	12,14	1014,81	110,333	0,00	9,12800	1350,00
Account-1	Documentation	0,00000	0,00	110,33	0,00	9,13	1350,00	110,333	0,00	9,12800	1350,000
Account-1	Deposit	11,58700	0,00	121,92	0,00	9,13	1350,00	110,333	0,00	20,71500	3063,68
Account-2	Profit	6,01327	0,00	94,60	260,64	20,72	3063,68	100,609	260,64	14,70173	2174,34
Account-3	Profit	5,57373	0,00	94,99	335,19	14,70	2174,34	100,565	335,19	9,12800	1350,00

Figura 9.26: Cas 3, amb la taxa d'interès canviada i només 2 comptes

Figura 9.27: Resultat del tercer test del sistema

Finalment, cal dir que els testos es poden fer també amb la interfície gràfica. S'ha optat per només fer-lo d'aquesta manera perquè el procés de testeig és més ràpid, senzill i dona el mateix resultat. Igualment, s'ha especificat com s'ha implementat el mode en què es comunica amb el contracte intel·ligent.

CAPÍTOL 10

Implantació i resultats

De cara a la implantació del projecte, hi ha moltes possibilitats. Una de les opcions com s'ha comentat és una solució depenent del grau de descentralització que desitgem. Una d'aquestes és la clàssica arquitectura Client-Servidor. Posem en marxa un servidor web amb l'aplicació amb el seu nom de domini (DNS). Tot seguit, està la possibilitat de fer-lo completament descentralitzat amb un node de Swarm i el seu domini (ENS, a la blockchain d'Ethereum) [21]. Un altre aspecte a tenir en compte és a quina cadena de blocs despleguem el contracte. Al projecte s'ha comentat el testnet de Teranyina i Alastria. Tristament, degut a temes logístics i tècnics amb el node de Alastria, no s'ha pogut fer el desplegament en aquest. Finalment, he decidit implantar-lo en Teranyina.

Per fer aquesta implantació simplement hem de seguir el tutorial que expliquen a la pàgina web del centre de recerca [25] i prèviament connectar-nos a la cadena de blocs que també expliquen en un tutorial [26]. Un cop connectats a Teranyina i feta la injecció de web3 que indica la guia, fem *deploy* del contracte i només hem de copiar l'adreça d'aquest i afegir-la al nostre codi. El resultat esperat és el mostrat en les següents imatges:

Figura 10.1: Compte de Metamask vinculat a Teranyina

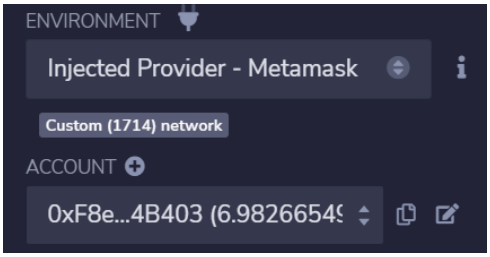


Figura 10.2: Contracte desplegat a Teranyina

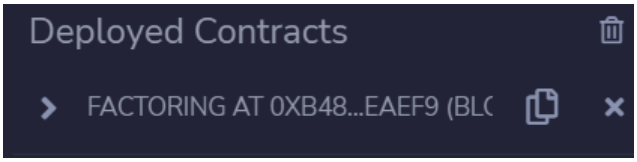
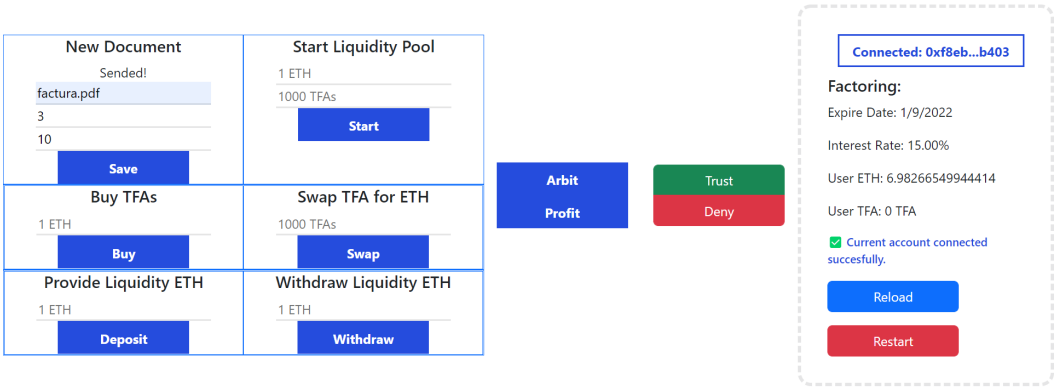


Figura 10.3: Interfície detectant el contracte



De cara als resultats s'ha decidit veure els donats per les proves fetes als testos de sistema i les simulacions fetes amb Excel, ja que seria fer exactament el mateix que amb Teranyina, la principal diferència és que l'entorn amb Ganache és local al nostre ordinador mentre que Teranyina no.

Primer de tot, compararem els resultats obtinguts amb la seva principal competència: els préstecs. És una de les alternatives més comunes i emprades a l'hora d'obtenir liquiditat a canvi d'endeutar-se. En aquest, intervenen dos agents: el creditor i el deutor. El primer proporciona finançament al segon a canvi que es retorni en una data acordada la quantitat més una extra coneguda com a interès. N'hi ha variants d'aquest, com en lloc de fer el pagament sencer a un dia en concret, es pot fer en terminis pagant una quantitat mensual al mes indexada a l'interès. Jo els tractaré com si fos a pagament únic.

Un cop descrits els dos procediments possibles per obtenir liquiditat, farem el cas d'estudi. Suposarem que tenim una factura amb una quantitat q. El client té dues alternatives, el facturatge tradicional basat en préstecs o el descentralitzat. Perquè sigui en igualtat de condicions, suposarem que tenen la mateixa taxa d'interès t.

Llavors podem definir el benefici obtingut per un creditor per la via de facturatge amb la següent funció:

$$f(q,t) = q \cdot t \tag{10.1}$$

Per tant, a mesura que incrementem la quantitat de diners a retornar, augmentarà el benefici que obtindrem. El mateix cas amb la taxa d'interès, però amb un ritme més lent, ja que és un percentatge. Però un ha d'estar atent, perquè si incrementem la quantitat, també pugem el risc o probabilitat de perdre la inversió. També cal dir que el facturatge tradicional només té un creditor. La principal diferència és que el benefici de tots els creditors serà el sumatori de totes les quantitats invertides. Aquestes són molt més probables que siguin quantitats petites, ja que qualsevol pot finançar però tampoc amb una gran inversió com a la tradicional. Per tant, la funció quedaria així:

$$g(q,t) = \sum_{i=1}^{n} q_i \cdot t \tag{10.2}$$

Ara veurem això amb un cas pràctic. Suposem que tenim una factura que té un valor per un import de 10.000 €. Tenim els dos casos de facturatge amb el mateix percentatge d'interès, del 15%.

TRADICIONAL		
Inversió	Import a pagar	Benefici
10.000,00 €	-11.500,00 €	1.500,00 €

Figura 10.4: Cas pràctic tradicional

En el descentralitzat, hem optat per unes inversions que la suma d'aquestes sigui igual als 10.000 € a finançar. Queda el següent:

DESCENTRALITZAT				
Inversió	Import a pagar	Benefici inversor	Import total	Benefici total
1.000,00 €	-1.150,00 €	150,00 €	-11.500,00 €	1.500,00 €
2.000,00 €	-2.300,00 €	300,00 €		
3.500,00 €	-4.025,00 €	525,00 €		
1.250,00 €	-1.437,50 €	187,50 €		
2.250,00 €	-2.587,50 €	337,50 €		

Figura 10.5: Cas pràctic descentralitzat

Aquí podem observar una de les principals diferències. Gràcies a la tokenització, qualsevol amb una petita quantitat pot invertir a finançar una factura i

rebre beneficis. A causa d'això, és molt més probable que la quantitat invertida sigui igual o més petita que el preu de la factura. Això es deu al fet que amplia l'oferta d'inversors disponibles que estan disposats a finançar. A més, dilueix el risc de perdre la inversió al ser més petita. El sistema és intrínsecament més segur també, ja que si es produeix un impagament, s'activa la garantia i també s'ha de retre comptes amb tots aquells que han donat confiança amb la inversió. Tot això recorda molt a l'arquitectura i fiabilitat tecnològica pròpia de Blockchain. Com es va veure a la implementació, l'usuari rep liquiditat progressivament a mesura que s'afegeix liquiditat al fons. Es tracta d'un cas molt semblant al dels edificis explicat abans a [Tokenització].

Amb tot això plegat, esmentaré els principals avantatges i inconvenients de cadascun dels procediments.

Avantatges:

✓ Oferta inversora ampliada: qualsevol amb una quantitat que vulgui pot participar i invertir-hi.

✓ Risc diluït: a l'haver-hi més participants, és més probable que es financi amb quantitats més petites.

✓ Més seguretat: presència de garantia i de molts inversors que si no es paga amb temps, desprestigiaran el nostre historial creditici. Molt risc per la part del propietari.

✓ Liquiditat progressiva: el propietari al fer l'arbitratge va rebent liquiditat a poc a poc.

✓ Interès dinàmic: la taxa d'interès és variable i s'incentiva al propietari a reduir-la proporcionant confiança al sistema.

Inconvenients:

× Pagament del gas: per cada transacció s'ha de pagar el gas i desincentiva inversions petites.

× Pèrdua impermanent: la inversió inicial pot tenir pèrdues respecte al final si el preu del criptoactiu pateix una fluctuació gran del seu valor. Per aquest motiu també pot tenir ingressos extraordinaris.

× Difícil verificació: encara que hi ha mitjans per donar fiabilitat és difícil convèncer a la societat de la fiabilitat del món digital envers el físic i tangible.

× Col·lateral: sembla contradictori fer aquest procés d'obtenir liquiditat i haver de proporcionar un actiu líquid al principi. A més el sistema implementat fomenta fons altament col·lateralitzats o sobrecol·lateralitzats.

× Interès dinàmic: pot fomentar la no verificiació, ja que aquesta fa que augmenti la taxa d'interès.

Cal dir que ambdues opcions són completament vàlides i que només es presenta com a possible alternativa a la tradicional. Aquesta també presenta avantatges molts bons i inconvenients, alguns d'aquests com per exemple: la flexibilitat i personalització entre creditor i deutor, el procediment és molt més simple que el tokenitzat, la fiabilitat del món real i la immediata liquiditat. El principal problema, en canvi, com s'ha pogut veure és el d'haver d'indicar la quantitat exacta que se sol·licita.

Amb això i les proves, podem veure que un dels principals objectius, desenvolupar un sistema de factoring amb l'ús de la tecnologia Blockchain, el podem considerar com satisfet. Un altre objectiu satisfet és el de la interfície gràfica i l'aplicació descentralitzada conjuntament. Els objectius que no s'han pogut assolir són el de la implantació en Alastria i l'ús de protocols.

Capítol 11

Conclusions

Podem dir que el projecte ha estat molt enriquidor, sobretot he estudiat i aplicat conceptes DeFi en primera persona: el desenvolupament d'un Automated Market Maker per a implementar una Liquidity Pool mitjançant Solidity, en el qual es pot fer intercanvis d'actius i dipositar o treure liquiditat; la tokenització d'un actiu emprant un estàndard, l'ERC-20; la creació d'un criptoactiu i definició de la seva estructura econòmica (tokenomics); aplicar la col·lateriltzació i fomentar que sigui elevat o que sobrepassi el límit; conjuntament amb una interfície, crear una aplicació descentralitzada.

A més, el sistema ha superat amb èxit diversos testos unitaris comprovant que s'inicialitza i es crea el fons correctament, faci bé els intercanvis d'actius, els dipòsits i la retirada de liquiditat siguin correctes, la taxa d'interès concordi amb la funció descrita prèviament i, finalment, que la ràtio de verificació funcioni perfectament. Seguidament, s'han agrupat tots aquests testos unitaris en diversos testos del sistema i descrit uns casos amb els seus resultats esperables. Podem concloure que el testeig ha estat satisfactori.

Respecte als requeriments funcionals, el sistema compleix en gran manera la majoria d'aquests. S'ha assolit gràcies al disseny proposat d'una aplicació descentralitzada formada per un frontend fet amb React.js i un backend amb Solidity. El primer està conformat per un model, una vista i un controlador. Mentrestant, el segon esta dividit en diferents contractes. Aleshores, el propietari pot: donar el nom del document, el seu valor i els dies que falten per a rebre el pagament; indicar els tokens amb el que vol fragmentar la factura i tot seguit s'afegeix a un fons de liquiditat amb el seu respectiu col·lateral; arbitrar i rebre el finançament progressivament; el sistema compensa la demanda de tokens afegint nova oferta al fons. Per altra banda, un usuari qualsevol pot consultar l'estat del fons comprovant quina quantitat hi ha de cada actiu i quanta posseïx ell, la data d'expiració, la taxa d'interès actual, i verificar o denegar el document. En últim lloc, els inversors poden intercanviar ethers per tokens (mitjançant el fons), consultar els tokens que tenen a disposició i obtenir el benefici un cop el propietari ha rebut els diners. Pel que fa als requeriments no funcionals: el sistema només accepta Ethers i Tokens com actius; tokenitza el document generant nous tokens ERC-20 (anomenat Tokfa o TFA); simula una taxa d'interès

situada inicialment al 15% i que pot canviar quan el propietari proporciona molt col·lateral o en excés i viceversa; dona com verificat el document al superar la ràtio 5:1; la interfície accepta ethers en lloc de weis per a facilitar la interacció amb l'usuari, els contractes, per altra banda, només operen amb weis.

Blockchain és una tecnologia molt innovadora i que té unes aplicacions boníssimes. Es podria considerar completament com un canvi de paradigma, ja que es poden trobar solucions perfectaments tant en el model tradicional com en Blockchain. L'aplicació descentralitzada feta al transcurs del projecte és un exemple clar. A més, té moltes semblances amb el procés de *crowdfunding* aprofitant al màxim les possibilitats que dona aquesta tecnologia. Per tot això, hi ha molt treball futur en aquest aspecte perquè és un món que està en constant ebullició i per explorar. Més endavant, al capítol 12 en detallo uns quants. També obre moltes portes professionalment i actualment els desenvolupadors en Blockchain són bastant demandats.

Per acabar, recomano molt aquesta temàtica per gent que li agrada molt la informàtica i també l'economia en general. El projecte m'ha agradat més del que esperava. Com ja s'ha dit, les temàtiques són infinites i és molt probable que hi hagi una que s'ajusti als gustos personals de cadascú.

CAPÍTOL 12

Treball futur

Al ser Blockchain una tecnologia molt recent i poc explorada, n'hi ha moltes possibilitats de cara a un treball futur. Aquí n'hi esmentaré les tasques que es podrien considerar a fer:

1. Fer servir una altra fórmula més complexa al fons de liquiditat. En aquest projecte s'ha fet servir la fórmula $x \cdot y = k$, però es poden considerar altres alternatives. En aquest article comenten vàries [27]

2. Canviar la criptomoneda per una stablecoin per evitar les pèrdues impermanents. Moltes estan basades en l'estàndard ERC-20. Simplement, s'hauria de substituir els ethers per la moneda. Per això només cal indicar l'adreça de la stablecoin com aquesta [28].

3. Fer el mateix projecte però amb altre llenguatge per contractes intel·ligents. Per exemple, Solana fa servir Rust o C [29] o Ethereum un llenguatge de nivell més baix, Yul [30].

4. Utilitzar un altre estàndard. El projecte s'ha fet amb el clàssic ERC-20, però es pot fer servir tokens fraccionats, que fan servir tant l'ERC-20 i l'ERC-721 o directament l'ERC-1155 [31].

5. Moltes més i que segurament no coneixem. Per exemple desplegar en altres blockchains o l'ús de protocols com Ntropika [32].

Respecte al mateix projecte en si, és obvi que falta la implantació en Alastria, com ja s'ha comentat per temes logístics i tècnics no està disponible el node al moment de desenvolupar el projecte. També s'ha descartat l'ús de protocols, ja que he prioritzat el bon transcurs del projecte i aquests com a secundari. Tampoc s'ha aprofundit molt a l'estètica, disseny i control de dades de la interfície.

Bibliografia

[1] La Vanguardia. Las micropymes y pymes retrasan el pago a sus proveedores por el coronavirus, (2020). https://www.lavanguardia.com/economia/20200518/481242765890/micropymes-pymes-retrasan-pagos-proveedores-coronavirus.html. (Cited on page 1.)

[2] El País. Casi el 30% de las empresas españolas tenía problemas de liquidez antes del Covid-19, (2020). https://cincodias.elpais.com/cincodias/2020/07/08/economia/1594205889_318379.html. (Cited on page 1.)

[3] Banco de España. Las necesidades de liquidez y la solvencia de las empresas no financieras españolas tras la perturbación del Covid-19, (2020). https://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosOcasionales/20/Fich/do2020.pdf. (Cited on page 1.)

[4] Finanzarel. Obtén liquidez sencilla y rápidamente para impulsar tu negocio. https://www.finanzarel.com. (Cited on page 1.)

[5] Meriem Guerar, Luca Verderame, Alessio Merlo, Mauro Migliardi. Blockchain-based risk mitigation for invoice financing. https://www.researchgate.net/publication/334578896_Blockchain-based_risk_mitigation_for_invoice_financing. (Cited on page 8.)

[6] DotCSV. HOY SÍ vas a entender QUÉ es el BLOCKCHAIN - (Bitcoin, Cryptos, NFTs y más), (2019). https://www.youtube.com/watch?v=V9Kr2SujqHw. (Cited on page 8.)

[7] VisualPolitik. La revolución BLOCKCHAIN: el FIN de la BUROCRACIA - VisualPolitik, (2019). https://youtu.be/WzA-5jhan8w. (Cited on page 8.)

[8] Josep Lluís de la Rosa Esteva. *Finances descentralitzades (DeFi)*. (Cited on page 8.)

[9] Generalitat de Catalunya CIDEM, Catalunya Innovació. FACTURATGE. (Cited on page 8.)

[10] Ethereum. Solidity Documentation, (2022). https://buildmedia.readthedocs.org/media/pdf/solidity/develop/solidity.pdf. (Cited on page 8.)

[11] Alberto Lasa. Tutoriales de Solidity en Español - Curso de Smart Contracts. https://youtube.com/playlist?list=PL-jD2YjEmwDqWgkBOiXEEXlHsiAUIqfM5. (Cited on page 8.)

[12] Ethereum. ESTÁNDAR DE TOKEN ERC-20, (2022). https://ethereum.org/es/developers/docs/standards/tokens/erc-20/. (Cited on pages 8, 32 and 45.)

[13] Whiteboard Crypto. Crypto Coin vs Token (Differences + Examples), (2021). https://youtu.be/422HORNUfkU. (Cited on page 8.)

[14] BBVA. ¿Por qué varía tanto el precio de las criptomonedas?, (2021). https://www.bbva.com/es/por-que-varia-tanto-el-precio-de-las-criptomonedas/. (Cited on page 8.)

[15] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. Smart contract: Attacks and protections. *IEEE Access*, PP:1–1, 01 2020. (Cited on pages 9 and 35.)

[16] Vishnu Sivan. Develop your first Dapp with Web3.js, (2022). https://coinsbench.com/develop-your-first-dapp-with-web3-js-c038bc10710b. (Cited on page 9.)

[17] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, (2009). https://bitcoin.org/bitcoin.pdf. (Cited on page 19.)

[18] Binance Academy. ¿Qué es Proof of Stake (PoS)?, (2022). https://academy.binance.com/es/articles/proof-of-stake-explained. (Cited on page 22.)

[19] Yahoo Finance. Ethereum USD, (2022). https://finance.yahoo.com/chart/ETH-USD. (Cited on page 27.)

[20] Eric Carriere. ¿Por qué se colapsó Terra LUNA y UST? ¿cómo ocurrió?, (2022). https://coinmotion.com/es/causas-colapso-terra-luna-ust/. (Cited on page 29.)

[21] Abhishek Chauhan. A Complete Guide to Building Ethereum dApps: Front-end and Back-end, (2022). https://betterprogramming.pub/a-complete-guide-to-build-ethereum-dapps-front-end-and-back-end-6fa44b66554b. (Cited on pages 36 and 71.)

[22] OpenZeppelin. OpenZeppelin Contracts, (2022). https://github.com/OpenZeppelin/openzeppelin-contracts. (Cited on page 41.)

[23] Enrique Sambola Giménez. Factoring, 2022. https://github.com/QuiqueSG/Factoring. (Cited on page 43.)

[24] Web3 Academy. Integrate Your Smart Contract With a Frontend. https://www.web3.university/tracks/create-a-smart-contract/integrate-your-smart-contract-with-a-frontend. (Cited on page 58.)

[25] Centre TECNIO EASY. Exemple ús Teranyina amb Remix. https://teranyina.centreeasy.com/exemple-us-teranyina-amb-remix/. (Cited on page 71.)

[26] Centre TECNIO EASY. Connectar Metamask a Teranyina. https://teranyina.centreeasy.com/connectar-metamask-a-teranyina/. (Cited on page 71.)

[27] Medium. A Mathematical View of Automated Market Maker (AMM) Algorithms and Its Future. https://medium.com/anchordao-lab/automated-market-maker-amm-algorithms-and-its-future-f2d5e6cc624a. (Cited on page 78.)

[28] Polygon. USDC Stablecoin Code. https://polygonscan.com/address/0xdd9185db084f5c4fff3b4f70e7ba62123b812226#code. (Cited on page 78.)

[29] Solana. Developing with Rust. https://docs.solana.com/developing/on-chain-programs/overview. (Cited on page 78.)

[30] Ethereum. Yul. https://docs.soliditylang.org/en/v0.8.15/yul.html. (Cited on page 78.)

[31] Ethereum. ERC-1155 MULTI-TOKEN STANDARD. https://ethereum.org/en/developers/docs/standards/tokens/erc-1155/. (Cited on page 78.)

[32] Ntropika. Ntropika. https://www.ntropika.io. (Cited on page 78.)