testDEX: Un DEX simple en la xarxa Elrond

Sergio González Rubio
Màster universitari de Ciberseguretat i Privadesa
Sistemes de blockchain

Consultor: Josep Lluís de la Rosa Esteva
Professor: Victor Garcia Font

9 de maig de 2022

FITXA DEL TREBALL FINAL

Títol del treball:	*testDEX: Un DEX simple en la xarxa Elrond*
Nom de l'autor:	*Sergio González Rubio*
Nom del consultor/a:	*Josep Lluís de la Rosa Esteva*
Nom del PRA:	*Victor Garcia Font*
Data de lliurament (mm/aa-aa):	*juny/2022*
Titulació o programa:	*Màster universitari de Ciberseguretat i Privadesa*
Àrea del treball final:	*Sistemes de blockchain*
Idioma del treball:	*Català*
Paraules clau:	*Elrond, blockchain, EGLD, smart contract, swap*

Resum del treball:

L'adveniment Bitcoin i posteriorment d'Ethereum varen definir les bases de la tecnologia *blockchain*. Cal destacar que han suposat una disrupció tecnològica, econòmica i fins i tot política, despertant un interès acadèmic creixent i nous casos d'ús. En aquest context, se'ns presenten reptes per aconseguir una descentralització plena que sigui capaç de substituir els sistemes centralitzats tradicionals de transaccions electròniques i moviment de capitals. Alt consum d'energia, baixa taxa de transaccions, escalabilitat, seguretat o interoperabilitat són alguns dels reptes plantejats per les tecnologies actuals de *blockchain* que resol Elrond. Aquest treball presenta un exemple d'implementació de *swaps* entre criptovalors amb l'objectiu de comprovar que la proposta de *blockchain* pública d'Elrond ha vençut les limitacions actuals de Bitcoin i Ethereum.

Abstract:

The advent of Bitcoin and later of Ethereum described the foundations of blockchain technology. It should be noted that they have led to a technological, economical and even political disruption, arousing growing academic interest and new use cases. In this context, we are being challenged to achieve full decentralization that is capable of replacing traditional centralized systems of electronic transactions and movements of capital. High energy consumption, low transaction rates, scalability, security or interoperability are some challenges of current blockchain technologies that Elrond solves. This paper presents an example of the implementation of swaps between tokens in order to verify that Elrond's public blockchain proposal has overcome the current limitations of Bitcoin and Ethereum.

Índex

Índex de figures

1 Introducció

1.1 Context i justificació del treball

El *white paper* de **Bitcoin** fou presentat l'any 2008 per una o diverses persones ocultes rere el pseudònim de Satoshi Nakamoto [4]. Tot i que existeixen intents anteriors[1], fou la primera solució "creïble" [8] que va permetre transferir fons sense la intervenció de terceres parts (cosa que inclou a bancs centrals encarregats d'emetre la moneda). Gràcies a la implementació d'un programari client –lliure i de codi obert– s'estableix una xarxa d'igual a igual (*peer-to-peer*) on es connecten nodes –no controlats– que verifiquen i emmagatzemen en una base de dades pública la comptabilitat dels moviments entre usuaris de la xarxa. Les operacions s'agrupen en blocs, que s'enllacen i xifren per assegurar que no hi hagi modificacions malicioses posteriors. Els càlculs per realitzar el xifratge tenen cert nivell de dificultat i comporten una despesa energètica considerable, per aquest motiu s'incentiva els nodes "miners" amb una recompensa quan aconsegueixen crear un nou bloc a la cadena. Aquest és el motiu que coneguem aquesta tecnologia que implementa aquest llibre de comptes compartit i immutable com a *blockchain*[2]. Així mateix, com afirma V. Buterin [8], l'altra gran aportació és la manera com aplica el concepte de *Proof-of-Work* (PoW) perquè els nodes arribin a un consens per validar les transaccions solucionant certs problemes (com, per exemple, el de la doble despesa). Amb PoW es pretén evitar que nodes de la xarxa tinguin comportaments indesitjats. Com ja s'ha remarcat, a Bitcoin els càlculs de xifratge per crear un nou bloc comporten molta feina computacional, però la validació d'un bloc ja creat requereix un esforç molt inferior. Si dos nodes distribueixen simultàniament diferents versions del següent bloc, el que tingui la cadena més llarga serà el que els nodes acceptin com a vàlid i descartaran la resta [4]. Finalment, el protocol de Bitcoin també disposa d'un llenguatge de *scripting*, encara que amb limitacions importants (per exemple, no és Turing complet i no té estat [8]).

El 2014 V. Buterin va presentar el *white paper* d'**Ethereum** [8] (la xarxa es va posar en producció el 30 de juliol de 2015 [11]). Al igual que amb Bitcoin, s'utilitza el concepte PoW com a mecanisme de consens entre nodes, però s'incorpora un llenguatge anomenat "Solidity" (que sí que és Touring complet) per a la creació de "*smart contracts*" i d'aplicacions descentralitzades (o "dApps"). Els contractes intel·ligents són programes desplegats (o guardats) a la cadena de blocs que s'executen automàticament quan es compleixen certes condicions. A més, una dApp és una aplicació que funciona sense la necessitat de servidors centrals (gràcies a la tecnologia

[1] Per exemple el "b-money" de W. Dai (1998) [5], el "Reusable Proofs of Work" de H. Finney (2005) [6] o el "Bit gold" de N. Szabo (2008) [7].

[2] No és l'objectiu del present treball explicar els sistemes de *blockchain*, però una gran explicació gràfica la realitza A. Brownworth [9][10].

descrita). En el moment en què redacto aquestes línies, la versió d'Ethereum 2.0 "Serenity" encara no està completament desenvolupada. Aquest nou *fork* ha d'introduir millores com substituir el PoW per *Proof-of-Stake* (PoS), per reduir el consum d'energia i introduir tècniques de *sharding* per augmentar l'eficiència (la versió actual de la xarxa no arriba a suportar les 20 transaccions per segon [12]).

Amb Bitcoin i Ethereum consolidats, "The Elrond Team" va publicar el 19 de juny de 2019 el *white paper* d'una nova solució de cadena de blocs pública sota el títol "**Elrond**: A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake". Entre els reptes que plantejaren [1][2]:

- Descentralització plena (sense necessitat de tercers).
- Seguretat robusta de les transaccions i prevenint qualsevol vector d'atac conegut.
- Alta escalabilitat, arribant a un nivell de rendiment similar al d'algun dels serveis similars amb arquitectura centralitzada.
- Eficiència a tots els serveis de xarxa amb el mínim consum energètic i esforç computacional.
- Millora de l'emmagatzematge i la sincronització de dades.
- Interoperabilitat entre cadenes de blocs des del disseny.

Tot i la data de publicació del *white paper*, l'equip d'Elrond assegura que el seu *mainnet* és actiu des de l'any 2018. A més, citen com a aconseguides en la data de redacció del present treball les següents fites [1][13]:

- Primera arquitectura de *blockchain* en producció amb fragmentació d'estat (*state sharding*).
- 1,5k TPS (escalable a més de 100k TPS[3]), latència de 6s i cost de $0,001 per transacció.
- Maiar App[4] (moneder mòbil d'Elrond), Elrond Web Wallet[5] i Maiar DeFi Wallet[6].
- *Smart Contracts*, *Staking & Delegation*, *Tokens*.
- Maiar DEX[7].

[3] S'han fet proves a la *testnet* amb pics de més de 260k TPS [1].
[4] https://maiar.com/
[5] https://wallet.elrond.com/
[6] Per aconseguir el *plugin* per al teu navegador: https://getmaiar.com/defi
[7] https://maiar.exchange/

- DeFi 2.0: Préstecs, sintètics.
- Validat mitjançant múltiples auditories per part de l'empresa Trail of Bits[8] i d'altres.

Referent al darrer punt, he cercat informació sobre quines auditories s'han fet i la informació que he trobat és escassa (pareix que per qüestions de seguretat han estat majoritàriament auditories internes [14]). Sí que apareix un acord de col·laboració per emprar les eines de l'empresa Runtime Verification[9] i que s'han oferit recompenses a *white-hat hackers* [15][16].

Tot i que hi ha altres projectes prometedors com Solana, Avalanche, Tron o Tezos [17], en la meva opinió, l'evolució d'Elrond des de l'aparició del seu *white paper* dona peu a parar el nostre interès acadèmic en aquest projecte i no en els altres. És per aquest motiu que es proposa desenvolupar un **DEX** molt simple que permeti fer intercanvis entre diferents criptomonedes emprant el protocol ***automated market makers*** (AMM) [18] amb la intenció de **posar a prova la xarxa d'Elrond**.

1.2 Objetius del treball

A nivell molt genèric, els quatre grans objectius que pretenc assolir són:

- L'anàlisi i estudi de la tecnologia de *blockchain* d'Elrond des d'un punt de vista acadèmic.
- La definició d'una proposta d'arquitectura per desenvolupar dApps en la xarxa d'Elrond.
- El desenvolupament d'un exemple pràctic emprant l'arquitectura anterior.
- La definició de proves per comprovar característiques de la xarxa d'Elrond.

Vull destacar en aquest apartat que no he trobat cap altre treball acadèmic sobre la xarxa d'Elrond en llengua catalana.

1.3 Enfocament i mètode seguit

Després de revisar ràpidament l'estat de l'art de les tecnologies *blockchain*, ha estat una aposta personal fer servir Elrond. És un projecte novedós i sé que, per aquest motiu, disposaré de menys bibliografia que amb Bitcoin o Ethereum. Aquest darrer fet espero que, més que ser una cosa negativa, em permeti fins i tot trobar nous objectius a mesura que vagi investigant.

[8] https://www.trailofbits.com/
[9] https://runtimeverification.com/

Així mateix, una bona planificació del projecte és ben necessària, on vull destacar que la fase d'investigació es realitzarà de forma paral·lela a la resta.

Les **fases** que es desenvoluparan són les següents:

- **Plantejament del problema**: Es duu a terme una entrevista amb Josep Lluís de la Rosa Esteva i es pacta desenvolupar un **DEX** emprant el protocol AMM amb el propòsit de provar i comprovar algunes de les característiques de la tecnologia d'Elrond. Com s'ha comentat, previ a l'entrevista, va haver-hi una feina d'investigació ràpida sobre l'estat de l'art en els sistemes *blockchain* que s'ha tocat només de passada per justificar l'elecció d'Elrond.
- **Pla de treball**: Es definiran els recursos necessaris per fer el projecte, les tasques a fer i la seva temporalització.
- **Revisió de la tecnologia d'Elrond**: Es descriurà a tall de resum la tecnologia i s'enumeraran els recursos que hi ha disponibles per desenvolupar en aquesta xarxa.
- **testDEX**:
  - **Anàlisi**: Definició dels requeriments i dels models relacionats amb aquests.
  - **Disseny**: Realització dels models que defineixen el disseny del sistema i l'arquitectura.
  - **Implementació**: Escriptura del codi font.
  - **Posada en producció**: Desplegament en producció del projecte.
  - **Proves**: La fase de proves es durà a terme de forma paral·lela a la implementació i a la de posada en producció. S'han de definir uns criteris de qualitat mínims abans de posar l'aplicatiu en producció (on es continuarà realitzant proves).
- **Redacció de la memòria**: Es farà de forma paral·lela mentre es duen a terme les fases anteriors.
- **Creació del vídeo de la presentació final**: Necessari per defensar el treball.
- **Investigació i formació**: Pel plantejament inicial del treball, es durà a terme a la vegada que es realitzen la resta de fases.

1.4 Planificació del treball

Els recursos necessaris per desenvolupar el projecte són mínims: Un ordinador personal amb el seu sistema operatiu i un editor de codi (*Visual Studio*

Code[10]). Per altra banda, les tasques a realitzar han estat enumerades en el punt anterior i seran descrites en el seu apartat corresponent. Gràficament en un diagrama de Gantt queden de la següent forma:

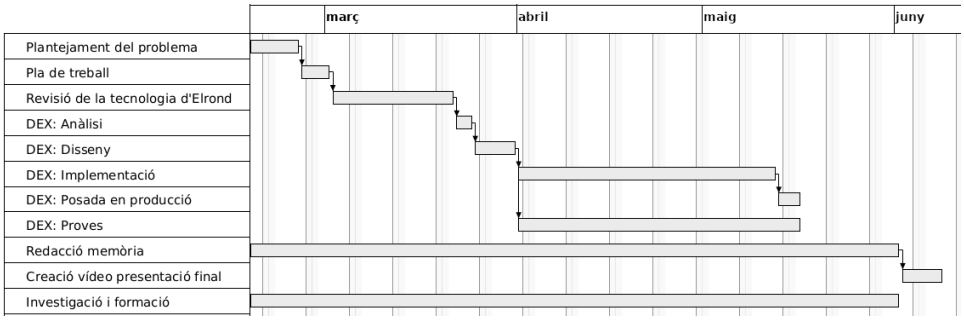


Figura 1: Diagrama de Gantt del projecte (creat amb GNOME Planner 0.14.6).

La següent figura presenta la temporalització de les fases del proyecte, s'ha de considerar que un dia de feina correspon a una jornada de quatre hores:

1	Plantejament del problema	17/02/22	24/02/22	6d
2	Pla de treball	25/02/22	01/03/22	3d (coincideix amb l'entrega de la PAC 1)
3	Revisió de la tecnologia d'Elrond	02/03/22	21/03/22	14d
4	DEX: Anàlisi	22/03/22	24/03/22	3d
5	DEX: Disseny	25/03/22	29/03/22	3d (coincideix amb l'entrega de la PAC 2)
6	DEX: Implementació	30/03/22	12/05/22	32d (la PAC 3 caurà en mig de la implementació)
7	DEX: Posada en producció	13/05/22	16/05/22	2d
8	DEX: Proves	01/04/22	16/05/22	Durant les fases d'implementació i proves
9	Redacció memòria	17/02/22	31/05/22	Durant tot el projecte (coincideix amb l'entrega de la PAC 4)
10	Creació vídeo presentació final	01/06/22	08/06/22	6d
11	Investigació i formació	17/02/22	01/06/22	Durant tot el projecte

Figura 2: Temporalització del projecte.

[10]Existeix el *plugin* "Elrond IDE": https://marketplace.visualstudio.com/items?itemName=Elrond.vscode-elrond-ide

1.5 Breu sumari de productes obtinguts

"No hay que entrar en detalle: la descripción detallada se hará en el resto de capítulos."

1.6 Breu descripció dels altres capítols de la memòria

"Explicación de los contenidos de cada capítulo y su relación con el trabajo en global."

2 Revisió de la tecnologia d'Elrond

2.1 Visió general de l'arquitectura

Elrond és una cadena de blocs[11] pública i d'alt rendiment. Altres punts a destacar són [1][19]:

1. **Entitats (usuaris i nodes)**: Els usuaris despleguen transaccions a la xarxa, en forma de transferència de valor o executant un *smart contract*. Per altra banda, els nodes són dispositius en la xarxa que executen el programari d'Elrond i processen les transaccions.

2. ***Shards***[12]: Particions de la xarxa d'Elrond que permeten escalar-la. La feina de cada *shard* és gestionar una part de l'estat i processar paral·lelament la corresponent part de la transacció.

3. ***Adaptive State Sharding***: S'aplica a tots els nivells (transacció, dades i xarxa) de forma adaptativa (això és, divisió i unió dinàmica de *shards* segons nombre de validadors disponibles i càrrega de la xarxa).

4. ***Validador***: Node de la xarxa Elrond amb almenys 2500 EGLD en *staking* (o bloquejats[13]) que processa les transaccions i assegura la xarxa per la seva participació en el mecanisme de consens per validar els blocs (serà recompensat amb les tarifes –*fees*– de les transaccions).

5. ***Secure Proof of Stake***: Els blocs són validats per consens entre els validadors del grup de consens, que és completat en dues passes de comunicació emprant una modificació de les signatures múltiples de Boneh-Lynn-Shacham (BLS) [20]. El grup de consens és seleccionat aleatòriament i només és possible conèixer la seva composició amb una ronda d'antelaci.

6. **Alta resiliència**: Capacitat de recuperar-se d'atacs maliciosos pel canvi de nodes entre els *shards* (en cada "època" un terç dels nodes seran reubicats per prevenir connivències entre els mateixos).

7. **Font d'aleatorietat segura**: Utilitzant la signatura BLS, cosa que la fa no esbiaixada i impredictible.

8. **Elrond WASM VM**: Màquina virtual específica que permet executar contractes intel·ligents escrits en qualsevol llenguatge de programació que es permeti compilar amb *WebAssembly*[14].

[11] En el present treball empraré indistintament "*blockchain*" i "cadena de blocs" (traducció del terme proposada pel Termcat).

[12] No hi ha entrada al Termcat pel terme "shard", jo l'he traduït per "fragment" (empraré ambdues paraules indistintament en el present treball).

[13] El Termcat encara no ha traduït el terme, és una traducció pròpia que penso que fa entendre el concepte.

[14] https://webassembly.org/

9. **Contractes intel·ligents**: Que s'executaran emprant l'*Adaptative State Sharding*. Elrond recomana que siguin escrits en Rust, però –com s'ha dit– es poden utilitzar altres llenguatges. Una característica única, comparant per exemple amb Ethereum, és que es permet modificacions dels mateixos una vedada desplegats [21] (per exemple per corregir errors).

10. **Execució ràpida de transaccions *cross-shard***: Gestionada de forma nadiua a nivell de protocol fent servir un algorisme d'expedició (*dispatching*) i un algorisme d'encaminament.

11. ***Metachain***: Cadena de blocs que s'executa en un *shard* especial. El seu rol principal no és processar transaccions sinó que és notificar i autenticar les capçaleres dels blocs processats.

12. **Elrond Gold (EGLD)**: Criptovalor (*token*) nadiu de la xarxa Elrond que serveix de mitjà de pagament per a les transaccions. Ajuda en el desplegament de dApps, en l'execució de contractes intel·ligents i també s'empra com a mecanisme de pagament pels validadors.

2.2 Entitats

Bàsicament, en la xarxa Elrond hi ha dos tipus d'entitats: usuaris i nodes [1]. Un **usuari** és qualsevol ens que gestioni un dels comptes –o més– de la xarxa Elrond. Això és un parell de claus criptogràfiques (una pública i una privada) que li permetran enviar transaccions signades a la xarxa. Els comptes tenen associat una quantitat d'EGLD que es coneix com a balanç i a més tenen un espai d'emmagatzematge per a valors arbitraris (com per exemple informació sobre *tokens* creats pels usuaris). Els comptes s'identifiquen de forma unívoca per una adreça que coincideix amb la clau pública de l'usuari (32 bytes i fa servir la representació Bech32). Normalment, els usuaris gestionen els seus parells de claus emprant unes aplicacions informàtiques que s'anomenen carteres (o *wallets*)[15]. Per altra banda, els **nodes** són dispositius connectats a la xarxa d'Elrond que realitzen les operacions sol·licitades pels seus usuaris. Els nodes poden ser passius (*observers*) o actius (*validators* i *fishermen*). Els validadors s'encarreguen del consens, d'afegir blocs i mantenir l'estat, essent premiats per la seva contribució. Els **validadors** són identificats de forma única per una clau pública BLS de 96 bytes. Per garantir el correcte funcionament dels nodes, els validadors han de tenir en *staking* com a mínim 2500 EGLD. Sense bloquejar EGLD els nodes poden fer d'**observadors**, però no rebran cap recompensa. Aquests darrers són membres passius de la xarxa que poden actuar com a interfície de lectura i retransmissió. Poden ser complets (*full)*, mantenint tota la història de la

[15]En el present treball empraré els termes "cartera" o "moneder" indistintament per referir-me al concepte de "*wallet*".

cadena de blocs, o lleugers (*light*), mantenint només 2 èpoques de l'historial de la cadena de blocs. Finalment, trobem els **pescadors** (*fishermen*). La seva tasca és verificar la validesa dels blocs després d'haver estat proposats, detectant així actors maliciosos [19]. Rebran també una recompensa i aquest rol pot ser exercit per observadors o validadors que no formin part de la ronda de consens en curs.[16]

2.3 Cronologia

En la xarxa d'Elrond s'organitza el temps en **rondes (*rounds*) i èpoques (*epochs*)** [1]. A la primera ronda de la primera època se l'anomena ronda de gènesi (*genesis round*), que òbviament coincideix amb la fase d'arrencada de la xarxa.

Les **rondes** tenen una durada fixa (que actualment és configurada en 5 segons). Per l'arquitectura en *shards* de la xarxa, en cada ronda només es podrà afegir un bloc a la cadena de blocs del *shard*. Òbviament, si no s'arriba a consens o quan el líder del grup de consens designat és fora de línia i no pot proposar un bloc, pot haver-hi rondes en què no s'afegeixi cap bloc a la *blockchain*.

Una **època** és una seqüència de rondes consecutives en què la configuració de la xarxa no canvia. El nombre de rondes en una època es calcula perquè aquesta darrera duri 24 hores (ja que aquesta és la configuració actual de la duració d'una època). Quan hi ha un canvi d'època s'aprofita per adaptar la topologia de la xarxa segons el nombre de validadors disponibles i càrrega d'aquesta, a més de per acomplir amb altres tasques per tancar l'època anterior (com calcular les recompenses per als validadors).

2.4 *Secure Proof of Stake*

El funcionament *Secure Proof of Stake* (SPoS) d'Elrond es pot desgranar en les següents passes [1][2]:

1. La font d'aleatorietat per seleccionar els validadors per al consens es calcula a partir del bloc anterior, que és signat pel líder de consens –també conegut com a proposador de blocs o *block proposer*– de la ronda que acaba. Això implica que aquesta font d'aleatorietat no podrà ser coneguda amb més d'una ronda d'antelació.

2. Se selecciona el grup de consens, compost per validadors i un únic *block proposer*. Una vegada coneguda la font d'aleatorietat, el procediment

[16] No es diu explícitament en la bibliografia però investigant pels grups de Telegram d'Elrond (concretament en `https://t.me/ElrondValidators`) vaig descobrir que el concepte de "*fisherman*" encara no està implementat.

de tria del grup és determinista (triga menys de 100ms[17] i no té requeriments de comunicació). S'ha de considerar que per triar els nodes es té en compte la quantitat d'EGLD en *staking* i una qualificació individual –revisada al final de cada època– que es basa en el comportament passat. Per exemple, es davallarà la seva puntuació si no proposa el bloc perquè és fora de línia, es detecta una activitat maliciosa, etc. Aquesta "meritocràcia" anima els propietaris dels nodes a tenir-los en bon funcionament.

3. El líder de consens (o validació) produeix el bloc per a la nova ronda. Si, pel motiu que sigui, no s'ha creat un bloc en una finestra de temps s'utilitzarà la font d'aleatorietat de l'últim bloc per seleccionar un nou grup de consens.

4. El líder de consens[18] envia el bloc que acaba de proposar als validadors.

5. Aquests darrers components del grup validen i també signen el bloc rebut, basant-se en una modificació de *Practical Byzantine Fault Tolerance* (pBFT).

6. Els validadors trameten les signatures al *block proposer*.

7. El proposador agrega les signatures i distribueix el bloc.

8. El *hash* del bloc, la signatura, les proves d'inclusió i el nombre de *shard* són tramesos a la *Metachain*.

Aquestes passes es poden veure resumides en la Fig. 3.

2.5 *Adaptive State Sharding*

El *sharding* va sorgir originàriament en el camp de les bases de dades com un mètode per distribuir les dades entre múltiples màquines amb l'objectiu de suportar alt rendiment amb grans volums d'informació [22]. Elrond empra aquesta tècnica d'escalat horitzontal per particionar tant la xarxa com l'estat i processament de les transaccions amb la finalitat que diferents nodes desenvolupin la seva tasca en paral·lel (amb el consegüent augment de l'eficiència). S'ha de destacar també que s'empren els tres tipus principals de *sharding* [1]:

[17] El SPoS d'Elrond es basa en la premissa que un actor malèvol només té el temps que dura una ronda per adaptar-se i intentar influir en el bloc que es proposarà.

[18] Per aclarir del tot aquest punt, en la bibliografia emprada, líder de consens, líder de validació i proposador de bloc (*block proposer*) fan referència al mateix concepte.

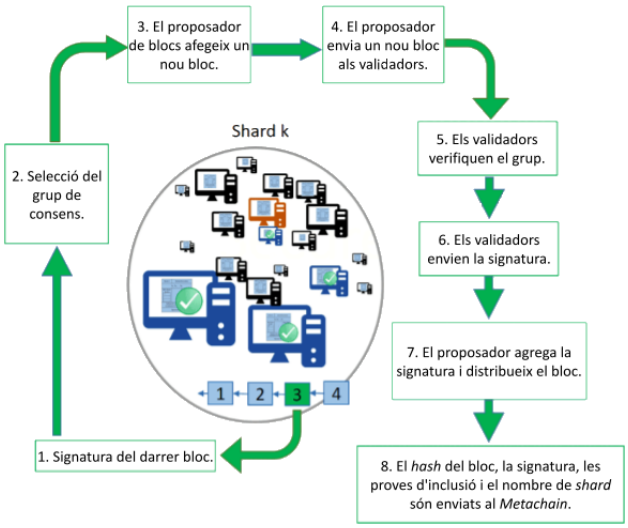


Figura 3: *Secure Proof of Stake*. (Font: Modificació d'imatge dins l'apartat corresponent a SPoS a [1]).

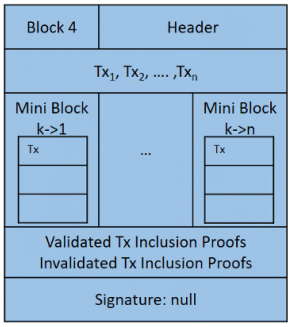


Figura 4: Estructura d'un bloc en la xarxa Elrond. (Font: Imatge dins l'apartat corresponent a SPoS a [1]).

- ***Network sharding***[19]: Gestiona com els nodes es distribueixen i agrupen entre els diferents *shards*. S'ha de tenir en compte que les comunicacions dins del *shard* són més ràpides que fer una propagació dels missatges a tota la xarxa (*broadcast*). Però s'ha de tenir en compte que si un atacant arriba a controlar un *shard* sencer suposaria un greu problema de seguretat.
- ***Transaction sharding***: S'ocupa la forma en què les transaccions s'assignen als *shards* encarregats del seu processament. Les transaccions són assignades a un *shard* de forma determinista emprant les adreces de les transaccions.
- ***State sharding***: Cada *shard* només manté una part de l'estat. Els dos mètodes anteriors, si no es combinessin amb aquest, haurien d'emmagatzemar una còpia sencera de tot l'estat. Així, si els comptes implicats en una transacció resideixen en *shards* diferents, l'execució de la transacció implicaria l'intercanvi de missatges entre nodes per modificar els respectius estats. Per augmentar la tolerància a atacs maliciosos quan acaba una època es redistribueixen entre els diferents *shards* un subconjunt dels nodes.

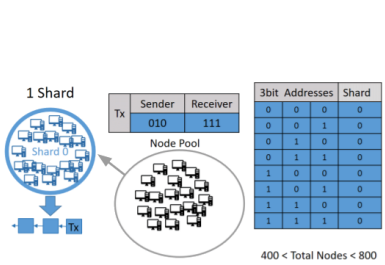
Com a resultat d'aquesta fusió s'aconsegueix [1]:

[19] Aquest concepte es podria traduir al català com "fragmentació de la xarxa" però he preferit deixar el terme anglès. El mateix passa amb *transaction sharding* i *state sharding*, que es podrien traduir respectivament per "fragmentació de les transaccions" i "fragmentació de l'estat".

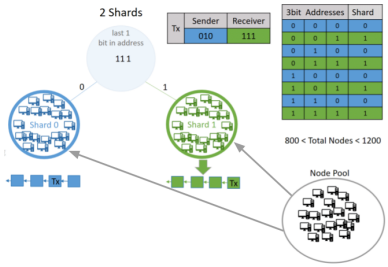
- **Escalabilitat sense afectar a la disponibilitat**: En la Fig. 5a podem observar la xarxa amb només un *shard*. Les figures 5b i 5c mostren una arquitectura amb dos i tres *shards*[20]. Sense entrar en detall, es pot veure que augmentant o disminuint el nombre de *shards* –gràcies a la forma en què s'assignen les adreces al seu fragment– s'evita temps d'inactivitat per qüestions de configuració.

- **Expedició (*dispatching*) ràpida i traçabilitat**: La Fig. 5c mostra com es calcula el *shard* de destí (les adreces de color blau aniran al 0, les verds a l'1 i les grogues al 2).

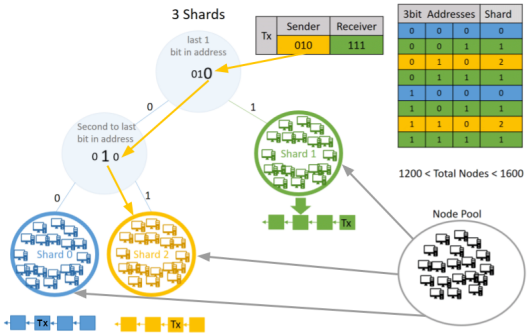
- **Eficiència i adaptabilitat**: El sistema permet processar transaccions en paral·lel i adaptar-se a la càrrega de treball i/o estat de la xarxa. Però s'ha de destacar que la distribució dels *shards* hauria de ser tan equilibrada com fos possible. Si ens fixem en la Fig. 5c veiem que no és equilibrada (ja que el nombre de *shards* no és una potència de 2).



(a) 1 *shard*



(b) 2 *shards*



(c) 3 *shards*

Figura 5: La xarxa d'Elrond configurada amb un (a), dos (b) i tres (c) *shards*. (Font: Modificació d'imatge dins l'apartat *Adaptive State Sharding* a [1]).

Una altra cosa a destacar, referent al fraccionament de l'estat, és que els

[20] En [2] es pot trobar una funció en pseudocodi amb la que calcular el nombre òptim de *shards*.

nodes guarden una **còpia de l'estat dels seus germans (redundància)** per aportar **tolerància a fallades** (Fig. 6).

Finalment, s'ha de remarcar el fet que els **nodes es barregin entre *shards* al final de cada època** per evitar la connivència entre nodes maliciosos. Això no es fa amb tots els nodes sinó que només es fa amb un nombre controlat de validadors que seran redistribuïts de manera no determinista i uniforme (aquesta tasca la realitza la *metachain* emprant una font d'aleatorietat procedent del bloc anterior de la metacadena). Es fa així, i no reorganitzant tots els nodes, per maximitzar la seguretat amb la mínima introducció de latències en el sistema. Els nodes triats es col·locaran als seus nous *shards* en una llista d'espera durant tota l'època actual fent la resincronització amb el nou fragment. Després el node es pot convertir en un validador elegible i unir-se al *shard* efectivament.[21]

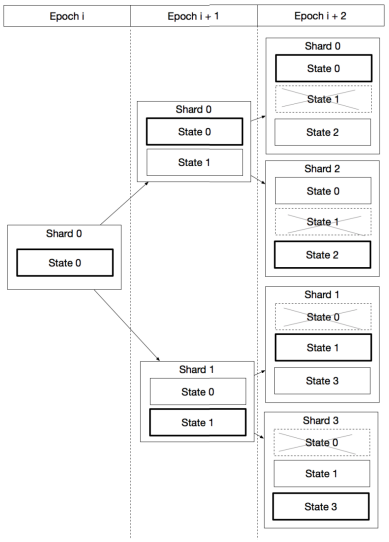


Figura 6: Redundància als *shards* entre èpoques. (Font: Captura de [2]).

2.6 Transaccions entre *shards*

Com a exemple de com s'executen les transaccions entre diferents *shards* i com es comuniquen amb la *metachain*, emprarem una **arquitectura amb només dos *shards***. Imaginem que un usuari des del seu *wallet* amb una adreça dins del *shard* 0 envia EGLD a l'adreça d'altre *wallet* que es troba en el *shard* 1 (Fig. 7). L'estructura dels blocs estarà formada per [2]:

- **Capçalera** (*header*): Conté la informació referent al bloc (*nonce*, *round*, *proposer*, *validators*, *timestamps*, etc.).

[21] Aquest paràgraf és un resum molt breu. Per referències més detallades consultar [2].

- **Llista de miniblocs**: Per a cada *shard* hi haurà un minibloc que contindrà transaccions a executar. S'ha de destacar que un minibloc és la unitat atòmica de processament, això significa que o es processen totes les transaccions del minibloc o no es processa cap (en aquest darrer cas, es posposarà l'execució al següent *round*). Dins d'un mateix bloc, poden aparèixer diversos miniblocs amb el mateix emissor i receptor (no hi ha cap limitació). Per exemple, en aquest cas simple amb només dos *shards*, un bloc en el *shard* 0 contindrà bàsicament tres tipus de miniblocs:
  - Minibloc 0: Conté les transaccions on les adreces del remitent i del destinatari són al *shard* 0.
  - Minibloc 1: Conté les transaccions on les adreces del remitent són al *shard* 0 i les del destinatari al *shard* 1.
  - Minibloc 2: Conté les transaccions on les adreces del remitent són al *shard* 1 i les del destinatari al *shard* 0.

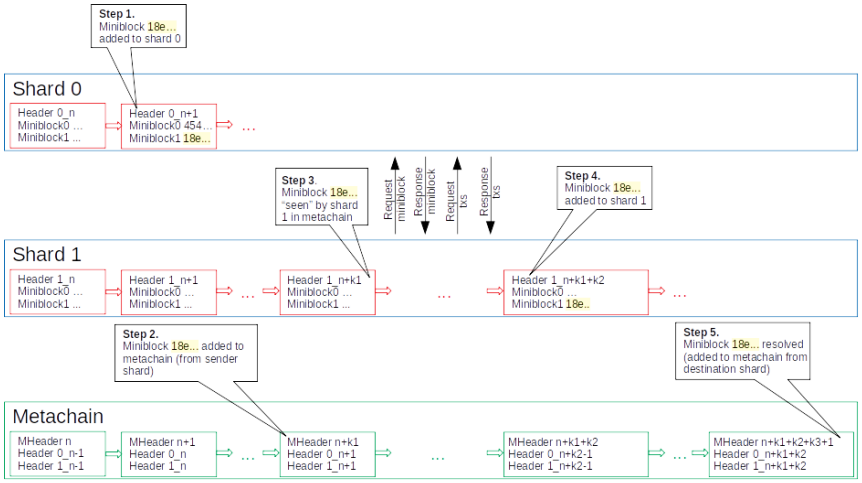


Figura 7: Exemple d'execució de transacció entre dos *shards*. (Font: Fig. 4 de [2]).

Observem en la **Fig. 7** un exemple de transacció asíncrona amb dos *shards* implicats. Després d'arribar la transacció al *shard* 0 (***step* 1**), la capçalera del bloc i els miniblocs s'envien a la *metachain*. Aquesta darrera dona fe[22] d'aquest bloc del *shard* 0 creant un nou bloc a la metacadena que conté la següent informació sobre tots els miniblocs: identificador del *shard*

[22] "Donar fe" o "autenticar" són les traduccions al català que he trobat més adients pel verb anglès "to notarize". El verb "notarizar" existeix en espanyol, però no existeix "notaritzar" en català. En qualsevol cas, empraré aquest darrer terme en algunes ocasions entre cometes per no generar confusions.

remitent, identificador del *shard* receptor i *hash* del minibloc (***step* 2**). El *shard* 1 veu el minibloc a la *metachain* i obté el seu *hash* del metabloc (***step* 3**), per després demanar el minibloc al *shard* 0 i executar-lo altra vegada (***step* 4**). Finalment, el *shard* 1 envia el resultat a la *metachain* que valida la transacció creuada (***step* 5**). En aquest punt la transacció ja es pot considerar finalitzada.

2.7 *La màquina virtual d'Elrond*

La màquina virtual d'Elrond executa WebAssembly (Wasm)[23], un llenguatge de baix nivell amb format binari compacte anomenat *bytecode* [23]. El fet que executi Wasm implica que es puguin escriure contractes intel·ligents en qualsevol llenguatge que sigui possible compilar cap a *bytecode* (C, C++, C#, Rust, Go, TypeScript, etc.). Tot i això, Elrond recomana als desenvolupadors que emprin Rust i els hi faciliten un *framework*[24] per a aquest llenguatge, així com un *plugin* per a l'IDE de Microsoft Visual Studio Code[25]. Entre les característiques a destacar de la màquina virtual d'Elrond [1][24]:

- **Sense estat**: Això significa que, quan un *smart contract* sigui executat, es guardarà la informació en una estructura de dades transitòria, en lloc d'escriure directament en l'estat. Quan acabi l'execució, si aquesta és exitosa, l'API aplicarà els canvis a l'emmagatzematge i/o a la cadena de blocs.

- **Execució fora de procés**: La màquina virtual serà executada en un procés independent i el node en si mateix en un altre, tot i que compartiran informació a través de canonades (*pipes*) anònimes a memòria principal. A més, el *bytecode* s'executarà en un entorn aïllat i la memòria del procés de la màquina virtual serà inaccessible.

- **Motor d'execució ràpida**: Com a motor d'execució s'empra una versió modificada per Elrond de Wasmer[26], una implementació de Wasm[27] de codi obert escrita en Rust per entorns servidor. Per les característiques de Wasmer, l'execució dels contractes intel·ligents es fa a una velocitat gairebé nadiua.

- **Trucades asíncrones entre contractes**: Els contractes intel·ligents poden executar trucades entre ells fent servir l'API de la màquina virtual. Com hem vist, la xarxa d'Elrond es fragmenta adaptativament,

[23] https://webassembly.org/

[24] https://github.com/ElrondNetwork/elrond-wasm-rs

[25] https://marketplace.visualstudio.com/items?itemName=Elrond.vscode-elrond-ide

[26] https://wasmer.io/

[27] Realitzada per l'empresa Wasmer Inc. que també ha creat un gestor de paquets que permet els desenvolupadors compartir mòduls empaquetats de codi Wasm.

el que pot fer que es truqui a un *smart contract* que es trobi en un altre *shard*. En aquest cas l'execució serà asíncrona. Si ambdós estan en el mateix fragment, l'execució serà síncrona. Tot això es fa de forma transparent per al desenvolupador.

En el cas general, els contractes intel·ligents seran compilats generant un arxiu WASM que serà desplegat en alguna de les xarxes d'Elrond.

2.8 Execució de contractes intel·ligents en l'arquitectura amb *shards*

Un important punt a destacar és l'execució dels *smart contracts* en aquesta arquitectura fragmentada. Elrond arriba a una solució amb execució asíncrona entre els *shards* [2].

El procés comença quan l'usuari crea una transacció per executar un contracte intel·ligent. Si el contracte intel·ligent no és ubicat en el mateix *shard*, el cost de la transacció es lleva del compte del remitent i s'afegeix a un minibloc –segons correspongui a l'adreça del receptor– del seu *shard*. La transacció és "notaritzada" per la *metachain* i després processada pel *shard* de destí. Al fragment de destí, la transacció es tracta com una invocació del mètode del contracte intel·ligent, ja que és on es troba (l'adreça de destí és la del *smart contract*). Per a la trucada del contracte intel·ligent, es crea un compte temporal que suplanta el compte del remitent, amb el saldo del valor de la transacció i es crida el contracte intel·ligent. Després de l'execució, el contracte intel·ligent pot retornar resultats que afecten diversos comptes en diferents *shards*. Els resultats que afecten els comptes del *shard* del contracte intel·ligent s'executen a la mateixa ronda, en cas contrari es crearan transaccions *Smart Contract Result* (SCR). En aquest darrer cas, es creen miniblocs SCR per a cada *shard* de destí que posteriorment són "notaritzats" per la *metachain* (de la mateixa manera que hem vist a la secció 2.6 per a les transaccions entre *shards*). Finalment, són processats pels *shards* respectius (on resideixen els comptes de destí). En el cas que un contracte intel·ligent truqui dinàmicament a un altre que es trobi en un *shard* diferent, aquesta trucada es desaria com a resultat intermedi i es tractaria de la mateixa manera que per als comptes. Aquesta solució necessitarà **almenys 5 rondes per completar-se**, però té els avantatges de què **no es necessita bloqueig ni moure estats entre *shards***.

2.9 Muntar un node validador

Com ja s'ha deixat entreveure, la **xarxa d'Elrond** és composta pels seus nodes i per la interconnectivitat entre ells. Com s'ha dit en la secció 2.2, per **node** s'entendrà qualsevol instància del programari de codi obert desenvolupat per Elrond per a aquesta tasca [28]. A més a més, qualsevol persona o

[28] Disponible a: https://github.com/ElrondNetwork/elrond-go.

entitat responsable de la gestió d'un o més d'aquests nodes es coneix com "**operador de nodes**". Com també s'ha vist, la xarxa és dissenyada per ser segura i per poder balancejar la seva càrrega. Així, quan un nou node s'uneix a la xarxa aporta més seguretat i eficiència. I la xarxa premiarà els nodes per la seva aportació, creant-se així una espècie de simbiosi.

Elrond és una xarxa descentralitzada de *blockchain*, és a dir, nodes de procedència desconeguda s'uneixen per crear seqüencialment blocs amb una cadència determinada. Els blocs contenen operacions que es realitzaren a petició dels usuaris de la xarxa pagant unes taxes (*fees*) per l'execució d'aquestes. Entre les operacions es troben la transferència de *tokens* entre comptes o l'execució de contractes intel·ligents (totes les operacions prenen la forma de transaccions). Per tant, es defineix així un llibre de comptabilitat distribuït que no depèn de cap entitat central.

Els nodes que estan autoritzats a prendre part en l'algorisme de consens s'anomenen **validadors**. Com que són els que veritablement produeixen i validen blocs, són els únics nodes recompensats amb EGLD per la seva contribució. Per assegurar el bon comportament dels validadors han de tenir bloquejats 2500 EGLD (*stake*). Com ja vàrem dir en la secció 2.2, els nodes sense *stake* s'anomenen **observadors** (no participen en el consens i no guanyen recompenses). Si el validador no funciona correctament (desconnexions de la xarxa en mig del procés de consens, accions malicioses, etc.) seran "multats" perdent EGLD (*stake slashing*) o fins i tot, en casos greus, se'ls hi llevarà l'estatus de validador (el node passarà a l'estat "*jailed*" i no es triarà per al consens si no paga una multa de 2,5 EGLD). Si això succeeix també davallarà la seva qualificació (*rating score*). Com que aquesta puntuació defineix la fiabilitat del validador, és considerada per l'algorisme de selecció de nodes per triar el grup de consens. Així, s'afavoreix l'elecció de nodes amb qualificacions altes, també se'ls hi premiarà amb més EGLD.

Els requeriments mínims del sistema són:

- 4 CPUs dedicades (Intel/AMD amb *flags* SSE4.1 i SSE4.2 activats).
- 8 GB de RAM.
- Disc dur SSD amb 200 GB d'espai lliure.
- Connexió a Internet de 100 Mbit/s sempre activa (mínim 4 TB/mes).
- Sistema operatiu Linux[29]/MacOS.

Com podem observar, són característiques d'un PC/Mac convencional en l'actualitat (i no d'ordinadors amb potents targetes gràfiques dedicades com passa amb el PoW de Bitcoin). No és l'objectiu del present treball, però

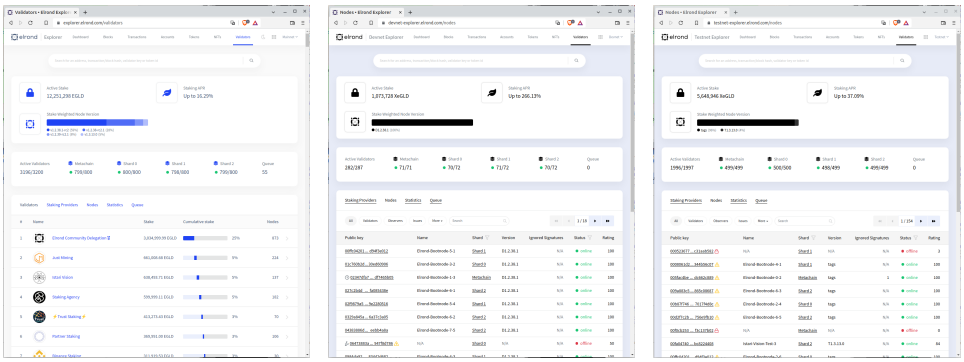
[29] Es recomana Ubuntu 20.04.

guies com-es-fa[30] per instal·lar el programari per muntar un node validador en la xarxa Elrond es poden trobar a [1] o [25].[31]

2.10 Xarxes Elrond disponibles

En Elrond tenim tres xarxes disponibles:

- ***Mainnet***: És la xarxa en producció d'Elrond (transaccions reals amb el pertinent cost econòmic). En el moment de redactar la present secció, té més de 3000 nodes validadors distribuïts en 3 *shards* i una *metachain*.
- ***Devnet***: És una xarxa pública de proves mantinguda per la comunitat d'Elrond on qualsevol desenvolupador pot provar els seus contractes intel·ligents i dApps en un entorn real. Té uns 300 nodes validadors distribuïts en 3 *shards* i una *metachain*. En el present treball empraré aquesta xarxa, en detriment de *testnet*, per tenir més estabilitat.
- ***Testnet***: També pública i mantinguda per la comunitat d'Elrond, però s'empra per fer proves de futures millores i rendiment [26] (això implica que el *blockchain* es pot reiniciar[32]). Té uns 2000 nodes repartits en 3 *shards* i un *metachain*.



(a) *Mainnet*: https://explorer.elrond.com/validators (b) *Devnet*: https://devnet-explorer.elrond.com/nodes (c) *Testnet*: https://testnet-explorer.elrond.com/nodes

Figura 8: Exploradors de les xarxes disponibles a Elrond.

[30] Traducció del Termcat del terme "*howto*".

[31] Aquesta secció torna a ser una síntesi de l'apartat "*Validarors*" de [1].

[32] En el grup de Telegram "*Elrond Validators Announcements*" (https://t.me/ElrondValidatorsAnn) es va definir una ronda de gènisi per la *devnet* (reinici) per a dia 29 de març de 2022.

2.11 Carteres

Com ja s'ha comentat en la secció 1.1 existeixen diferents carteres que es poden fer servir per enviar, rebre i emmagatzemar EGLD de forma segura.

- **Moneder web**. Existeix una versió per a cadascuna de les xarxes:[33]
  - *Mainnet*: Accesible a https://wallet.elrond.com/.
  - *Devnet*: Accesible a https://devnet-wallet.elrond.com/.
  - *Testnet*: Accesible a https://testnet-wallet.elrond.com/.

  Tampoc és l'objectiu del present treball explicar com crear un moneder, però una guia com-es-fa es pot trobar en [1]. Quan es crea s'haurà d'introduir una contrasenya[34] i se'ns donaran 24 paraules secretes (que permeten generar la clau privada). El procés finalitzarà descarregant un fitxer json (*keystore file*). Com es veu en la Fig. 9, podem accedir al nostre moneder amb aquest arxiu json i introduint la contrasenya que hem creat (1 a Fig. 9). Una altra opció és generar un fitxer "PEM" (*Privacy Enhanced Mail*) amb la nostra clau privada i emprar-lo per accedir (nombre 3 a Fig. 9), que es pot generar emprant la utilitat d'Elrond "erdpy" (s'hauran d'introduir les 24 paraules secretes separades per un espai):

  ```
  erdpy --verbose wallet derive testdev-wallet.pem --mnemonic
  ```

  Finalment, es pot iniciar sessió amb un moneder de maquinari (per exemple amb un Ledger Nano S) o llegint un codi QR amb l'aplicació mòbil Maiar (nombres 2 i 4 a Fig. 9, respectivament).

- **Maiar DeFi Wallet**. Es tracta d'una extensió per a navegadors de la família de Chrome[35]. El funcionament i prestacions és similar al moneder web (i, fent la comparació, a l'extensió "Metamask"[36] de la xarxa Ethereum). En la Fig. 10b es pot observar que també es pot canviar entre les tres xarxes disponibles.

- ***Maiar***[37]. DApp que implementa una cartera digital per dispositius iOS i Android. Permet enviar o rebre diners emprant el número de telèfon mòbil o un *herotag* (una espècie de nom d'usuari que es pot emprar en lloc de la teva adreça). És completament descentralitzada, sense custòdia i Elrond no té accés als fons de l'usuari en cap moment. Es pot treballar amb criptoactius com Elrond Gold (EGLD),

[33] Per a la *devnet* i la *testnet* es disposa d'una opció *Faucet* per obtenir xEGLD (que no té cap cost en moneda *fiat* i et permetrà poder relalitzar transaccions).
[34] Ha de tenir almenys nou caràcters, una lletra majúscula, un símbol i un nombre.
[35] https://chrome.google.com/webstore/detail/maiar-defi-wallet/
[36] https://metamask.io/
[37] https://maiar.com/

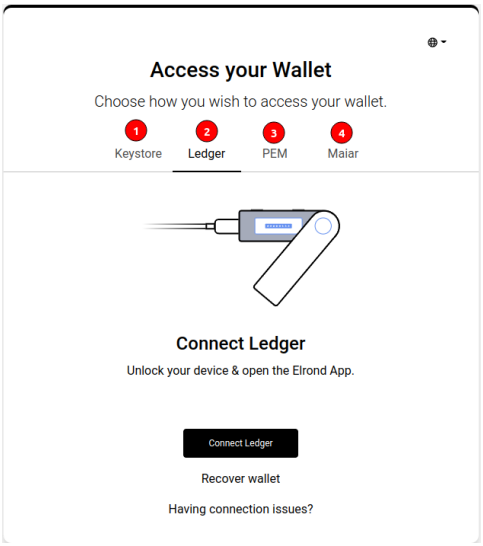
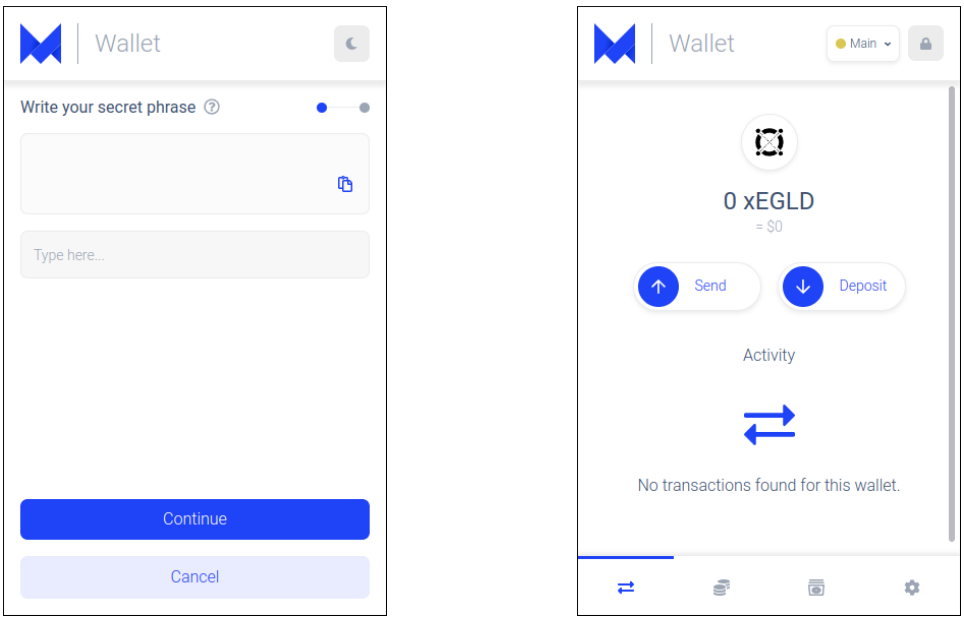


Figura 9: Mètodes d'accés al moneder web.



(a) (b)

Figura 10: Maiar DeFi Wallet.

Binance (BNB), Ethereum (ETH) o Bitcoin (BTC). Com no pot ser d'altra forma, es disposen de mecanismes de recuperació (un frase) per rescatar els fons en cas de pèrdua o robatori del nostre telèfon intel·ligent. No he trobat forma de connectar a la *devnet* o *testnet* des de Maiar, penso que déu ser així perquè és principalment dissenyada

per treballar amb la *mainnet*. Òbviament, aquesta dApp és pensada per comprar criptoactius a través de pagaments amb targeta bancària o transferències SEPA (s'han d'emprar passarel·les en ambdós casos). Curiosament, no permet fer *swaps* entre criptoactius (només es poden fer des de Maiar Exchange).

Figura 11: Maiar per iPhone.

- ***Webhooks***. Són enllaços que criden[38] a la cartera de l'usuari perquè iniciï sessió o ompli un formulari per realitzar una transacció (del tipus pagament) amb els arguments proporcionats. Un cop realitzada l'acció, l'usuari és redirigit a un URL amb informació d'estat d'èxit o d'error. Un exemple seria [1]:

  `https://wallet.elrond.com/hook/login?callbackUrl=https://example.com/`

- ***Ledger***. Ja s'ha comentat al punt "Moneder web", i a més és recomanat per Elrond si es treballa amb grans quantitats d'EGLD, també podem emmagatzemar els nostres criptoactius en moneders de maquinari com són els dispositius *Ledger Nano S* o *Ledger Nano X*.

[38] En català balear fem ús del verb "cridar" en lloc del verb "trucar". Així diem per exemple "cridar per telèfon" o "cridar el mètode d'un objecte".

2.12 Maiar Exchange

Maiar Exchange[39] és un DEX (*Decentralized EXchange*) basat en *liquidity pools* –fons de liquiditat– creat per Elrond emprant la seva arquitectura escalable. Permet els usuaris efectuar la compravenda de criptoactius (*trading*). A més a més, els usuaris poden convertir-se en proveïdors de liquiditat aportant els seus actius (guanyant *tokens* MEX per cada transacció d'intercanvi realitzada segons el parell de *tokens* aportats com a reserva). MEX és el *token* nadiu de Maiar Exhange. A part de tenir la funció d'incentivar els usuaris perquè proporcionin liquiditat, també s'emprarà en processos de governança. Així, ens trobem davant d'un Exchange de *criptoactius* sense intermediaris, cosa que abarateix els costos de les transaccions i dels *swaps*.

Entre el que podem fer amb Maiar Exchange vull destacar [27][28]:

- ***Swapping*** (intercanvi): Consisteix a intercanviar un criptoactiu per un altre. El mecanisme d'intercanvi consisteix en permutar els actius de l'usuari amb els *liquidity pools* (fons de liquiditat), i no directament amb altres participants en el mercat. Els preus dels actius es fixen a partir d'una fórmula matemàtica del que s'anomena *Automated Market Making*. En lloc d'utilitzar un llibre de comptabilitat com en un intercanvi tradicional, els actius tenen uns preus calculats segons un algorisme. Elrond empra la fórmula $x * y = k$, on x és la quantitat d'un token al *liquidity pool* i y és la quantitat de l'altre. k és una constant fixa, el que significa que la liquiditat total del grup sempre romandrà en la mateixa proporció. Hi ha una comissió del 0,3% per intercanviar tokens (el 0,25% pels proveïdors de liquiditat proporcional a la seva contribució a les reserves de liquiditat i amb la resta es compraran MEX que es cremaran). En la data de redacció d'aquesta secció, els actius entre què es poden fer intercanvis són MEX, USDC, RIDE, CRU, ZPAY, ISET, AERO, EFFORT i WAM contra EGLD. Això significa que hi ha fons de liquiditat MEX-EGLD, EGLD-USDC, etc.

- ***Liquidity Pools***: Són reserves de tokens que es troben en contractes intel·ligents del DEX i estan disponibles perquè els usuaris facin intercanvis. Aquests tokens els proporcionen proveïdors de fons de liquiditat amb els quals els usuaris de la plataforma poden negociar. Els proveïdors són incentivats amb recompenses (tokens MEX) pels intercanvis que es produeixen en els seus *pools* (segons s'ha explicat en el punt anterior). S'ha de destacar que els proveïdors de liquiditat en qualsevol moment poden retirar els fons que han dipositat (independentment de les condicions del mercat).

[39] https://maiar.exchange/

- ***Farming***[40]: Consisteix en el fet que els usuaris bloquegin la seva liquiditat per obtenir recompenses. Si posem els nostres actius en un *liquidity pool* obtindrem les recompenses descrites en els apartats anteriors (això és el *farming* i és el que s'ha descrit en el punt d'adalt). Però hi ha també l'opció de posar els nostres *liquidity pools* en *staking*, el que significa que no els podrem retirar durant un temps definit a canvi d'obtenir una recompensa major.

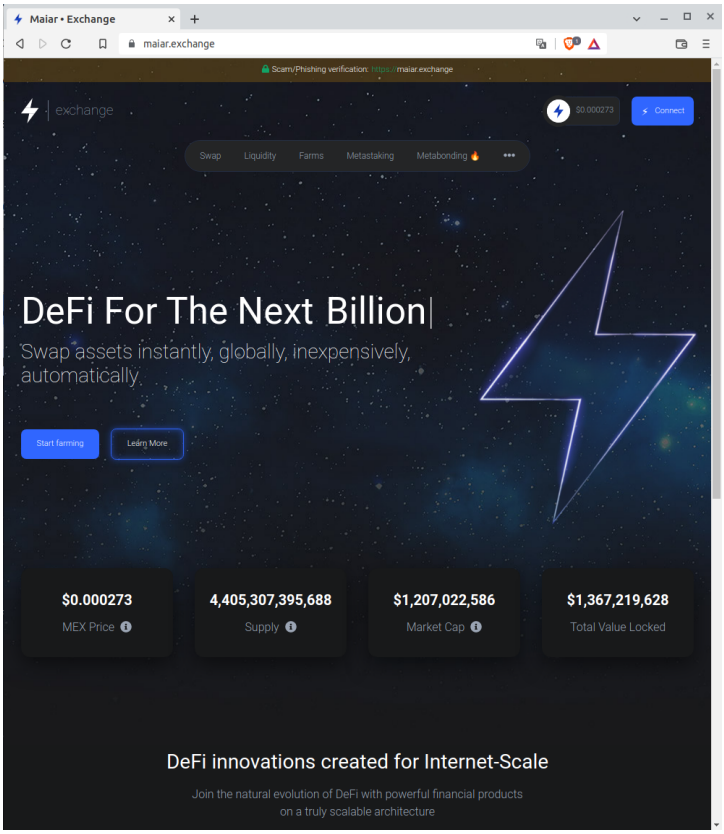


Figura 12: Maiar Exchange.

2.13 Altres punts destacables

- **Camps d'una transacció**:

  Els camps d'una transacció sense signar en la xarxa Elrond són [1]:

  - **nonce** (*number*): Nombre de seqüència del compte. És obligatori.

[40] No m'he atrevit a traduir aquest terme, però un mot que m'agrada del català és "conrear".

- **value** (*string*): El valor a transferir. És obligatori (pot ser 0).
- **receiver** (*string*): Adreça del receptor (format *bech32*). És obligatori.
- **sender** (*string*): Adreça del remitent (format *bech32*). És obligatori.
- **gasPrice** (*number*): El preu del *gas* que s'utilitzarà en l'abast de la transacció. És obligatori.
- **gasLimit** (*number*): El nombre màxim d'unitats de *gas* assignades per a la transacció. És obligatori.
- **data** (*string*): Informació arbitrària sobre la transacció, codificada en base64 . No és obligatori.
- **chainID** (*string*): Identificador de cadena. 'D' per a la xarxa *devnet*, 'T' per a *textnet* i '1' per a la *mainnet*. És obligatori.
- **version** (*number*): La versió de la transacció (p. ex 1). És obligatori.

En les transaccions que han de ser signades, amb la clau pública del remitent (amb l'algorisme Ed2519), s'ha d'afegir el següent camp:

- **signature** (*string*): Signatura digital que consta de 128 caràcters hexadecimals (64 bytes en representació en brut).

Hi ha disponibles diferents eines per realitzar la signatura des de la *shell* de Linux i des del codi font: erdpy, erdwalletjs-cli o elrond-core-js.

- **Decimals per treballar amb EGLD**: Un EGLD és configurat per tenir 18 decimals. Així, per exemple 2,5 EGLD tendrien un valor de $2,5 * 10^{18} = 2500000000000000000$. L'usuari quan creï ESDTs podrà definir el seu nombre de decimals (amb un màxim de 18).

- **Costos de processament** [1]: Una transacció en la xarxa Elrond té un cost de processament (*processing cost*), que s'expressa com una quantitat d'unitats de *gas*. Quan es llança una transacció s'ha de proporcionar un *gasLimit*, que és el cost màxim que estem disposats a assumir. El **consum real de gas** (*actual gas consumption* o també *used gas*) és la quantitat real que s'ha consumit del *gasLimit* requerida perquè la xarxa processi la transacció. Per calcular aquest darrer, la xarxa el divideix en dos components d'utilització del *gas* [1]:

  - **Moviment de valors i tractament de dades**: Això seria per exemple una transferència d'EGLD entre adreces. El cost es calcula amb la fórmula:

```
tx.gasLimit =
    networkConfig.erd_min_gas_limit +
    networkConfig.erd_gas_per_data_byte * lengthOf(tx.data)
Nota:
networkConfig.erd_min_gas_limit <= tx.gasLimit i
tx.gasLimit <= networkConfig.erd_max_gas_per_transaction
```

  - **Execució de contracte intel·ligent**: Una trucada a un *smart contract* requeriria aquest i l'anterior. El cost és més difícil de determinar, ja que depèn del codi font concret del contracte intel·ligent. Es solen emprar simulacions i estimacions.

Finalment, la **tarifa de processament** (*processing fee*) es calcula amb respecte de l'*actual gas consumption* i dels seus dos components. Per moviment de valors i tractament de dades s'especifica en la transacció un *gas price per gas unit* (que ha de ser igual o superior al paràmetre de la xarxa *erdmin_gas_price*). Per execució de contracte intel·ligent el *gas price per gas unit* es calcula respecte a un altre paràmetre de xarxa anomenat $erd\_gas\_price\_modifier$:

```
value_movement_and_data_handling_price_per_unit = tx.GasPrice
contract_execution_price_per_unit = tx.GasPrice * networkConfig.erd_gas_price_modifier
```

- **ESDT (*Elrond Standard Digital Token*)**: La xarxa Elrond admet de forma nadiua l'emissió de *tokens* creats pels usuaris (des de codi, des d'un contracte intel·ligent o des de la *web wallet*). Això implica que no és necessari emprar contractes com els de tipus ERC20 de la xarxa Ethereum. Ja que el suport és nadiu, no es requereix processament extra per part de la màquina virtual (amb el que són tan eficients com el mateix EGLD).
- **NFT (*Non-Fungible Token*) i SFT (*Semi-Fungible Token*)**: Suport nadiu d'NFT i SFT afegint metadades i atributs a sobre de l'ESDT. Així, són bastant similars a aquests darrers però amb atributs extra (*NFT Name*, *Quantity*, *Royalties*, *Hash*, *Attributes* i *URI*).
- ***Local Testnet***: Es pot configurar una *testnet* en local per fer proves i *debugging* del codi. Es pot configurar amb l'eina *erdpy* i conté nodes validadors, nodes observadors, un *seed node* i un *Elrond Proxy*.
- **API REST**: Té dues capes a les quals es pot accedir públicament:
  - **https://gateway.elrond.com**: La de més baix nivell. Gestiona l'encaminament de les peticions de forma transparent segons els mecanismes de fragmentació que s'han descrit en els apartats anteriors.
  - **https://api.elrond.com**: La de més alt nivell (empra els serveis de la capa anterior). Aporta serveis com un mecanisme de memòria cau, cerques històriques amb Elasticsearch[41], etc.

[41] https://www.elastic.co/

3 testDEX

3.1 Anàlisi

3.1.1 DEX i AMM

Un ***Decentralized exchange*** **(DEX)** és un mercat on els negociants de criptomonedes fan transaccions directament entre ells –d'igual a igual (*peer-to-peer*)– sense haver de lliurar la gestió dels seus fons a un intermediari [3]. És a dir, són dissenyats per eliminar qualsevol autoritat de supervisió en els intercanvis (*swaps*) de criptoactius, evitant que s'hagin d'enviar dades de caràcter personal (noms, adreces, etc.). Per executar les ordres d'intercanvi sense intermediaris s'empren contractes intel·ligents. Això xoca amb el concepte de ***Centralized exchange*** on la gestió és responsabilitat d'una organització, com ara un banc o qualsevol altra corporació, que requerirà que els seus usuaris estiguin identificats i que custodiaran els actius dels usuaris (òbviament cercant un benefici econòmic).

En el present treball ens centrarem en una versió simplificada de DEX fent servir *Automated Market Makers* (AMM). Els diferents tipus de DEX es poden observar en la figura 13.

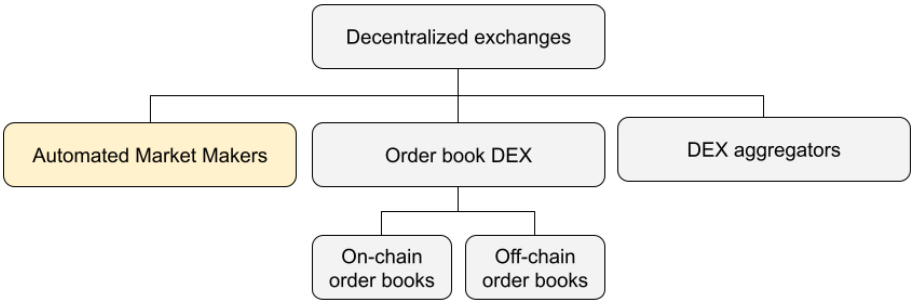


Figura 13: Tipus de DEX. (Font: [3]).

En [29] es descriu que una de les formes de prescindir del llibre d'ordres d'un *Exchange* tradicional és emprant AMM. Aquests protocols fan servir una fórmula matemàtica per calcular el preu de l'actiu i un fons de liquiditat per a cada parell de criptoactius (tal qual hem vist en la secció 2.12). Quan es vulgui fer un intercanvi entre criptoactius s'interactuarà directament amb un contracte intel·ligent que determinarà el preu de compra de l'actiu destí segons l'estat del fons de liquiditat. La fórmula per calcular això, també vista en la secció 2.12, és $x * y = k$ (amb x la quantitat d'un token al fons de liquiditat, y és la quantitat de l'altre i k és una constant fixa per mantenir la proporcionalitat entre el parell de *tokens*). Així, imaginem que tenim un fons de liquiditat amb el parell xEGLD-UOC –"UOC" és un ESDT creat per mi– amb 10 xEGLD i 100000000 UOC. Tenint que $x * y = k$ i, pel fons descrit, la fórmula quedarà com a $10 * 100000000 = 1000000000$. Si volem

calcular la quantitat $q_{withoutFee}$ d'UOC que obtindríem per 1 xEGLD en aquest fons:

$$(10+1)*(100000000-q_{withoutFee})=1000000000$$

$$1100000000-11*q_{withoutFee}=1000000000$$

$$1100000000-1000000000=11*q_{withoutFee}$$

$$100000000=11*q_{withoutFee}$$

$$q_{withoutFee}=100000000/11$$

$$q_{withoutFee}=9090909,090909091$$

A partir del resultat anterior, podem calcular el **la quantitat d'UOC que obtindrem amb 1 EGLD**:

$$price_{EGLD-UOC}=q_{withoutFee}/q_{EGLD}$$

$$price_{EGLD-UOC}=9090909,090909091/1=9090909,090909091$$

Ara tenim que el *liquidity pool* ha quedat amb el parell xEGLD-UOC com $11-90909090,909090909$ ($90909090,909090909=100000000-9090909,090909091$). Amb això tenim que $11*90909090,909090909=1000000000$ (es manté la constant k)[42]. El **preu pagat per una unitat d'UOC en EGLD** haurà estat:

$$price_{UOC-EGLD}=1/9090909,090909091=0,00000011$$

En el procediment anterior no hem aplicat cap comissió, i com s'ha dit els *liquity providers* aportaran als fons de liquiditat parells d'actius obtenint una recompensa segons els *swaps* que es realitzin amb els seus parells. Recalculant la fórmula d'abans aplicant una comissió fee_{EGLD}:

$$(10+(1-fee_{EGLD}))*(100000000-q_{withFee})=1000000000$$

$$11-10*fee_{EGLD})*(100000000-q_{withFee})=1000000000$$

$$11*(100000000-q_{withFee})-10*fee_{EGLD}*(100000000-q_{withFee})=1000000000$$

$$1100000000-11*q_{withFee}-1000000000*fee_{EGLD}+10*fee_{EGLD}*q_{withFee}=1000000000$$

$$-11*q_{withFee}+10*fee_{EGLD}*q_{withFee}=1000000000-1100000000+1000000000*fee_{EGLD}$$

$$q_{withFee}*(-11+10*fee_{EGLD})=-100000000+1000000000*fee_{EGLD}$$

$$q_{withFee}=(-100000000+1000000000*fee_{EGLD})/(-11+10*fee_{EGLD})$$

[42] Aquesta operació deixa intuir que per mantenir el valor de k constant serà un problema, ja que l'arrodoniment provocarà que s'acumuli error.

Aplicant, per exemple, una comissió del 0,3%:

$$q_{withFee} = (-100000000 + 1000000000 * 0,003)/(-11 + 10 * 0,003)$$

$$q_{withFee} = -97000000/ - 10,97$$

$$q_{withFee} = 8842297,174111212$$

I el **preu en EGLD per unitat d'UOC** haurà estat (augmenta un poc per la comissió):

$$price_{UOC-EGLD} = 1/8842297,174111212 = 0,000000113$$

També podem calcular a partir de la fórmula anterior la **comissió pagada en EGLD** (fee_{EGLD}):

$$q_{withFee} * (-11 + 10 * fee_{EGLD}) = -100000000 + 1000000000 * fee_{EGLD}$$

$$-11 * q_{withFee} + 11 * 10 * fee_{EGLD} = -100000000 + 1000000000 * fee_{EGLD}$$

$$11 * 10 * fee_{EGLD} - 1000000000 * fee_{EGLD} = -100000000 + 11 * q_{withFee}$$

$$fee_{EGLD} * (11 * 10 - 1000000000) = -100000000 + 11 * q_{withFee}$$

$$fee_{EGLD} = (-100000000 + 11 * q_{withFee})/(11 * 10 - 1000000000)$$

Com que ja hem calculat $q_{withFee}$:

$$fee_{EGLD} = (-100000000 + 11 * 8842297,174111212)/(11 * 10 - 1000000000)$$

$$fee_{EGLD} = -2734731,084776668/ - 999999890$$

$$fee_{EGLD} = 0,002734731$$

Finalment, podem calcular la **comissió en UOC** (fee_{UOC}) fent la substracció de quantitat d'UOC que es dona sense i amb comissió:

$$fee_{UOC} = q_{withoutFee} - q_{withFee}$$

$$fee_{UOC} = 9090909,090909091 - 8842297,174111212$$

$$fee_{UOC} = 248611,916797879$$

Si tornem a comprovar la constant k, tenim que el *liquidity pool* ha quedat amb el parell xEGLD-UOC com 10+1 i $100000000-8842297,1741112121-248611,916797879$ (s'ha de pensar que la comissió s'ha de llevar del pool perquè se li envia al proveïdor de liquiditat). És a dir, 11 i $90909090,909090909$. Amb això tenim que $11 * 90909090,909090909 = 1000000000$. Així, l'usuari haurà pagat 1 xEGLD per $9090909,090909091$ UOC i el fons de liquiditat mantindrà la mateixa constant k. El **preu pagat per una unitat de UOC en EGLD** haurà estat:

$$price_{UOC-EGLD} = 1/9090909,090909091 = 0,00000011$$

En resum, amb aquest exemple s'ha mostrat que amb 1 xEGLD l'usuari ha pagat 0,002734731 xEGLD de comissió i ha rebut 8842297,174111212 UOC. La citada comissió serà tramesa al moneder del proveïdor de liquiditat automàticament.

A tot això se li ha d'afegir una cosa més: **la comissió que pagarà l'usuari a la xarxa Elrond**. En la secció 2.13 s'ha vist que emprar la xarxa Elrond duu associat uns costos pel moviment de valors i tractament de dades (transferim una quantitat de tokens al contracte intel·ligent) i per execució de contracte intel·ligent (el *smart contract* que implementa l'AMM). Això encarirà l'operació a l'usuari i les operacions que es realitzen dins aquest contracte influiran en el cost. En temps de disseny s'haurà de triar una opció òptima a nivell de repercusió en els costos de l'usuari.

3.1.2 Les variants de DEX i AMM a testDEX

Per assolir els objectius del projecte no fa falta arribar a desenvolupar un DEX de l'estil de Maiar Exchange, Uniswap[43] o PancakeSwap[44]. Òbviament, un projecte d'aquesta envergagura seria poc realista per una limitació clara de temps. Per exemple, el protocol AMM no és exempt de problemes que s'haurien de tractar [30]. Així en aquesta versió inicial faré una sèrie d'adaptacions per ajustar-me al temps disponible:

- El propietari del contracte intel·ligent serà l'únic que pot definir *liquidity pools*.
- El propietari del contracte serà, per tant, l'únic que rebi les comissions pels *swaps*.
- Tots els parells tindran com a un dels seus components EGLD.

3.1.3 Requeriments funcionals

Els requeriments funcionals responen a la pregunta "què ha de fer un sistema". Pel present projecte s'especifiquen els següents:

- RF 1: El sistema permetrà el propietari del contracte intel·ligent dipositar un parell EGLD-ESDT en el fons de liquiditat des del seu *wallet*.
- RF 2: El sistema permetrà el propietari del contracte intel·ligent recuperar un parell EGLD-ESDT del fons de liquiditat cap al seu *wallet*, que prèviament haurà dipositat.
- RF 3: El sistema permetrà el propietari del contracte definir una taxa ("*fee*") que serà gravada als usuaris quan intercanviïn *tokens*.

[43] https://uniswap.org/
[44] https://pancakeswap.finance/

- RF 4: El sistema permetrà el propietari recuperar els guanys acumulats al seu contracte intel·ligent resultant del pagament de l'anterior taxa cap al seu *wallet*.
- RF 5: El sistema permetrà els usuaris intercanviar (*swap*) una quantitat d'un *token* ESDT de la cartera de l'usuari per un altre del fons de liquiditat (si existeix el parell adient).
- RF 6: El sistema permetrà els usuaris conèixer els parells donats d'alta al DEX.
- RF 7: El sistema permetrà els usuaris consultar si un parell és disponible per fer *swaps* (cap dels dos components pot tenir valor 0).
- RF 8: El sistema permetrà els usuaris consultar la constant K d'un parell en un moment donat.
- RF 9: El sistema mostrarà els preus dels actius en el moment actual (tant de compra com de venda).
- RF 10: El sistema enregistrarà el resultat de les transaccions per realitzar estadístiques.

3.1.4 Requeriments no funcionals

Els requeriments no funcionals responen a la pregunta "com ha de fer un sistema". Pel present projecte s'especifiquen els següents:

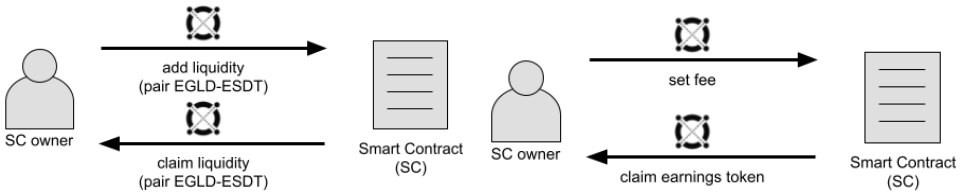
- RNF 1: La informació emprada en el sistema viatjarà per la xarxa emprant protocols segurs.
- RNF 2: El sistema serà accessible des de qualsevol ubicació a través d'Internet.
- RNF 3: Els usuaris del sistema el podran emprar utilitzant qualsevol moneder vàlid de la xarxa Elrond.

3.2 Disseny

3.2.1 Casos d'ús

Tindrem un usuari privilegiat (el propietari del contracte intel·ligent) i usuaris genèrics. El primer, òbviament, podrà realitzar també les operacions dels segons. El sistema permetrà només el propietari interactuar amb el contracte intel·ligent de la següent forma:[45]

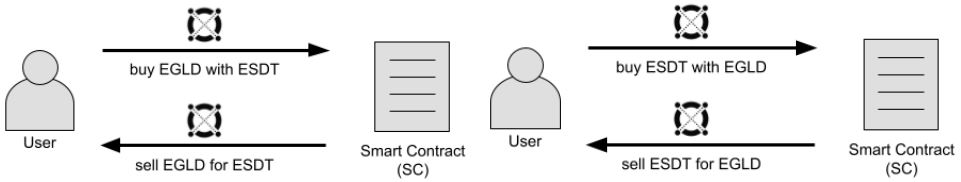
[45]He imitat els diagrames explicatius de la documentació d'Elrond [1] i per això no he fet servir diagrames de casos d'ús d'UML. El contingut de les figures es troba en anglès ja que és l'idioma que he emprat en el codi font.



(a) Requerimens funcionals 1 i 2. (b) Requeriments funcionals 3 i 4.

Figura 14: Interaccions del propietari amb el contracte intel·ligent.

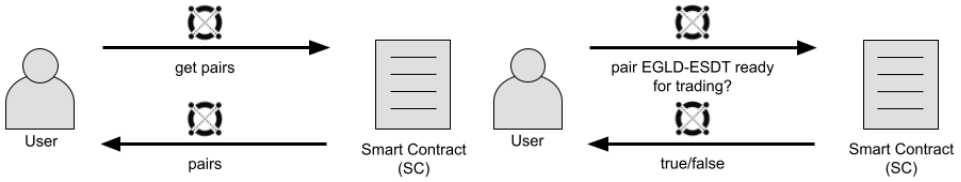
Amb les interaccions anteriors es podran **definir *liquity pools*** i una **taxa (*fee*)** que es cobrarà per les operacions de *swap*. Allò següent a definir seran els intercanvis en si mateixos. El criptoactiu de referència serà EGLD (tots els parells el tindran). Així es podrà passar una determinada quantitat d'EGLD a una quantitat d'ESDT, això també es pot anomenar "**comprar ESDT amb EGLD**". A la inversa, podrem passar ESDT a EGLD, que dit d'altra forma seria "**vendre ESDT per EGLD**". La següent figura ho mostra:



(a) RF 5 (EGLD-ESDT). (b) RF 5 (ESDT-EGLD).

Figura 15: Operacions de compra i venda de *tokens* (*swaps*).

Per altra banda, serà necessari que el sistema mostri als usuaris els **parells disponibles per fer intercanvis i si es pot operar amb ells en un moment donat** (requeriments funcionals 6 i 7).



(a) RF 6. (b) RF 7.

Figura 16: Parells donats d'alta preparats per a intercanvis.

A més, el sistema mostrarà la **constant k del protocol AMM** (que s'ha de mantenir constant) i el **preu dels actius en el moment actual**. Internament es treballa amb nombres sencers on es destinen les 18 posicions

a la dreta per decimals, però realment són nombres sencers. Això farà que la constant k pugui fluctuar i que estigui sotmesa a correccions per mantenir-la en el seu valor inicial. Així mateix, els usuaris hauran de conèixer en temps real els preus de les parelles de tokens. Els casos d'ús sobre aquests punts es mostren en la Fig. 17.

(a) RF 8. (b) RF 9 (EGLD-ESDT). (c) RF 9 (ESDT-EGLD).

Figura 17: Valor actual de la constant k i preus de compra-venda dels parells de *tokens*.

El darrer requeriment funcional és un poc especial i fa referència a què **el sistema tractarà el resultat de les transaccions** per generar estadístiques per intentar comprovar les característiques de la xarxa Elrond.

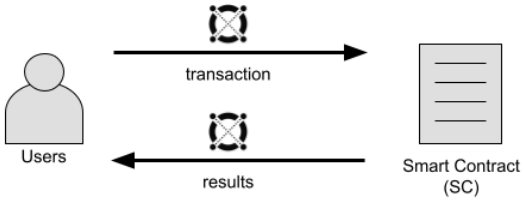


Figura 18: Requeriment funcional 10.

Finalment, pel que fa als **requeriments no funcionals**, no cal afegir res ja que formen part de les característiques intrínseques de les tecnologies d'Internet i de la xarxa Elrond.

3.2.2 *Smart contract*

El contracte intel·ligent serà públic a la xarxa *devnet* d'Elrond. S'ha de remarcar que totes les dades dels *smart contracts* són disponibles públicament, tot i que pot ser complicat cercar manualment l'emmagatzematge del contracte i per això es solen definir *getters* públics[46]. Quan es desplegui a la xarxa se li assignarà automàticament una adreça amb el format que s'ha descrit en seccions anteriors.

En la Fig. 19 es mostren les propietats i mètodes del contracte intel·ligent que emprarem (anomenat "TestDEX"). Cal dir, que en argot específic de *blockchain* seria més apropiat anomenar a una propietat "emmagatzematge".

[46]Quan s'implementin les propietats i mètodes públics en el lleguatge Rust s'anotaran amb #[endpoint] o #[view].

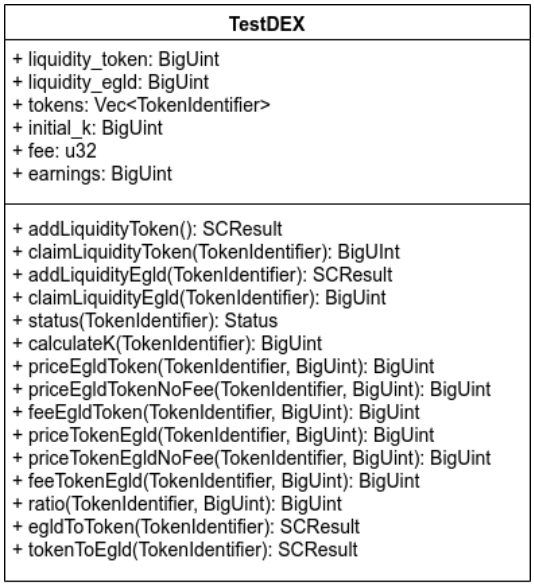


Figura 19: Estructura del contracte intel·ligent.

Aquesta és la seva descripció:

- **liquidity_token(TokenIdentifier)**: Emmagatzema la liquiditat dels *tokens* (la meitat del parell). Per als identificadors del token dels que volguem guardar la informació, tindrem un valor del tipus BigUint[47] associat que defineix la quantitat de *token*.
- **liquidity_egld(TokenIdentifier)**: Guarda la liquiditat en EGLD dels *token* (l'altra part del parell). Per als identificadors del token dels que volguem guardar la informació, tindrem un valor del tipus BigUint associat que defineix la quantitat d'EGLD.
- **tokens**: Guarda un vector de TokenIdentifier, que descriu els *tokens* dels que disposem un parell disponible –*token*-EGLD o EGLD-*token*– per operar.
- **initial_k(TokenIdentifier)**: Per a cada parell (EGLD o EGLD-*token*), emmagatzema un BigUint amb la seva constant k del moment en què es va donar d'alta.
- **fee**: Guarda la taxa –amb un valor del tipus u32[48]– que s'aplicarà quan es faci un intercanvi. S'inicialitza quan es creï el contracte intel·ligent.
- **earnings_egld(TokenIdentifier)**: Emmagatzema les taxes (*fees*) –BigUint– cobrades pels diferents *tokens*.

[47] Estructura de dades que representa un nombre sencer molt gran sense signe.
[48] Sencer sense signe representat amb 32 bits.

- **init(u32)**: Mètode que realitza la tasca de constructor del contracte intel·ligent. Se li passa la taxa que es cobrarà en els intercanvis.
- **add_liquidity_token() ->SCResult**: Afegeix liquiditat a la part "*token*" del parell. Retorna un *enum* anomenat SCResult, forma predeterminada de retornar opcionalment un error ($Err(SCError)$) o una confirmació de resultat satisfactori ($Ok(T)$). El token i la quantitat les agafarà de la mateixa transacció dins de la xarxa Elrond. Mètode que només pot ser cridat pel propietari del contracte.
- **claim_liquidity_token(TokenIdentifier) ->SCResult**: Envia els fons de liquiditat d'un *token* determinat aportats pel propietari del contracte intel·ligent a l'adreça del seu moneder. Se li ha de passar com a paràmetre l'identificador del *token* del qual volem reclamar la liquiditat. Mètode que només pot ser cridat pel propietari del contracte. Retorna un BigUint equivalent a la quantitat transferida. El parell quedarà en estat "Funding".
- **add_liquidity_egld(TokenIdentifier) ->SCResult**: Afegeix liquiditat a la part EGLD del parell. Se li ha de passar com a paràmetre l'identificador d'EGLD. La quantitat l'agafarà de la mateixa transacció dins de la xarxa Elrond. Mètode que només pot ser cridat pel propietari del contracte.
- **claim_liquidity_egld(TokenIdentifier) ->SCResult**: Envia els fons de liquiditat en EGLD del parell d'un determinat *token* aportats pel propietari del contracte intel·ligent a l'adreça del seu moneder. Se li ha de passar com a paràmetre l'identificador del token del que volem guardar la informació. Mètode que només pot ser cridat pel propietari del contracte. Retorna un BigUint equivalent a la quantitat transferida. El parell quedarà en estat "Funding".
- **status(TokenIdentifier) ->Status**: Mostra l'estat d'un parell. Tindrà dos possibles estats: "Successful" i "Funding". El primer significa que està preparat per fer intercanvis i el segon indica que encara s'han d'afegir fons a algun dels components del parell. Se li ha de passar com a paràmetre l'identificador del *token* del que volem retornar aquesta informació. Retorna o bé "Successful" o bé "Funding".
- **calculate_k(TokenIdentifier) ->BigUint**: Calcula la constant k per a un parell en un moment donat de temps. Se li ha de passar com a paràmetre l'identificador del token que identifica el parell del que volem retornar aquesta informació (el que no sigui EGLD). Retorna la citada constant.
- **claim_earnings(TokenIdentifier) ->BigUint**: Envia els beneficis acumulats al contracte resultants d'aplicar la taxa als intercanvis d'un

token determinat. Se li ha de passar com a paràmetre l'identificador del *token* del que volem transferir els beneficis. Mètode que només pot ser cridat pel propietari del contracte. Retorna un BigUint equivalent a la quantitat transferida.

- **price_egld_token(TokenIdentifier, BigUint) ->BigUint**: Calcula el preu d'una certa quantitat d'un *token* determinat en EGLD. Se li ha de passar com a arguments l'identificador del *token* del que volen calcular el preu i la citada quantitat. Retorna el preu com a nombre sencer. S'aplica la taxa (*fee*).

- **price_egld_token_no_fee(TokenIdentifier, BigUint)->BigUint**: Igual que l'anterior però al preu retornat no se li aplica la taxa.

- **fee_egld_token(TokenIdentifier, BigUint) ->BigUint**: Calcula la taxa que es pagarà per una certa quantitat d'un *token* determinat en EGLD. Se li ha de passar com a arguments l'identificador del *token* del que volen calcular el preu i la citada quantitat. Retorna aquest cost com a nombre sencer.

- **price_token_egld(TokenIdentifier, BigUint) ->BigUint**: Calcula el preu d'una certa quantitat d'EGLD en un *token* determinat. Se li ha de passar com a arguments l'identificador del *token* amb el que volem pagar i la quantitat d'EGLD que volem obtenir. Retorna el preu com a nombre sencer. S'aplica la taxa (*fee*).

- **price_token_egld_no_fee(TokenIdentifier, BigUint) ->BigUint**: Igual que l'anterior però al preu retornat no se li aplica la taxa.

- **fee_token_egld(TokenIdentifier, BigUint) ->BigUint**: Calcula la taxa que es pagarà per una certa quantitat d'EGLD en un *token* determinat. Se li ha de passar com a arguments l'identificador del *token* i la citada quantitat. Retorna aquest cost com a nombre sencer.

- **ratio(TokenIdentifier) ->BigUint**: Retorna un nombre sencer amb la relació que hi ha entre els *tokens* que formen el parell. Si la relació és 0 perquè el primer és més petit que l'altre, es retornarà 1. S'emprarà per corregir la constant k per evitar l'error introduit pel redondeig de nombres sencers. Se li ha de passar com a argument el *token* que identifica el parell.

- **egld_to_token(TokenIdentifier) ->SCResult**: Compra de *token* amb EGLD. La quantitat d'EGLD s'agafarà de la transacció. S'enviarà la quantitat del *token* al moneder de l'usuari que faci la compra. Les taxes pagades s'acumularan al contracte intel·ligent. S'haurà de passar com a argument l'identificador del *token* que es vol comprar. Les taxes

es llevaran de la quantitat que ha de rebre l'usuari com a resultat de l'intercanvi

- **token_to_egld() ->SCResult**: Compra d'EGLD amb *token*. El *token* i la quantitat d'EGLD s'agafaran de la transacció. S'enviarà la quantitat d'EGLD al moneder de l'usuari que faci la compra. Les taxes pagades s'acumularan al contracte intel·ligent. Les taxes es llevaran de la quantitat que ha de rebre l'usuari com a resultat de l'intercanvi.

3.2.3 Arquitectura

Una **dApp** ens permetrà escriure en la blockchain i llegir l'emmagatzematge –o estat– del contracte intel·ligent descrit en l'apartat anterior. Com es pot comprovar en la Fig. 20 **per escriure en la cadena de blocs** la dApp haurà de llençar una **transacció** que invoqui a un mètode del contracte. Si per contra només necessitem **llegir l'estat** del contracte intel·ligent, emprarem una **API** aportada per Elrond per invocar certs mètodes del contracte, cosa que no farà pas cap canvi en la cadena de blocs.

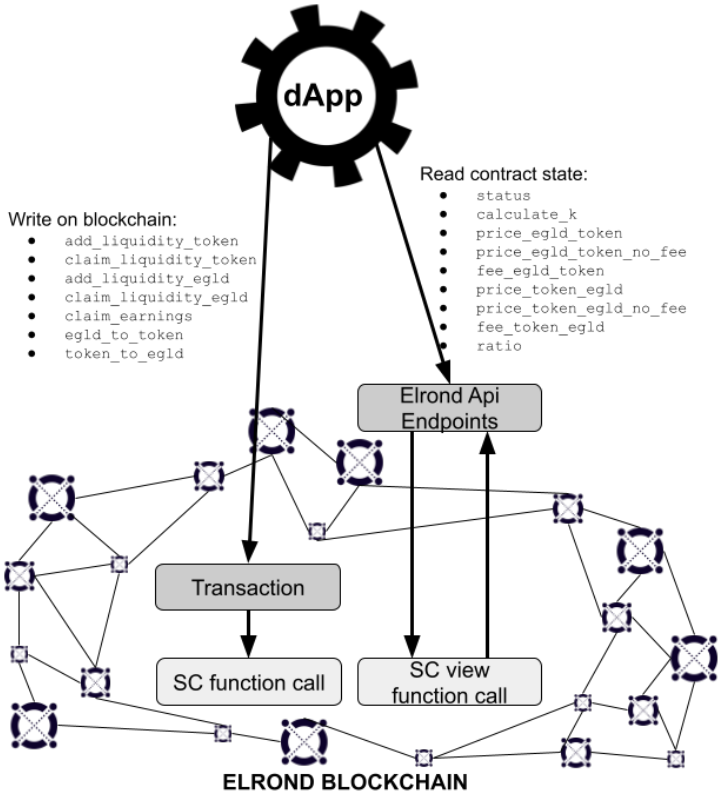


Figura 20: Arquitectura de testDEX. (Font: Elaboració pròpia inspirada en [1]).

Pot parèixer una descripció molt breu, però és que és veritablement una arquitectura molt senzilla.

3.2.4 Pantalles

Per **accedir a la dApp** s'haurà de triar un dels mètodes habilitats per fer-ho a la xarxa Elrond (Fig. 21).

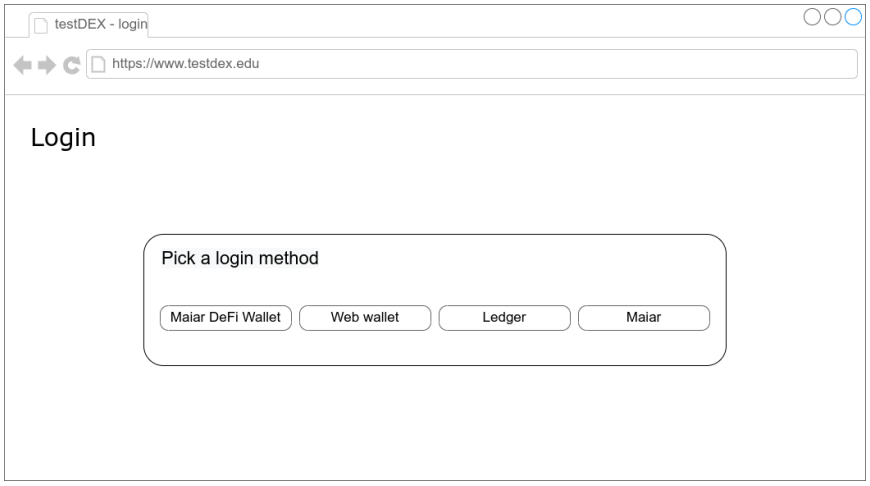


Figura 21: Finestra de login.

Només per a l'**usuari que sigui el propietari del contracte**, li apareixerà en el menú d'adalt a la dreta l'**opció "Fund"**. En aquesta pantalla podrà **crear els diferents fons de liquidesa del contracte intel·ligent** (Fig. 22).

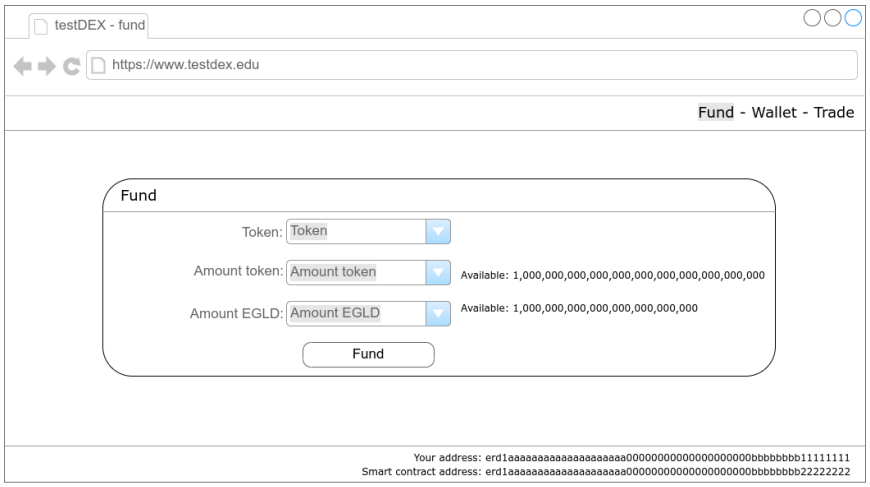


Figura 22: Finestra on es provisionen els *liquidity pools*.

L'**opció de menú "Wallet"** simplement ens durà a una finestra on l'usuari podrà comprovar l'**estat del seu moneder** (Fig. 23).

Figura 23: Finestra on es mostra el moneder de l'usuari.

En l'**opció del menú "Trade"** trobem la finestra amb el cor de l'aplicació (Fig. 24). Podem observar que apareixerà un gràfic on es pot consultar l'**evolució dels preus del parell en el temps**. A la dreta del gràfic es podran **seleccionar el parells i realitzar compres o vendes**. En la part inferior de la pantalla, a l'esquerra apareixerà un **històric de les operacions fetes** i a la seva dreta les **estadístiques sobre les transaccions**. **Es mesuraran el nombre de transaccions, el temps transcorregut i la mitjana de transaccions per minut**.

3.3 Implementació

3.4 Posada en producció

Començarem compilant el contracte intel·ligent. Executarem dins el directori on es trobi el codi font del contracte intel·ligent:

```
erdpy contract build
```

Amb això generarem el fitxer "testdex.wasm" que desplegarem a la *devnet* d'Elrond amb la comanda:

```
erdpy contract deploy --pem="~/wallet/wallet1.pem" \
  --recall-nonce --gas-limit=100000000 --project=. \
  --proxy="https://devnet-gateway.elrond.com" \
  --chain="D" --arguments 0x05 --send
```

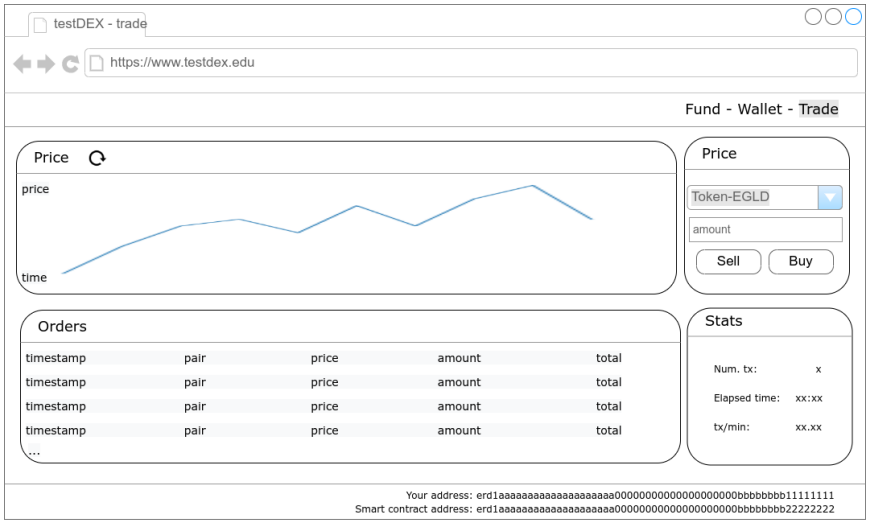


Figura 24: Finestra on es fan els intercanvis.

D'entre els paràmetres, s'ha de destacar que dins de " /wallet/wallet1.pem" es troba la clau privada de l'usuari de la devnet d'Elrond que desplegarà el contracte intel·ligent a la xarxa (que serà el propietari) i amb "–arguments 0x05" li passem en hexadecimal l'argument necessari pel mètode que fa de constructor (un valor de 5 implica una comisió del 0.05%). Aquest comanda mostrarà per la consola l'adreça del contracte intel·ligent dins de la *devnet* d'Elrond, en el nostre cas:

`erd1qqqqqqqqqqqqpgq65ukpqvhhyghtnyysjtltcjndg7wlhl4mq2qey8vpl`

3.5 Proves

Proves.

4 Conclusions

"Este capítulo tiene que incluir:

- Una descripción de las conclusiones del trabajo: Qué lecciones se han aprendido del trabajo?.
- Una reflexión crítica sobre el logro de los objetivos planteados inicialmente: Hemos logrado todos los objetivos? Si la respuesta es negativa, por qué motivo?
- Un análisis crítico del seguimiento de la planificación y metodología a lo largo del producto: Se ha seguido la planificación? La metodología prevista ha sido la adecuada? Ha habido que introducir cambios para garantizar el éxito del trabajo? Por qué?
- Las líneas de trabajo futuro que no se han podido explorar en este trabajo y han quedado pendientes."

5 Glossari

"Definición de los términos y acrónimos más relevantes utilizados dentro de la Memoria."

6 Bibliografia

Referències

[1] The Elrond Team, "Docs · the internet scale blockchain," 2022. [Online]. Available: https://docs.elrond.com/

[2] ——, "Elrond a highly scalable public blockchain via adaptive state sharding and secure proof of stake," 2019. [Online]. Available: https://elrond.com/assets/files/elrond-whitepaper.pdf

[3] cointelegraph.com, "What are decentralized exchanges, and how do dexs work?" [Online]. Available: https://cointelegraph.com/defi-101/what-are-decentralized-exchanges-and-how-do-dexs-work

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[5] W. Dai, "b-money," 1998. [Online]. Available: http://www.weidai.com/bmoney.txt

[6] H. Finney, "Rpow - reusable proofs of work," 2005. [Online]. Available: http://fennetic.net/irc/finney.org/~hal/rpow/

[7] N. Szabo, "Unenumerated: Bit gold," 12 2008. [Online]. Available: https://unenumerated.blogspot.com/2005/12/bit-gold.html

[8] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_White_Paper_-_Buterin_2014.pdf

[9] A. Brownworth, "Blockchain 101 - a visual demo - youtube," 11 2016. [Online]. Available: https://www.youtube.com/watch?v=_160oMzblY8

[10] ——, "Blockchain 101 - part 2 - public / private keys and signing - youtube," 12 2017. [Online]. Available: https://www.youtube.com/watch?v=xIDL_akeras

[11] S. Tual, "Ethereum launches — ethereum foundation blog," 7 2015. [Online]. Available: https://blog.ethereum.org/2015/07/30/ethereum-launches/

[12] D. Mechkaroska, V. Dimitrova, and A. Popovska-Mitrovikj, "Analysis of the possibilities for improvement of blockchain technology," *2018 26th Telecommunications Forum,*

TELFOR 2018 - Proceedings, 2018. [Online]. Available: https://www.researchgate.net/publication/330585021_Analysis_of_the_Possibilities_for_Improvement_of_BlockChain_Technology

[13] L. Mincu, "The maiar defi wallet: A powerful web extension for internet-scale defi · elrond," 9 2021. [Online]. Available: https://elrond.com/blog/maiar-defi-wallet/

[14] B. Mincu, "(11) beniamin mincu on twitter: Everything we've built at elrond is prepared for the mainnet launch, pending security audits. spent the whole day with our team in an internal hacking sprint taking the network apart. everything else matters only if elrond is secure. heavy security audits ongoing. / twitter," 11 2019. [Online]. Available: https://twitter.com/beniaminmincu/status/1200835749412777984

[15] D. Nelson, "Elrond will pay you 60,000 usd to break its blockchain - coindesk," 6 2020. [Online]. Available: https://www.coindesk.com/tech/2020/06/10/elrond-will-pay-you-60000-to-break-its-blockchain/

[16] B. Mincu, "Battle of yields: 100,000 usd maiar exchange incentivized dex competition · elrond," 9 2021. [Online]. Available: https://elrond.com/blog/battle-of-yields-announcement/

[17] Baro Virtual - Coinmonks, "Ethereum and solana competitors," 9 2021. [Online]. Available: https://medium.com/coinmonks/ethereum-and-solana-%D1%81ompetitors-2638afd96ef2

[18] Cryptopedia Staff, "What is an automated market maker (amm)?" 3 2021. [Online]. Available: https://www.gemini.com/cryptopedia/amm-what-are-automated-market-makers

[19] The Elrond Team, "Technology built for internet scale · elrond," 2022. [Online]. Available: https://elrond.com/technology/

[20] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology 2004 17:4*, vol. 17, pp. 297–319, 7 2004. [Online]. Available: https://link.springer.com/article/10.1007/s00145-004-0314-9

[21] J. Ćwirko, "Elven tools - tips on buying nfts on the elrond blockchain." [Online]. Available: https://www.elven.tools/docs/tips-on-buying-nfts-on-the-elrond-blockchain.html

[22] MongoDB, "Sharding — mongodb manual," 2021. [Online]. Available: https://docs.mongodb.com/manual/sharding/

[23] M. Foundation, "Webassembly — mdn," 2 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/WebAssembly

[24] B. Mincu, "The elrond virtual machine - smart contracts at internet scale," 12 2020. [Online]. Available: https://elrond.com/blog/elrond-virtual-machine-arwen-wasm-vm/

[25] J. Penaflor, "How do i setup my elrond validator?" 12 2019. [Online]. Available: https://medium.com/@jeff.penaflor1983/how-do-i-setup-my-elrond-validator-52a35aec43f2

[26] B. Mincu, "After exceeding 10k tps in testnet, elrond is going open source," 6 2019. [Online]. Available: https://medium.com/elrondnetwork/after-exceeding-10k-tps-in-testnet-elrond-is-going-open-source-61dd25e93fd9

[27] Everstake, "Bringing defi to the next billion with maiar exchange," 9 2021. [Online]. Available: https://medium.com/everstake/bringing-defi-to-the-next-billion-with-maiar-exchange-869101bf555a

[28] The Elrond Team, "Welcome to maiar dex — maiar exchange docs," 2021. [Online]. Available: https://docs.maiar.exchange/

[29] J. L. de la Rosa Esteva, *Finances descentralitzades (DeFi)*, V. G. Font, Ed. Barcelona: FUOC, 2021.

[30] V. Buterin, "Improving front running resistance of x*y=k market makers," 03 2014. [Online]. Available: https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281

7 Annexos

"Listado de apartados que son demasiado extensos para incluir dentro de la memoria y tienen un carácter autocontienido (por ejemplo, manuales de usuario, manuales de instalación, etc.)

Dependiente del tipo de trabajo, es posible que no haya que añadir ningún anexo."