Desenvolupament d'una dApp sobre Ethereum

Jordi Alejandre Camps

Resum– Amb el creixement exponencial en el sector de les tecnologies blockchain i davant de la innovació que aquestes aporten, la necessitat de desenvolupar aplicacions que interactuïn de forma directa amb la cadena de blocs ha augmentat de forma paral·lela. Aquestes aplicacions són les anomenades dApps o aplicacions descentralitzades. En aquest projecte, s'ha decidit desenvolupar una dApp concretament sobre Ethereum. Es tracta d'un exchange descentralitzat de criptomonedes, rep aquest nom, ja que no fa ús d'una figura centralitzada per realitzar els intercanvis sinó que en lloc d'això fa servir la tecnologia dels smart-contracts, l'exchange fa intercanvis entre dues criptomonedes, la primera és l'Ether, la qual és la moneda nativa d'Ethereum i la segona és el TFG Token, que ha estat creada amb el propòsit de poder fer intercanvis. Dins de l'article també es fa èmfasi en veure quins són els passos que se segueixen per a poder crear una aplicació descentralitzada i sobretot quines eines s'hi utilitzen i per que.

Paraules clau– Blockchain, dApp, Ethereum, exchange, criptomonedes, smart-contracts, Ether, Token, ReactJS, Web3.js, Metamask, Solidity, Truffle, Chainlink Data Feeds, DeFi.

Abstract– With the exponential growth in the blockchain technologies and in the face of the innovation that they bring, the need to develop applications that interact directly with the blockchain has increased in parallel. These applications are called dapps or decentralised applications. In this project, I have decided to develop a dApp specifically on the Ethereum blockchain. It is a decentralised exchange of cryptocurrencies, given that it does not use a centralized figure to perform the exchanges, but instead uses smart-contracts, the exchange does the operations between two cryptocurrencies, the first being Ether, which is the native currency of Ethereum and the second being TFG Token, which has been created for the purpose of making this operations possible. Within the article, emphasis is also placed on seeing what steps are taken to create a decentralised application and, above all, which tools are used and why.

Keywords– Blockchain, dApp, Ethereum, exchange, cryptocurrencies, smart-contracts, Ether, Token, ReactJS, Web3.js, Metamask, Solidity, Truffle, Chainlink Data Feeds, DeFi.

1 Introducció - Context del treball

Aquest projecte té el seu context en el món DeFi (Decentralized Finance) el qual consisteix en una infraestructura financera basada en la tecnologia Blockchain. [1]

En altres paraules, DeFi és un projecte financer que té com a propòsit descentralitzar els principals casos tradicionals d'ús financer, com les inversions, els préstecs, la gestió del patrimoni, el comerç, les assegurances, etc. I ho aconsegueix fent ús de les dApps o Aplicacions Descentralizades, que consisteixen en aplicacions que s'executen sobre una cadena de blocs utilitzan smart-contracts. Un smart-contract és un programa infòrmatic que implica un contracte d' autoexecució que comprén els termes de l'acord entre els usuaris implicats en aquest, que s'escriuen directament en un llenguatge de programació. Aquests smart-contracts es despleguen a la blockchain perquè puguin ser funcionals. [2]

Concretament en el cas d'aquest projecte la blockchain sobre la qual es construirà la dApp i, per tant, en la que s'executaran els smart-contracts dissenyats perquè l'aplicació funcioni com es desitja, és la blockchain d'Ethereum. [3] Precisament en aquesta cadena de blocs tenim l'anomenada Ethereum Virtual Machine la qual és una màquina virtual descentralitzada on s'hi executen els smart-contracts.

- E-mail de contacte: jordi.alejandre10@gmail.com
- Menció realitzada: Tecnologies de la Informació
- Treball tutoritzat per: Jordi Herrera Joancomarti (DEIC)
- Curs 2021/22

Per entrar més en detall sobre l'ecosistema de dApps d'Ethereum, cal afegir que donada la naturalesa de codi obert d'Ethereum i, per tant, la facilitat que dona a l'hora de crear dApps, des del 2015, Ethereum s'ha convertit en l'ecosistema unificador per a moltes organitzacions, fins al punt que aproximadament la meitat de totes les dApps que funcionen al mercat es basen en la xarxa d'Ethereum amb més de 600.000 usuaris actius que interactuen amb les dApps regularment. És la segona cadena de blocs més gran, en bona part a causa de l'èxit del projecte per atraure desenvolupadors per crear dApps a la seva xarxa. És per aquestes raons que s'ha decidit treballar i crear la dApp d'aquest projecte sobre aquesta cadena de blocs. [4]

L'objectiu d'aquest projecte, per tant, consisteix a construir una dApp junt amb els seus respectius smart-contracts per a poder oferir un servei financer, el qual es tracta d'un exchange o intercanvi de divises descentralitzat, a més a més d'un segon mòdul de visualització de detalls de l'última transacció realitzada, on es poden observar algunes de les possibilitats que proporciona la tecnologia de Web3.js, que permet extreure la informació de la transacció. També es fa èmfasi dins de l'article en quines han estat les eines utilitzades i la raó del seu ús.

2 OBJECTIUS

Tal com s'ha descrit al final de l'apartat anterior l'objectiu principal d'aquest projecte consisteix a desenvolupar una dApp sobre la blockchain d'Ethereum i aquesta dApp en qüestió constarà de dos mòduls principals, a més a més de la seva interfície d'usuari.

En primer lloc, l'aplicació descentralitzada comptarà amb un exchange descentralitzat el qual funcionarà mitjançant smart-contracts i la llibreria web3.js perquè aquests puguin interactuar amb el Front-End que veurà l'usuari.

Un exchange descentralitzat (DEX) és una plataforma digital que funciona com un exchange tradicional, però que a diferència d'aquest, en el centre del servei hi opera un smart-contract. Això el que permet és eliminar una sèrie d'intermediaris, i que d'aquesta manera siguin més transparents i conservin més la privacitat de l'usuari. La confiança i la gestió de capital en un exchange descentralitzat no recau en una figura central sinó que en lloc d'això els usuaris d'aquest mantenen en tot moment el control absolut dels seus actius, la qual és una característica que agrega un alt nivell de seguretat i privacitat. [5]

A més a més, també es farà ús de la tecnologia Chainlink Data Feeds, perquè d'aquesta manera dintre de l'exchange es puguin obtenir els preus de mercat en temps real a través dels propis smart-contracts, i així poder actualitzar l'exchange rate dinàmicament.

En segon lloc, el segon mòdul de l'aplicació consistirà en un visualitzador dels detalls de l'última transacció realitzada, és a dir, si per exemple es fa una operació de compra de TFG Tokens, aquest visualitzador s'encarregarà de mostrar els detalls d'aquesta transacció dins de la blockchain (hash de la transacció, número de bloc, adreça del contracte, etc.), tota aquesta informació s'obtindrà fent ús principalment de Web3.js el qual té funcions per poder realitzar aquest tipus d'activitats.

En resum, l'objectiu principal del projecte consisteix a desenvolupar una dApp sobre la blockchain d'Ethereum que disposa de dos mòduls principals com hem vist anteriorment, un exchange descentralitzat i un visualitzador de detalls de l'última transacció realitzada, a més a més de comptar amb la seva pròpia interfície d'usuari, funcionen a partir dels seus smart-contracts respectius que s'implementaran durant el projecte.

3 ESTAT DE L'ART

Com ja s'ha mencionat anteriorment, les dApps o aplicacions descentralitzades és una tecnologia que amb els anys ha anat creixent cada cop més, agafant un paper cada cop més important per al desenvolupament del que s'anomena Web3, la propera iteració de la web. Web3 es tracta de la descentralització de la web tal com la coneixem i d'aquesta manera donar als usuaris més control sobre les seves dades [6]. Abans d'entrar més en detall, primer cal entendre els models anteriors a aquest:

La primera versió de la web o Web1, era un ecosistema obert. Estava dirigit per persones reals que van crear els seus propis llocs web, però aquests llocs eren limitats, ja que la gran majoria eren només de lectura, de manera que les dades fluïen de la web a l'usuari i aquí acababa la interacció entre els dos.

Anys més tard, les grans empreses com Google o Facebook, van crear una nova iteració de la web (Web2), la qual és la que tenim avui en dia, on les pàgines són molt més interactives amb l'usuari, ja que li permeten realitzar moltes més accions com ara compartir arxius, descarregar-ne, xatejar, etc. No obstant això, la web en conseqüència va anar agafa'n tendència a la centralització, ja que aquestes grans empreses tenen grans servidors als quals arriben totes les peticions dels diversos usuaris. A més a més, les interaccions d'aquests, amb el temps ha guanyat una importància vital, ja que les empreses utilitzen aquesta informació per crear noves plataformes i també anuncis orientats i personalitzats. Llavors, en aquest sentit, hi ha teòrics que opinen que això crea un entorn on l'usuari té poca o cap autonomia sobre les seves dades.

I finalment, sorgeix el concepte de Web3, el qual va ser creat el 2014 pel cofundador d'Ethereum, Gavin Wood tot i que no ha agafat realment consciència pública i popularitat dins de la comunitat fins a l'any passat.

Respecte a les iteracions de la web anteriors, els canvis més rellevants que aporta Web3 estan focalitzats en la forma que les plataformes web es connecten a les fonts de dades i on resideixen aquestes. Tal com s'ha mencionat anteriorment, quasi totes les aplicacions de Web2 depenen d'una base de dades centralitzada. En el cas de Web3, les aplicacions i serveis utilitzen una cadena de blocs o "blockchain". La idea principal que hi ha al darrere, és que no existeix una autoritat central arbitraria, sinó una forma de consens distribuït, aquesta tecnologia brinda una forma d'autogovern amb un enfocament a la descentralització de la web, això permet també als usuaris aprofitar aquesta tecnologia per tenir molt més control sobre les seves dades respecte al model anterior. Cal mencionar que les finances i la capacitat de pagar béns i serveis amb una forma de pagament descentralitzat s'habiliten amb l'ús de les criptomonedes que també es construeixen sobre les cadenes de blocs.

En conclusió, és aquí on les dApps o aplicacions descentralitzades prenen un paper important en tot aquest nou

ecosistema, ja que per a poder fer ús d'aquesta tecnologia de cadena de blocs que proporciona descentralització i privacitat entre altres coses, es necessiten aplicacions especialitzades fetes a partir dels smart-contracts que l'utilitzin i d'aquesta manera serveixin de ponts entre la blockchain i els usuaris.

4 METODOLOGIA I PLANIFICACIÓ

4.1 Metodologia

La metodologia de treball que s'ha utilitzat per a la realització d'aquest projecte ha estat la metodologia Kanban, la qual consisteix en un sistema de producció tant eficient com efectiu. Es tracta d'una metodologia àgil i el seu objectiu és gestionar la realització de les tasques fins a la seva finalització. [7]

L'organització d'aquesta metodologia consisteix en un taulell de tasques amb el que poder millorar la feina a fer i mantenir un ritme sostenible. Aquest taulell s'organitza per columnes en les quals s'anotaran els diferents estats del flux de treball, i d'aquesta manera cobrir tots els estats que pugui arribar a tenir una tasca des de la seva creació fins a la seva finalització. A més a més les tasques es vagin creant poden prioritzar-se i col·locar-se en les seccions o columnes més convenients.

Aquest mètode es basa en el desenvolupament incremental, és a dir, en la divisió del treball en diferents parts. Això també implica, que s'eviti la sobreproducció, ja que, en aquesta metodologia hi ha certs principis que s'han de seguir com el fet que es doni prioritat a aquelles tasques que es marquin com a urgents i també que totes les tasques que estiguin en curs no poden excedir un cert nombre de tasques, d'aquesta manera també podrem controlar més eficientment el flux de treball.

S'ha decidit utilitzar aquesta metodologia, ja que permet veure de una forma visual i eficaç el repartiment de la feina i per tant facilitar en gran mesura l'organització d'aquesta en el temps, a més a més de la flexibilitat que dona a l'hora de treballar.

4.2 Planificació

Pel que fa a la planificació, aquest és un punt també molt important per a poder dur a terme el projecte correctament i en els terminis estipulats. Durant el transcurs del treball, han anat apareixent imprevistos de temps que han fet que la planificació d'aquest s'anés reajustant.

No obstant, les fases principals de la planificació s'han mantingut. Principalment, s'ha dividit en dues fases: començant pel desenvolupament del BackEnd i després el desenvolupament del FrontEnd.

La primera, ha consistit en el desenvolupament dels dos smart-contracts principals que fan funcionar l'exchange de l'aplicació dins de la blockchain d'Ethereum. En primer lloc, s'ha començat pel smart-contract encarregat de gestionar l'exchange, és a dir, el responsable de fer els intercanvis de divises i posteriorment s'ha procedit amb l'encarregat de generar la criptomoneda TFG Token. A banda del desenvolupament dels contractes, en aquesta primera fase també ha calgut una investigació i un aprenentatge previs sobre el funcionament del llenguatge que utilitzen, Solidity; i també quins són els diferents entorns que existeixen per a treballar amb smart-contracts i quins són els que millor s'ajusten al projecte.

La segona fase, el desenvolupament del FrontEnd, ha estat constituïda de dues activitats principals, la primera d'aquestes ha estat construir el pont entre la interfície d'usuari i la blockchain on estan els nostres smart-contracts que faran de BackEnd, per a poder fer aquesta connexió s'ha investigat i treballat amb les eines Web3.js i Metamask, explicades amb més detall en següents apartats de l'article. De forma paral·lela a la construcció d'aquest pont, s'ha anat programant la interfície d'usuari, per la qual les eines a usar han estat ReactJS i Bootstrap principalment. Per aquesta última activitat també ha calgut una preparació i un aprenentatge del funcionament d'aquestes eines.

Per acabar, cal mencionar que durant el desenvolupament de les diferents tasques en aquest projecte s'ha començat treballant amb una Blockchain local (Ganache) mentre es feia el desenvolupament i després s'ha passat a treballar amb una testnet pública (Rinkeby) per a realitzar les proves finals.

5 APLICACIÓ

5.1 Entorn

En aquest apartat s'explicaran les diferents tecnologies utilitzades per a la realització del projecte i les raons de la seva utilització.

5.1.1 Web3.js

En primer lloc, una de les eines més importants pel projecte ha estat Web3.js, aquesta es tracta d'una col·lecció de móduls els quals contenen una funcionalitat específica per l'ecosistema d'Ethereum, la llibreria s'encarrega de fer de pont i connectar la informació que esta guardada en els nodes de la blockchain d'Ethereum amb el nostre front-end, a través de diferents funcions.

5.1.2 Metamask

En aquest cas, és una extensió de navegador intel·ligent que actua com una wallet. Metamask ens permet crear una wallet per a Ethereum i els tokens basats en l'estàndard ERC-20.

Aquesta wallet de navegadors fa ús de la llibreria Web3.js, vista a l'apartat anterior. I per tant conjuntament amb aquesta, Metamask està pensat per a ser una wallet per a Ethereum i a més com una eina per a poder interactuar amb les dApps. Es genera un canal de comunicació entre l'extensió de navegador i la dApp. Qualsevol dApp pot detectar Metamask i l'usuari accedir-hi. D'aquesta manera podem accedir a la informació de la blockchain que necessitem per a poder fer funcionar l'aplicació descentralitzada. [8]

5.1.3 Truffle

Truffle és un conjunt d'eines que permet als desenvolupadors crear aplicacions sostenibles i professionals en qualsevol blockchain, també anomenat entorn de desenvolupament de smart-contracts. [9][10]

Un desenvolupador blockchain pot crear smart-contracts pel seu compte o pot utilitzar aquests entorns de desenvolupament. El motiu darrere del seu ús, és el fet que faciliten la realització de la feina a fer, oferint al desenvolupador una plataforma per implementar smart-contracts, fer-ne els seus deployments respectius a la blockchain, compilar i sobretot el que més ens interessa: poder fer-ne el seu testeig, a més a més d'altres funcionalitats. És per aquest motiu que qualsevol aplicació descentralitzada mínimament funcional dins del ecosistema, compte amb l'ús d'aquest tipus de software.

Es va arribar a la conclusió de fer servir aquest tipus d'eina respecte d'altres com la molt popular Remix, per les següents raons: En primer lloc, el sistema d'organització de codi, ja que tot i que Remix et permet emmagatzemar online els teus smart-contracts, existeix el problema de què si estàs desenvolupant una aplicació descentralitzada amb la seva interfície i les seves connexions per Web3.js, és molt més còmode per al desenvolupador tenir els smart-contracts en una carpeta dins del mateix projecte i a més a més també poder treballar amb ells dins del mateix IDE que ja es feia servir per poder codificar la resta de l'aplicació, per tant, el desenvolupador hi està més familiaritzat, a aquesta raó se li suma el fer més fàcil portar un millor control de versions, ja que si els smart-contracts estan dins d'una carpeta del projecte també es podrà anar seguint la seva evolució quan es guardi tot en un repositori, per tant, el primer punt és la comoditat de cara al desenvolupador.

En segon lloc, una característica crítica mencionada anteriorment i que verdaderament suposa una diferència respecte Remix, consisteix en la possibilitat d'automatitzar tests dels smart-contracts del projecte, i a més, poder-ho fer amb una sola línia de comanda ("truffle test"). Quan s'està desenvolupant una aplicació completa que estarà de cara als usuaris és necessari poder fer tests per assegurar-se del seu correcte funcionament, i en aquest aspecte, Truffle aporta moltes facilitats perquè pugui realitzar-se en condicions, tal com es veurà més endavant en l'apartat de desenvolupament.

En tercer lloc, hi ha un punt molt important a tenir en compte, i aquest consisteix en que Truffle permet fer un deployment escriptable, és a dir, tu pots programar com vols que es faci el deployment dels smart-contracts a la blockchain. Per entrar més en detall, pot posar-se d'exemple el cas particular d'aquest projecte on es fa ús d'aquesta opció. En el projecte tal com s'ha mencionat en apartats anteriors es desenvolupen principalment dos smart-contracts, l'encarregat de fer els intercanvis entre criptomonedes i el responsable de generar una criptomoneda (TFG Token), perquè l'usuari pugui fer intercanvis, el smart-contract que gestiona les operacions necessita poder disposar de TFG Tokens els quals es generen quan es fa el deployment de l'altre smart-contract, per tant la solució a aquest problema és crear un script que en el moment de passar per línia de comandes que es desitja fer el deployment de tots els smart-contracts del projecte, automàticament enviï tots els TFG Tokens generats a l'exchange i d'aquesta manera pugui operar com s'espera que ho faci.

Finalment, en quarta posició, hi ha altres característiques que no s'han de passar per alt com el fet que Truffle compti amb una CLI intuïtiva de cara al desenvolupador, ja que en alguns casos, accions que de normal comportarien varies línies de comanda, en el cas de Truffle, es tradueix en una sola, com és el cas de "truffle test"o "truffle build".

Per tant, un cop revisades tota aquesta sèrie de raons pot arribar-se a la conclusió que una eina com Remix és de caràcter més experimental, per exemple per si s'està començant a programar en solidity i sorgeix la necessitat de fer certes proves, però si el que es busca, en canvi, és una eina de caràcter més productiu, és a dir, una eina que compti amb tots els recursos necessaris per a poder desenvolupar una aplicació descentralitzada completa, objectivament, és una opció molt més òptima fer ús d'una eina com Truffle.

Per aquest motiu, va haver-hi una fase de selecció i una fase de revisió de la documentació de l'entorn seleccionat. En la primera, es van explorar diferents opcions, principalment Truffle i Hardhat. Va decidir-se utilitzar l'entorn de Truffle (tot i que no es descartava canviar d'entorn en el futur si fos necessari) a causa de la seva àmplia utilització per part de la comunitat en els últims anys i el fet de proporcionar les eines essencials per al desenvolupament, i d'aquesta manera no continuar desviant tant de temps a l'exploració i comparació dels diferents entorns, i així poder-se centrar més en el desenvolupament.

5.1.4 ReactJS

React és una biblioteca JavaScript de codi obert dissenyada per a crear interfícies d'usuari amb l'objectiu de facilitar el desenvolupament d'aplicacions en una sola pàgina. És mantingut per Facebook i la comunitat de programari lliure.

La raó d'utilitzar aquesta eina és causada per la naturalesa d'una aplicació descentralitzada. En el cas d'una aplicació convencional tenim, per una banda, el FrontEnd, és a dir, la part del client, i el BackEnd, la banda del servidor, en aquest cas un servidor que processarà les nostres peticions. Però en el cas d'una aplicació descentralitzada, la diferencia principal està en el fet que el BackEnd passi a ser una blockchain, això es tradueix en que la dApp farà ús de Web3.js i Metamask per poder modificar qualsevol tipus de dades i, per tant, utilitzarà les claus pública i privada emmagatzemades a Metamask, per tant, per fer les transaccions necessàries no es farà servir un servidor, ja que aquest mai es farà càrrec d'aquestes claus i, per tant, no serà responsable de modificar les dades de cap cadena de blocs, aquesta és una responsabilitat que passa a ser només de l'usuari, llavors, pot arribar-se a la conclusió que la banda del client necessitarà tenir bastanta més intel·ligència que en una aplicació convencional, i és per això que una eina enfocada a desenvolupar aplicacions en una sola pàgina com ReactJS, és a dir, on la majoria de la lògica de l'aplicació resideix en el FrontEnd és molt més convenient.

5.1.5 Etherscan

Etherscan és una organització independent que examina la cadena de blocs compartida d'Ethereum Blockchain (similar a una base de dades descentralitzada) i posa aquesta informació a la disposició del públic en general a través del seu lloc web. [11]

Etherscan és una eina valuosa per als usuaris de la xarxa Ethereum que desitgen realitzar un seguiment de les transaccions, examinar contractes intel·ligents, obtenir estadístiques i, en general, mantenir-se al dia amb el que succeeix en la cadena de blocs d'Ethereum.

En el cas del projecte, s'ha usat de manera freqüent per a poder fer proves i poder fer comprovacions de com es comporten els smart-contracts del projecte dins de la Blockchain.

5.1.6 Firebase

Firebase de Google és una plataforma en el núvol per al desenvolupament d'aplicacions web i mòbil. [12]

Dins d'aquesta plataforma s'ha fet ús concretament de Firebase Hosting, el qual és un servei de hosting de contingut web amb nivell de producció orientat a desenvolupadors.

En la fase final del projecte s'ha configurat l'aplicació web perquè pogués ser allotjada a internet a través d'aquesta tecnologia.

5.2 Desenvolupament

En aquest apartat de desenvolupament, un cop havent revisat les diferents eines utilitzades dins del projecte, s'entrara en detall a cada fase de la realització del projecte.

5.2.1 Fase creació i organització de smart-contracts

La primera fase, un cop havent fet la recerca sobre Solidity i sobre com funciona la llibreria Web3.js, a més a més de la cerca d'exemples útils per a la realització de l'aplicació, consisteix precisament en el desenvolupament dels smart-contracts que formaran part del projecte. Començant pels referents a l'exchange descentralitzat tal com va exposar-se en la planificació.

En aquest cas, va explorar-se una varietat d'opcions, ja que, es van trobar tota classe d'exemples, a causa de totes les possibilitats existents per a poder elaborar un exchange descentralitzat. Les opcions van anar des d'utilitzar les eines que proporcionaven algunes plataformes com és el cas de Moralis amb el seu 1inch Dex plugin [13], el qual es tracta d'un exchange descentralitzat ja existent que ja compta amb els seus propis smart-contracts per poder funcionar, fins a la construcció manual dels smart-contracts que constitueixen el exchange descentralitzat.

Aquesta última opció, a efectes pràctics presentava una limitació en les possibilitats que es podien dur a terme amb l'exchange en qüestió, però per altra banda considerant que el projecte consisteix a construir una aplicació descentralitzada des de zero, aquesta opció es presentava com l'opció més atractiva i amb més sentit respecte a la mencionada anteriorment, ja que permetia demostrar els coneixements adquirits en l'elaboració de smart-contracts i en el seu llenguatge de programació, Solidity. Per tant, es va decidir seguir aquesta línia de treball.

Un cop preses aquestes decisions, es van començar a desenvolupar els smart-contracts esmentats. L'exchange en qüestió, compleix la seva funció principal de compra i venda, concretament, de dues criptomonedes diferents, la primera és l'Ether, la qual és la criptomoneda nativa de la blockchain d'Ethereum, ja que estem construint aquesta aplicació sobre aquesta, i la segona és TFG Token, una criptomoneda creada en aquest projecte amb el propòsit de poder fer aquests intercanvis entre les dues criptomonedes dins del exchange i poder proporcionar les eines necessàries al exchange per funcionar, les quals ara s'entrara en més detall de quines es tracten.

L'organització d'aquests contractes intel·ligents, va consistir en la que pot apreciar-se en el diagrama següent:

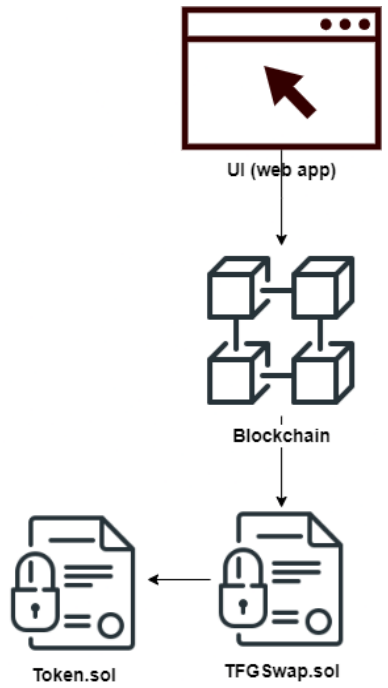

Fig. 1: Diagrama de l'organització dels smart-contracts de l'exchange dins de l'aplicació

Principalment, l'exchange descentralitzat està constituït per dos smart-contracts tal com pot apreciar-se en el diagrama, en primer lloc, tenim el TFGSwap.sol i, en segon lloc, el Token.sol. El primer consisteix en el propi exchange, és a dir, és l'encarregat de recollir les peticions de compra i venda de tokens, i executar-les.

I el segon, és el smart-contract encarregat de fer la creació del TFG Token esmentat anteriorment, en aquest cas, el smart-contract està construït a partir del estàndard Token ERC-20 que habilita Ethereum per a poder crear tokens dins de la seva pròpia blockchain i que aquests puguin funcionar com a criptomoneda sense la necessitat d'haver de crear la seva pròpia cadena de blocs.

Aquest estàndard consisteix en un conjunt de funcions i events que proporcionen la propietat que fa que cada token sigui exactament igual (en tipus i valor) que un altre token. Proporciona funcionalitats com transferir tokens d'un compte a un altre, per obtenir el saldo actual de tokens d'un compte i també el subministrament total del token disponible a la xarxa. A més d'aquestes, també té altres funcionalitats com ara aprovar que una quantitat de token d'un compte pot ser gastada per un compte de tercers. Per a poder realitzar-se aquesta part es va consultar la documentació que proporciona Ethereum per l'estàndard ERC-20 [14].

El funcionament entre aquests dos contractes, és el següent: Ens interessa que el smart-contract encarregat d'actuar com a exchange (TFGSwap.sol) tingui possessió de tots els TFG tokens, ja que, és des d'aquest smart-contract des d'on s'efectuaran les operacions de compra i venda, per tant, el primer que es farà és, un cop fet el deployment de Token.sol on s'hi haurà definit en aquest, el subministrament total de tokens, es transferiran tots, fent ús del seus propis mètodes definits per ERC-20, a TFGSwap.sol, perquè pugui complir amb el seu propòsit.

Seguint amb el TFGSwap.sol, aquest smart-contract necessita principalment dues funcions, la de comprar tokens (BuyTokens) i la de vendre aquests tokens (SellTokens). En els dos casos per a poder dur a terme la implementació, s'haurà d'importar primer el Token.sol al segon smart-contract, ja que, se'n farà ús dels seus mètodes per a poder transferir tokens entre diferents comptes dins de la blockchain.

En primer lloc, per a poder implementar l'operació de compra, principalment és farà servir el mètode de transferència d'un compte a un altre, a més a més d'altres instruccions com el càlcul de tokens a transferir segons el rate (el qual de moment en aquesta fase és fix en 1 ETH = 100 TFG Tokens, però canviarà més endavant amb la implementació d'oracles). I en segon lloc, la funció SellTokens es té que és diferent de l'anterior perquè a part de fer l'operació contrària necessita un pas addicional, a causa del funcionament de Solidity. Per a poder gastar tokens d'un compte que no pertany al smart-contract, necessita el seu permís abans de disposar dels seus fons per a poder efectuar l'operació (transferFrom).

Per implementar completament ambdós smart-contracts, el següent pas a seguir és el de fer-ne el seu testeig, el qual és un pas crític i important dins de l'apartat de desenvolupament, ja que, es vol assegurar de què el codi elaborat compleix el seu proposit sense problemes. A més a més, en aquest cas és especialment rellevant ja que tractant-se de smart-contracts, aquests un cop s'hagi fet el seu deployment a la blockchain a causa de la seva naturalesa no es podran modificar posteriorment.

Per tant, per a poder continuar implementant l'aplicació eficientment i correctament es fa ús d'un entorn de desenvolupament de smart-contracts, per a poder acomplir totes aquestes tasques de forma més automatitzada i ordenada possible, tal com s'exposa en apartats anteriors això suposa un canvi en la planificació del projecte.

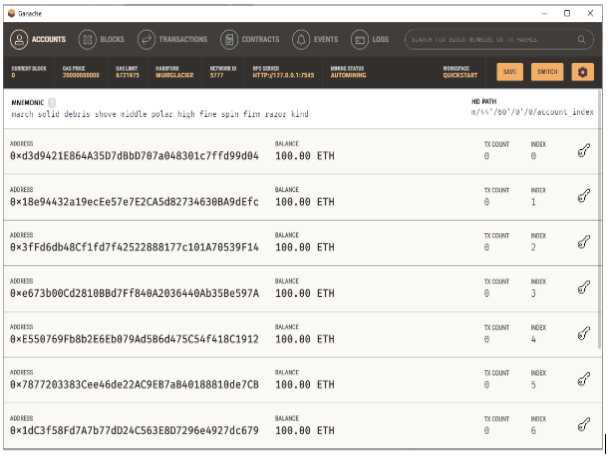

Fig. 2: Blockchain local proporcionada per Ganache

Aquest entorn és el de Truffle Suit, el qual compte també amb una eina anomenada Ganache, que consisteix en la ràpida implementació d'una blockchain local en la que disposes de vàries comptes juntament amb les seves claus, amb les que pots fer-ne proves, ja que gaudeixen cada una d'elles de 100 Ether vàlids dintre d'aquesta blockchain de testeig en la que també s'hi pot fer el deployment dels teus smart-contracts. A més a més, aquesta eina compta amb una interfície gràfica molt útil que és pot observar a la figura adjunta d'aquest apartat.

En aquest punt del projecte, veient el cicle natural de les aplicacions que utilitzen Truffle Suit, va decidir-se fer ús de l'eina Ganache per a fer les proves adients durant el desenvolupament dels smart-contracts i després, per a realitzar les proves finals a l'aplicació es va decidir fer ús de la testnet pública Rinkeby, la qual és una blockchain per desenvolupadors que vol emular la blockchain principal d'Ethereum.

5.2.2 Implementació de Web3.js i FrontEnd de l'aplicació

Després de desenvolupar els smart-contracts necessaris per a fer funcionar l'exchange descentralitzat i que aquests passin els seus respectius tests fent ús de la llibreria chai, que implementa l'entorn de Truffle, es passa a la fase de la implementació de la tecnologia de Web3.js.

Aquesta tasca, a mesura que es va anar desenvolupant l'aplicació va quedar clar que anava completament de la mà amb la fase del desenvolupament web del Front-End de l'aplicació, per tant, ambdues tasques es van realitzar paral·lelament. Per començar explicant aquesta fase del projecte primer cal comentar com s'ha dut a terme la part del Front-End, en aquest cas, les eines utilitzades han sigut principalment React.js i Bootstrap. [15] [16]

ReactJS és una biblioteca Javascript de codi obert dissenyada per crear interfícies d'usuari amb l'objectiu de facilitar el desenvolupament d'aplicacions a una sola pàgina. Dos dels exchanges de criptomonedes més importants: Coinbase i Crypto.com usen aquesta eina per al seu desenvolupament.

Un dels motius en l'ús d'aquesta eina, a més a més del ja comentat en apartats anteriors, ve influenciat i inspirat per no haver tractat prèviament amb aquesta eina, el fet que sigui tan usada avui dia i precisament en el sector de les aplicacions descentralitzades, va portar a decidir utilitzar-la i guanyar nous coneixements a mesura que s'anava dissenyant l'aplicació.

Respecte a la decisió d'usar Bootstrap, en aquest cas es va decidir utilitzar aquesta eina de codi obert per a no haver de dedicar massa temps a la part dels estils de la web, ja que es preferia invertir aquest temps als aspectes més funcionals d'aquesta.

En la imatge següent es poden observar els elements principals del exchange descentralitzat, en primer lloc, tenim l'input i l'output, que varien quan es prem el botó verd amb el símbol de les fletxes, varien segons l'acció que l'usuari desitja realitzar en aquell moment, en el cas de la imatge esta posada l'acció per defecte, que és la de realitzar la compra de tokens, és per aquest mateix motiu que en l'input ens apareix la indicació a la dreta que la quantitat que hi posem serà en Ether i en l'Output ens indica la quantitat equivalent al que haguem introduït en TFG Tokens, si tal com s'ha descrit abans es premes el botó verd aquests dos elements s'intercanviarien per donar lloc al procés de venda de TFG Tokens. També pot observar-se a la imatge que en aquesta fase del projecte el "Exhange rate"que determinarà les quantitats que s'intercanvien es fix, més endavant amb els Oracles aquest deixarà de ser fix.

Fig. 3: Elements de l'exchange dintre de la interfície

Un cop repassada la interfície veurem com s'utilitza Web3.js per comunicar-se amb la blockchain, en aquesta fase del projecte, segueix sent Ganache (blockchain local).

Llavors en aquest punt, per a poder realitzar-se aquesta comunicació, necessitarà fer-se en dos pasos. El primer consisteix a connectar el navegador que s'estigui usant a la blockchain. Per a poder fer això, s'usa Metamask [17], aquesta és l'eina que s'utilitza per poder interactuar amb la blockchain i fer-hi les nostres transaccions. Per a poder usar Metamask correctament primer ha de connectar-se a la blockchain que interessi, en aquest cas, està al local host ja que s'està utilitzant Ganache.

El segon pas, consisteix a connectar la pròpia aplicació a la Blockchain i això es farà a través de Web3.js. En aquest pas s'aprofita que Metamask pugui utilitzar-se com a proveidor, ja que aquest últim el que fa és connectar-se a un node de la blockchain.

Per tant, es podria referir a Web3.js com a conjunt de llibreries que fan la funció de pont entre l'aplicació i la blockchain. Per a la seva implementació s'ha fet ús de la pròpia documentació que ofereix Web3.js i d'altres exemples d'aplicacions descentralitzades. [18]

Els indicis que s'han fet les crides correctes i que s'ha connectat i implementat bé, en el cas de la imatge on es presentava l'interfície de l'exchange descentralitzat, és el fet que automàticament en l'apartat de "Balance" aparegui el saldo del compte que en aquell moment estigui configurat a la wallet de Metamask. A més a més, dintre de l'aplicació web en la franja superior d'aquesta a la dreta, apareix l'adreça d'aquest compte.

El pas final en aquesta fase consistirà en que un cop fets els respectius deployments dels smart-contracts a la blockchain, utilitzant Truffle, es farà ús de les eines que proporciona Web3.js per a connectar els mètodes dels smart-contracts a la mateixa interfície, els principals en aquest cas són els mètodes de compra de tokens i venda de tokens vistos prèviament.

5.2.3 Pas de blockchain local (Ganache) a la testnet Rinkeby

Després d'haver completat el pas anterior i haver fet totes les proves convenients per assegurar-se del correcte funcionament del exchange. El pas que continua és el de canviar de blockchain tal com s'havia anticipat anteriorment que es faria.

El canvi es realitzarà entre la blockchain local que s'havia estat utilitzat fins ara (Ganache) i la testnet pública Rinkeby.

Per a poder procedir amb aquest canvi, primer es va haver de fer ús d'un faucet, concretament el Rinkeby Authenticated Faucet per a poder obtenir Ether per a fer les proves i els deployments corresponents en aquesta testnet. [19]

Un cop això, la següent tasca va consistir en preparar l'aplicació perquè en lloc que automàticament es connectés a la blockhain local, ho fes a la que interessava. Perquè això sigui així, el que s'ha de modificar concretament és un arxiu que té qualsevol projecte que estigui usant l'entorn de Truffle anomenat truffle-config.js. En aquest arxiu s'han de modificar certs paràmetres perquè l'acció es porti a terme. En aquest pas es va fer ús també de la plataforma Infura, la qual consisteix en un proveïdor d'Infraestructura com a Servei (IaaS) que ofereix una varietat de serveis i eines per als desenvolupadors de blockchain, entre aquestes es troba el servei usat per aquest projecte, la utilització de la seva API com a proveïdor a la xarxa, es va fer la creació d'un endpoint que connectes amb la Rinkeby network, I un cop creades les claus de la API, omplir els paràmetres necessaris per a poder fer la connexió. [20]

Per a poder realitzar aquesta fase es va fer ús de la documentació de Truffle i es van consultar exemples d'utilitat.

5.2.4 Utilització d' Oracles per obtenir els preus en temps real

El següent pas realitzat en aquesta etapa del projecte consisteix en la implementació dels Oracles al exchange descentralitzat actual.

La idea principal ha estat fer ús dels anomenats Data Feeds de Chainlink, aquesta última és una xarxa d'oracles que connecta smart-contracts a tota mena de dades del món real des de preus, dades esportives, dades de transport, dades climàtiques, etc. És un sistema aplicable a qualsevol blockchain i actualment és utilitzat per un nombre considerable d'organitzacions DeFi reconegudes entre les quals es troben Aave, Kyber Network, Synthetix i Ampleforth.

Llavors, els Data Feeds de Chainlink són la manera més ràpida de connectar els contractes intel·ligents amb els preus dels actius del mercat real. [21]

En el cas de l'aplicació d'aquest projecte l'ús que se'ls hi ha volgut donar és el d'agafar el preu d'Ethereum en dòlars en temps real des del smart-contract que actua com a exchange (TFGSwap.sol) per després poder modificar l'exchange rate i que això influeixi en els intercanvis realitzats dintre de l'aplicació descentralitzada.

Els càlculs han estat els següents: Primer de tot es fixa un preu per al token creat (TFG Token), en aquest cas té un preu de 65 dòlars, un cop això, es fa ús de la funció proporcionada pels Data Feeds de Chainlink (getLatestPrice) tal com s'observa en la captura del codi adjunta. Amb aquests dos valors ja pot fer-se una regla de 3 amb la que s'obté el

exchange rate que es buscava. Amb aquest valor es multiplicarà o es dividirà segons l'acció que es desitja dur a terme (comprar o vendre tokens).

A més a més, la funció es fa servir també per a actualitzar i mostrar el valor del exchange rate a l'usuari en la propia interfície.

```
function getLatestPrice() public view returns (int) {
    (
        /*uint80 roundID*/,
        int price,
        /*uint startedAt*/,
        /*uint timeStamp*/,
        /*uint80 answeredInRound*/
    ) = priceFeed.latestRoundData();
    return price/100000000;
}
```

Fig. 4: Funció utilitzada per obtenir el valor d'Ether en dolars utilitzant Data Feeds de Chainlink

El pas d'implementar els data feeds per a que funcionessin amb les instruccions de venta i de compra correctament, es va dificultar degut a un problema de la implementació que volia fer-se en un principi i les normes i limitacions del llenguatge Solidity amb els tipus de dades que es poden utilitzar a l'hora de construir smart-contracts, tot i així es va poder trobar una solució per aquest problema i tal i com mostren les imatges següents, l'objectiu s'assoleix amb èxit.

Per a la realització d'aquesta fase de l'aplicació es va fer ús de la documentació proporcionada per Chainlink respecte el ús dels Data Feeds.

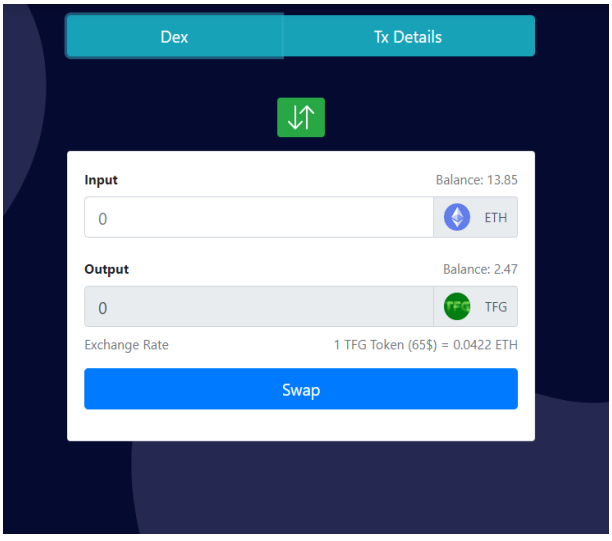

Fig. 5: Actualització de l'exchange rate exchange rate dinàmic gràcies a l'ús de Data Feeds

5.2.5 Desenvolupament del segon modul i hospedatge de l'aplicació amb Firebase Hosting

En aquesta última fase del projecte s'ha desenvolupat el segon mòdul de l'aplicació i posteriorment s'ha procedit al deployment de l'aplicació utilitzan Firebase Hosting, de manera que pugui interactuar-se amb aquesta des d'internet.

Per a poder dur a terme el segon mòdul, principalment s'ha fet ús d'una de les opcions que proporciona Web3.js. Aquesta consisteix a poder agafar l'anomenat "receipt" un cop s'ha minat la transacció realitzada i està a dins de la blockchain. El receipt, és un conjunt d'informació lligat a cada transacció, entre aquesta informació està la que s'ha decidit mostrar en la interfície d'usuari de l'aplicació: el hash de la transacció, el número de bloc on es troba la transacció, l'adreça de l'usuari, en altres paraules, qui ha realitzat la transacció, l'adreça del smart-contract amb el que s'ha operat, és a dir, l'adreça destí i el gas utilitzat en aquesta transacció.

Afegit a aquesta informació, s'ha incorporat un botó que reencamina a la visualització d'aquesta transacció des d'Etherscan. D'aquesta manera, també des d'aquesta plataforma pot visualitzar-se el codi en solidity i en bytecode del smart-contract amb el que s'ha operat.

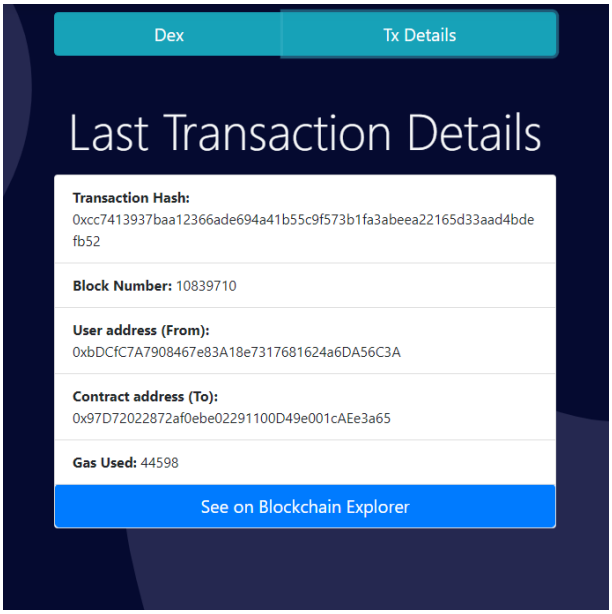

Fig. 6: Visualització de la última transacció realitzada

Finalment, s'ha realitzat el deployment de l'aplicació web utilitzant Firebase, per a fer-ho l'únic requisit era tenir un compte de Google, un cop això, se segueixen les instruccions que proporciona la pròpia documentació de Firebase per a realitzar el deployment.

6 Conclusions

En aquest apartat es discutiran les conclusions del projecte, repassant els coneixements obtinguts durant la realització d'aquest i les possibles millores que s'haguessin pogut implementar.

Durant el projecte, considero que he assolit diferents coneixements que són necessaris per a poder desenvolupar una aplicació descentralitzada sobre la Blockchain d'Ethereum.

En primer lloc, he reforçat els meus coneixements fonamentals sobre el funcionament del protocol blockchain, en aquest projecte més concretament he millorat els coneixements respecte al funcionament de la Blockchain d'Ethereum.

En segon lloc, des d'un punt de vista de desenvolupador web s'han obtingut coneixements necessaris per entendre

com una aplicació web és capaç de connectar-se i interactuar amb la Blockchain d'Ethereum a través de tecnologies com Metamask i Web3.js. Després, cal mencionar també l'aprenentatge en la experiencia de construir una aplicació descentralitzada des de zero, on durant aquest procés s'han assolit coneixements en àrees prèviament desconegudes com els entorns de desenvolupament de smart-contracts dins d'un projecte d'aquestes característiques.

En tercer lloc, pel que fa al desenvolupament web, i l'apartat del projecte que consistia a fer recerca i ús de l'eina React.js ha permès assolir també un dels objectius en termes d'aprenentatge, que consistia a poder establir unes bases sobre el funcionament del framework. Finalment, cal mencionar també tot allò que podria haver-se fet millor o d'una forma molt més òptima durant el transcurs del projecte. En aquest cas, el primer aspecte a millorar seria el comunicatiu, ja que si en lloc d'haver-me quedat encallat en alguns punts del projecte hagués demanat ajuda o possibles suggerències externes, s'haguessin pogut trobar solucions per aquests problemes molt abans i d'aquesta manera utilitzar el temps d'una forma molt més productiva.

Per acabar, també penso que hauria de millorar en l'aspecte planificatiu, ja que, si durant el projecte s'hi han hagut de fer certs canvis en la planificació, en part ha sigut a causa de no haver estat prou precís a l'hora de definir i assignar les hores per a cada tasca segons la seva complexitat.

6.1 Línies Futures

Un cop finalitzat el projecte, aquest apartat té com a objectiu comentar possibles millores respecte al treball realitzat.

En primer lloc, cal mencionar l'addició d'un mòdul de Liquidity Pools, ja que aquest va començar sent un dels objectius dins d'aquesta aplicació descentralitzada que més tard es va substituir a causa de un reajustament de la planificació del projecte per la utilització d'oracles de chainlink i pel mòdul de visualització de l'última transacció realitzada. Les Liquidity Pools són una de les tecnologies que més abunden en l'ecosistema DeFi, a més a més de constituir una part essencial dels AMM. Més concretament, es tracta d'un conjunt de fons bloquejats en un smart-contract que serveix per a crear un mercat entre dues o més criptomonedes. Perquè funcioni, en aquest fons ha d'haver-hi la mateixa quantitat en diners de les dues criptomonedes, és a dir, que mantinguin una relació 50:50. S'acostumen a utilitzar per a facilitar aspectes com el comerç i els préstecs. Dintre de les Liquidity Pools, existeixen unes figures clau que permeten que tot aquest sistema pugui portar-se a terme, aquests són els liquidity providers, que tal com diu el seu nom proveeixen de liquiditat al fons en qüestió, en altres paraules, subministren al pool un valor equivalent dels dos tokens, i en conseqüència cobraran unes comissions proporcionals a la contribució. Aquesta funcionalitat dotaria a l'aplicació de més interacció per part dels usuaris amb la blockchain d'Ethereum, a més a més d'aumentar la complexitat dels smart-contracts, ja que es treballarien conceptes com el de AMM, Automated Market Maker.

Finalment, fent autocrítica del treball presentat, una de les possibles millores a realitzar també podria ser el fet de portar molt més pes de les operacions de l'exchange als smart-contracts.

Agraïments

Vull agrair al meu tutor de treball de final de grau, Jordi Herrera, per les recomanacions i consells que m'ha anat procurant durant el transcurs del treball, a més a més de la resolució dels dubtes que m'han pogut sorgir en la realització d'aquest.

I finalment, també vull donar les gràcies a familiars i amics, pel suport moral durant aquesta etapa.

Referències

[1] Ibm.com. 2022. ¿Qué es la tecnología de blockchain? - IBM Blockchain | IBM. [online] Available at: <https://www.ibm.com/es-es/topics/what-is-blockchain> [Accessed 27 June 2022].

[2] Decentralized Finance: On Blockchain and Smart Contract Based Financial Markets. Fabian Schar. [online] Available at: <https://doi.org/10.20955/r.103.153-74> [Accessed 27 June 2022].

[3] Ethereum Explained: A Guide to the World Supercomputer - Cryptopedia Staff. [online] Available at: <https://www.gemini.com/cryptopedia/ethereum-blockchain-smart-contracts-dapps#section-ethereums-d-app-ecosystem> [Accessed 27 June 2022].

[4] Bit2Me Academy. 2022. Smart Contracts: ¿Qué son, cómo funcionan y qué aportan? - Bit2Me Academy. [online] Available at: <https://academy.bit2me.com/que-son-los-smart-contracts/> [Accessed 27 June 2022].

[5] Bit2Me Academy. 2022. ¿Qué es un Exchange Descentralizado (DEX)?. [online] Available at: <https://academy.bit2me.com/exchange-descentralizado-dex/> [Accessed 27 June 2022].

[6] (www.dw.com), D., 2022. Web3: What is it? How does it work? | DW | 14.04.2022. [online] DW.COM. Available at: <https://www.dw.com/en/web3-what-is-it-how-does-it-work/a-61362914> [Accessed 27 June 2022].

[7] APD España. 2022. Metodología Kanban: en qué consiste y cómo utilizarla. [online] Available at: <https://www.apd.es/metodologia-kanban/> [Accessed 27 June 2022].

[8] Profesional Review. 2022. Metamask: qué es y cómo configurar esta wallet. [online] Available at: <https://www.profesionalreview.com/2021/11/02/metamask-la-mejor-wallet-para-almacenar-tus-ethereum/> [Accessed 27 June 2022].

[9] BeInCrypto. 2022. ¿Qué es Truffle en Ethereum? Una guía básica - BeInCrypto. [online] Available at: <https://es.beincrypto.com/aprende/truffle-ethereum-eth-guia-basica/> [Accessed 27 June 2022].

[10] Trufflesuite.com. 2022. Truffle Documentation - Truffle Suite. [online] Available at:

<https://trufflesuite.com/docs/> [Accessed 27 June 2022].

[11] Bybit Learn. 2022. Bybit Learn | ¿Qué es Etherscan y para qué funciona?. [online] Available at: <https://learn.bybit.com/es/blockchain/que-etherscan-y-para-que-funciona/> [Accessed 27 June 2022].

[12] Firebase. 2022. Firebase Hosting | Firebase Documentation. [online] Available at: <https://firebase.google.com/docs/hosting> [Accessed 27 June 2022].

[13] Moralis » The Ultimate Web3 Development Platform. 2022. How to Create a DEX in 5 Steps » Moralis » The Ultimate Web3 Development Platform. [online] Available at: <https://moralis.io/how-to-create-a-dex-in-5-steps/> [Accessed 27 June 2022].

[14] ethereum.org. 2022. ERC-20 Token Standard | ethereum.org. [online] Available at: <https://ethereum.org/en/developers/docs/standards/tokens/erc-20/> [Accessed 27 June 2022].

[15] Es.reactjs.org. 2022. Empezando – React. [online] Available at: <https://es.reactjs.org/docs/getting-started.html> [Accessed 27 June 2022].

[16] Mark Otto, a., 2022. Introduction. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/4.1/getting-started/introduction/> [Accessed 27 June 2022].

[17] Docs.metamask.io. 2022. Introduction | MetaMask Docs. [online] Available at: <https://docs.metamask.io/guide/#why-metamask> [Accessed 27 June 2022].

[18] Web3js.readthedocs.io. 2022. web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation. [online] Available at: <https://web3js.readthedocs.io/en/v1.7.3/> [Accessed 27 June 2022].

[19] Faucet.rinkeby.io. 2022. Rinkeby: Authenticated Faucet. [online] Available at: <https://faucet.rinkeby.io/> [Accessed 27 June 2022].

[20] Docs.infura.io. 2022. Ethereum (execution layer) - Infura Docs. [online] Available at: <https://docs.infura.io/infura/networks/ethereum> [Accessed 27 June 2022].

[21] Docs.chain.link. 2022. Using Data Feeds | Chainlink Documentation. [online] Available at: <https://docs.chain.link/docs/get-the-latest-price/> [Accessed 27 June 2022].

APÈNDIX

```
pragma solidity ^0.5.0;

import "./Token.sol";
import "@chainlink/contracts/src/v0.5/interfaces/AggregatorV3Interface.sol";

contract TFGSwap {
    string public name = "TFGSwap Cryptocurrency Exchange";
    Token public token;
    uint public rate = 0;
    AggregatorV3Interface internal priceFeed;


    event TokensBought(
        address account,
        address token,
        uint amount,
        uint rate
    );

    event TokensSold(
        address account,
        address token,
        uint amount,
        uint rate
    );

    constructor(Token _token) public {
        token = _token;
        priceFeed = AggregatorV3Interface(0x8A753747A1Fa494EC906cE90E9f37563A8AF630e);
    }

    function getLatestPrice() public view returns (int) {
        (
            /*uint80 roundID*/,
            int price,
            /*uint startedAt*/,
            /*uint timeStamp*/,
            /*uint80 answeredInRound*/
        ) = priceFeed.latestRoundData();
        return price/100000000;
    }

    function buyTokens(uint n_tokensToBuy) public payable {
        // Calculate the number of tokens to buy

        // Amount of tokens
        uint tokenAmount = n_tokensToBuy;

        // To be sure we have enough tokens to do the operation
        require(token.balanceOf(address(this)) >= tokenAmount);

        // Transfer tokens to the user
        token.transfer(msg.sender, tokenAmount);

        // Emit an event
        emit TokensBought(msg.sender, address(token), tokenAmount, rate);
    }

    function sellTokens(uint _amount, uint _etherToRedeem) public {
        // Users can't sell more tokens than they have
        require(token.balanceOf(msg.sender) >= _amount);

        // Calculate the amount of Ether to redeem
        uint etherAmount = _etherToRedeem;

        // To be sure we have enough Ether to do the operation
        require(address(this).balance >= etherAmount);

        // Perform sale
        token.transferFrom(msg.sender, address(this), _amount);
        msg.sender.transfer(etherAmount);

        emit TokensSold(msg.sender, address(token), _amount, rate);
    }

}
```

Listing 1: Smart Contract TFGSwap.sol

```
pragma solidity ^0.5.0;

contract Token {
    string  public name = "TFG Token";
    string  public symbol = "TFG";
    uint256 public totalSupply =
    1000000000000000000000000; // 1 million
    tokens
    uint8   public decimals = 18;


    event Transfer(
        address indexed _from,
        address indexed _to,
        uint256 _value
    );

    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint256 _value
    );

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


    constructor() public {
        balanceOf[msg.sender] = totalSupply;
    }

    function transfer(address _to, uint256 _value
    ) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;
        emit Transfer(msg.sender, _to, _value);
        return true;
    }

    function approve(address _spender, uint256
    _value) public returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender,
    _value);
        return true;
    }


    function transferFrom(address _from, address
    _to, uint256 _value) public returns (bool
    success) {
        require(_value <= balanceOf[_from]);
        require(_value <= allowance[_from][msg.
    sender]);
        balanceOf[_from] -= _value;
        balanceOf[_to] += _value;
        allowance[_from][msg.sender] -= _value;
        emit Transfer(_from, _to, _value);
        return true;
    }
}
```

Listing 2: Smart Contract Token.sol