UNIVERSITAT POLITÈCNICA DE CATALUNYA

SOLIDITY

Treball dirigit

LLENGUATGES DE PROGRAMACIÓ

Hash: 7674

INDEX

1. QUÈ ÉS SOLIDITY ? ……..3

1.1. Propòsit del llenguatge……..3

1.2. Què és un “smart contract” ?.....3

1.2.1. Beneficis……4

1.3. Què és una blockchain ?.....4

1.3.1. Ethereum (blockchain principal de Solidity).....5

1.3.2. Altres plataformes blockchain compatibles amb Solidity……..6

1.3.3. Blockchain vs Bitcoin (no són el mateix).....6

2. HISTÒRIA……6

3. APLICACIONS DEL LLENGUATGE (aplicacions d’un “smart contract”).....7

4. INFLUÈNCIES……8

5. SISTEMA D’EXECUCIÓ……9

6. SISTEMA DE TIPUS……9

7. PARADIGMES……9

8. EXEMPLES……..10

8.1. Smart Contract: SimpleStorage……10

8.2. Smart Contract: Coin ……11

9. FUTUR DEL LLENGUATGE……..13

10. Referències (segona part).....14

1. QUÈ ÉS SOLIDITY ?

1.1 Propòsit del llenguatge

Solidity és un llenguatge de programació relativament nou, el qual està avançant i guanyant popularitat a gran velocitat, del qual actualment se'n fa una release nova de forma regular cada 2-3 setmanes.

Solidity té el propòsit, i per tant, s'usa, per implementar "smart contracts", en varies plataformes de blockchain.

1.2 Què és un "smart contract" ?

El terme "smart contract" va ser usat per primera vegada l'any 1997, per Nick Szabo, el qual el definia així: un "smart contract" és un conjunt de promeses, especificades en forma digital, inclosos els protocols en què les parts compleixen aquestes promeses.

Actualment el terme "smart contract" fa referència a un programa d'ordinador, el qual té la intenció d'automàticament executar-se, i controlar legalment accions i també fets, segons els termes d'un contracte, o un acord.

Per tant, un "smart contract" és el mateix que un contracte en el món real, l'única diferència és que aquests són completament digitals. Un "smart contract" sempre està guardat dins d'una blockchain (més endavant s'explica que és una blockchain).

Per exemple, una recol·lecció de fons pel finançament d'un projecte (és a dir, la pràctica de crowdfunding) es podria realitzar mitjançant un "smart contract", ja que al cap i a la fi és un contracte entre les persones que donen els diners, i la persona / persones que el reben, per a assolir un determinat objectiu.

En aquest exemple, les persones que volen finançar el projecte transfereixen els diners al "smart contract". Si el projecte finalment queda totalment finançat, el contracte ("smart contract") automàticament passa tots els diners al creador/creadors del projecte. Però, en el cas que per exemple el projecte no aconsegueixi el seu objectiu, llavors automàticament l'smart contract retorna els diners a les persones que principalment els havien donat, tal les regles del contracte estableixen.

D'aquesta manera, no hi ha ningú que estigui al control dels diners, que no sigui el contracte.

1.2.1 Beneficis

Els principals beneficis d'un smart contract són:

- **Confiança i transparència:** El fet que no hi ha cap tercer implicat i com que els registres xifrats de les transaccions es comparteixen entre els participants, el contracte no es pot modificar per al benefici personal d'una persona.

  A més, a part que els contractes són distribuïts entre tots els participants, aquests també són inmutables. Això vol dir que, una vegada un smart contract s'ha creat, no pot tornar-se a canviar, i per tant ningú pot fer trampa.

- **Seguretat:** Els registres de transaccions de Blockchain estan xifrats, cosa que els fa molt difícils de piratejar. A més, com que cada registre està connectat per registres anteriors i posteriors, per intentar piratejar un sol registre, s'hauria de fer canviar tota la cadena de registres.

- **Velocitat, eficiència i precisió:** Un cop es compleix una condició, el contracte s'executa immediatament. Com que els contractes intel·ligents són digitals i automatitzats, no hi ha tràmits per processar ni temps dedicat a conciliar els errors que sovint es deriven de l'ompliment manual de documents.

- **Estalvis:** Els contractes intel·ligents eliminen la necessitat que els intermediaris gestionin les transaccions i, per tant, tots els retards.

1.3. Què és una blockchain?

Blockchain és un "llibre major" (és a dir, com una espècia de base de dades) compartit i immutable que facilita el procés d'enregistrament de transaccions i el seguiment dels actius en una xarxa. Per tant, una blockchain enregistra "smart contracts".

Un actiu pot ser tangible (una casa, cotxe, efectiu, terreny) o intangible (propietat intel·lectual, patents, drets d'autor, marca). Pràcticament qualsevol cosa de valor es pot

rastrejar i comerciar en una xarxa blockchain, reduint el risc i reduint costos per a tots els implicats.

La manera en que una blockchain funciona, és aquella en que a mesura que es produeix cada transacció, es registra com un "bloc" de dades. A més, cada bloc està connectat amb els anteriors i posteriors. I les transaccions es bloquegen juntes en una cadena irreversible: una cadena de blocs.

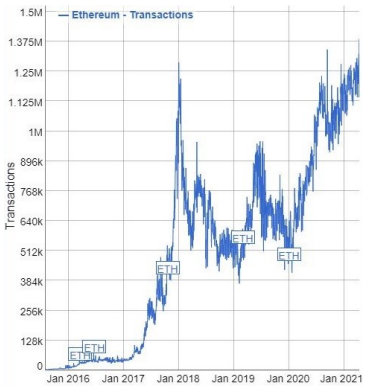
Cal destacar que una xarxa blockchain pot ser de diversos tipus: píublica, privada...

1.3.1. Ethereum (blockchain principal de Solidity)

La xarxa blockchain per la qual es va dissenyar que treballés Solidity és Ethereum. Per tant, Solidity acostuma a treballar amb Ethereum, però això no vol dir que no sigui compatible amb altres xarxes blockchain.

Ethereum és una xarxa blockchain creada específicament per treballar amb smart contracts. Ethereum és una cadena de blocs descentralitzada i de codi obert, i va ser creada l'any 2013 pel programador Vitalik Buterin (junt amb altres).

Al ser de codi obert, la plataforma permet a qualsevol desplegar-hi aplicacions descentralitzades permanents i immutables, amb les quals els usuaris poden interactuar.



Nombre de transaccions diàries confirmades a Ethereum des de l'abril de 2021

Ethereum és una xarxa d'ordinadors (nodes) no jeràrquica que es construeix sobre una sèrie de "blocs" o lots de transaccions cada cop més gran. Cada bloc conté un identificador de la cadena que l'ha de precedir si el bloc es considera vàlid. Sempre que un node afegeix un bloc a la seva cadena, executa les transaccions en el seu ordre.

Aquests equilibris i valors, coneguts col·lectivament com l'estat, es mantenen a l'ordinador del node per separat de la cadena de blocs.

Cada node es comunica amb un subconjunt relativament petit de la xarxa, que són els peers de cada node.

1.3.2. Altres plataformes blockchain compatibles amb Solidity

Solidity, però, també és compatible amb altres xarxes de blockchain. Per exemple:

- Tron.
- Hedera Hashgraph.
- Binance Smart Chain.
- O Avalanche.

1.3.3. Blockchain vs Bitcoin (no són el mateix)

Un cop una persona llegeix la definició del que és una blockchain, pot pensar que aquesta és el mateix que el Bitcoin. No obstant, es tracta de coses diferents. Les diferències són:

- Bitcoin és una criptomoneda, mentre que blockchain és una base de dades distribuïda.
- Bitcoin promou l'anonimat, mentre que blockchain tracta de transparència, és a dir, que la cadena de blocs ha de complir totes les regles del contracte.
- Bitcoin transfereix moneda entre usuaris, mentre una xarxa blockchain es pot utilitzar per transferir tot allò que es pugui fer en un contracte.

2. HISTÒRIA

Solidity, al ser un llenguatge dissenyat per Ethereum, es va crear junt amb aquest. Per tant, Solidity va ser creat per un dels creadors de Ethereum, Gavin Wood. És d'importància dir que, SWIFT ha desplegat una prova de concepte utilitzant Solidity que s'executa amb Burrow.

La raó principal per la qual es va crear el llenguatge, i Ethereum, és amb l'argument que bitcoin i les blockchains podrien beneficiar-se d'altres aplicacions a més dels diners i

necessitava un llenguatge més sòlid per al desenvolupament d'aplicacions que pogués portar a connectar actius del món real, com ara accions i propietats, a la blockchain.

3. APLICACIONS DEL LLENGUATGE (aplicacions d'un "smart contract")

Com ja explicat a l'apartat que explica el que són "smart contracts", tot allò que es pugui establir en un contracte físic, també pot ser establert en un contracte digital, per tant, en un "smart contract", i per tant amb el llenguatge del qual estem parlant, Solidity.

No obstant, algunes de les aplicacions més usades pels "smarts contracts" i per tant per Solidity, són:

- **Per bancs:** préstecs, pagaments automàtics...
- Processar reclamacions.
- Enviaments i pagaments, entre persones, empreses...
- Per votacions.
- carteres multisignament.
- Crowdfunding.
- **Vetllar per l'eficàcia dels medicaments:** Sonoco i IBM estan treballant per reduir els problemes en el transport de medicaments augmentant la transparència de la cadena de subministrament.
- **Augment de la confiança en les relacions entre minoristes-proveïdors:** Home Depot (la cadena de botigues de bricolatge més gran que hi ha als Estats Units) utilitza contractes intel·ligents a blockchain per resoldre ràpidament disputes amb venedors. Mitjançant la comunicació en temps real i una major visibilitat de la cadena de subministrament, estan construint relacions més sòlides amb els proveïdors, donant lloc a més temps per al treball crític i la innovació.
- **Fer el comerç internacional més ràpid i eficient**: Per exemple, amb l'empresa **we.trade**.

4. INFLUÈNCIES

Solidity és un llenguatge que fa ús de parèntesis (curly-bracket), i el qual la seva creació ha estat influenciada per altres llenguatges. Alguns d'aquests llenguatges:

- **JavaScript:** Quan Solidity es va crear, aquest es va influencia notablement en JavaScript. Això es devia a que Solidity tenia l'scope de les variables a nivell de funció. I a més, també feia ús de la keyword `var`. .

  Cal destacar que aquesta influència s'ha reduit des de la versió 0.4.0, ja que ara Solidity comparteix moltes menys coses que JavaScript, cosa que fa que Solidity ara s'assembli a la majoria dels altres llenguatges de programació que usen parèntesis, i no tingui aquesta influència tant gran amb JavaScript.

  Algunes de les coses, però, que segueix compartint amb JavaScript, són per exemple la manera en que es declaren les funcions: amb `function`.

- **C++:** Actualment el llenguatge amb el qual Solidity és més influenciat, és amb C++.

  Aquesta influència es pot observar en la sintaxi de les declaracions de variables, els bucles, el concepte de funcions de sobrecàrrega, les conversions de tipus implícites i explícites i molts altres detalls.

- **Python:** Els modificadors de Solidity es van afegir intentant modelar els decoradors de Python amb una funcionalitat molt més restringida. A més, l'herència múltiple, i la paraula `super` han sigut extretes del llenguatge Python, així com l'assignació general i la semàntica de còpia dels tipus de valor i referència.

- **Altres**

5. SISTEMA D'EXECUCIÓ

Pel que fa al sistema d'execució, Solidity vindria a ser com Java, és a dir, un llenguatge que és compila, i després s'interpreta en una màquina virtual.

Solidity, per tant, podriem dir que, pel que seria el sistema d'execució, utilitza un tipus de sistema **mixt**, ja que es compilat però també interpretat.

La màquina virtual que utilitza Solidity té el nom: Ethereum Virtual Machine.

Ethereum Virtual Machine és l'entorn d'execució dels contractes intel·ligents a Ethereum. Aquesta màquina virtual està totalment aïllada, el que significa que el codi que s'executa dins de la màquina no té accés a la xarxa, el sistema de fitxers o altres processos. Els smart contracts fins i tot tenen accés limitat a altres contractes intel·ligents.

6. SISTEMA DE TIPUS

Seguint les influències de C++, per una banda Solidity és **weakly typed**, ja que té algunes regles "màgiques" que permeten que es puguin convertir els tipus automàticament.

I, també, Solidarity és de tipat stàtic, ja que el tipus de les variables del programa són conegudes en temps de compilació, de forma que la comprovació de tots els tipus de les variables es fa en compilació.

7. PARADIGMES

Seguint també les seves influències, sobretot de Python i C++, podem dir que Solidity és un llenguatge multiparadigma.

Es pot considerar, un llenguatge **Imperatiu**, el qual està **orientat a objectes**, i que és també **funcional**, ja que es té la possibilitat de crear funcions.

8. EXEMPLES

A continuació es mostra un exemple molt senzill d'un smart contract, escrit amb Solidity:

8.1 Exemple smart contract: SimpleStorage

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

Llavors, veiem el que s'està fent en aquest exemple:

A la primera línia de codi (si no tractem el comentari com una línia), el que estem fent és especificar per a quines versions de Solidity està escrit el codi. En aquest cas, estem dient que el codi serà compatible per totes aquelles versions que siguin la "0.4.16" o superiors, fins la "0.9.0", aquesta última, però, no hi és inclosa.

A continuació, podem observar la declaració de la colecció de codi que conté un contracte, el contracte "SimpleStorage". En Solidity un contracte és una colecció (com la que estem observant), en que hi ha funcions i dades (les quals són l'estat). Tot això està guardat en una posició específica de la blockchain Ethereum.

La línia `uint storedData;` declara una variable d'estat anomenada `storedData` de tipus `uint` (número enter sense signe de 256 bits). En aquest exemple, el contracte defineix les funcions set i get que es poden utilitzar per modificar o recuperar el valor de la variable.

Aquest contracte no fa res més que permetre a qualsevol (a la xarxa ethereum) emmagatzemar un únic número accessible per qualsevol persona del món. Qualsevol persona podria tornar a trucar a set amb un valor diferent i sobreescriure el número, però tot i això, el

número encara s'emmagatzemaria a l'historial de la cadena de blocs de la blockchain Ethereum.

Cal destacar, que, per accedir a un membre (com una variable d'estat), no sempre cal afegir `this.` Per exemple, en el cas de `storedData` ho podem fer directament.

8.2 Exemple smart contract: SimpleStorage

Aquest, és un exemple una mica més complex que l'anterior. Es tractaria d'un contracte molt simple de transaccions de criptomonedes:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.4;

contract Coin {
    // The keyword "public" makes variables
    // accessible from other contracts
    address public minter;
    mapping (address => uint) public balances;

    // Events allow clients to react to specific
    // contract changes you declare
    event Sent(address from, address to, uint amount);

    // Constructor code is only run when the contract
    // is created
    constructor() {
        minter = msg.sender;
    }

    // Sends an amount of newly created coins to an address
    // Can only be called by the contract creator
    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    // Errors allow you to provide information about
    // why an operation failed. They are returned
    // to the caller of the function.
    error InsufficientBalance(uint requested, uint available);

    // Sends an amount of existing coins
    // from any caller to an address
    function send(address receiver, uint amount) public {
        if (amount > balances[msg.sender])
            revert InsufficientBalance({
                requested: amount,
                available: balances[msg.sender]
            });

        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Per començar, podem verue que la línia `address public minter;` declara una variable estat de tipus `address`. `address` és un valor de 160 bits que no permet cap operació aritmètica.

La paraula `public` és molt semblant al significat en els altres llenguatges. En aquest, ens serveix perquè altres contractes puguin accedir a la variable.

A continuació, a la següent línia, a: `mapping (address => uint) public balances;` el que estem fent és mapejar adresses en integers sense signe.

La línia `event Sent(address from, address to, uint amount);` declara un event el qual serà emès a la última línia de la funció `send` . Clients de la xarxa Ethereum (com per exemple aplicacions web) poden escoltar aquests events. Un cop es rebin, això es farà amb els paràmetres `from`, `to` i `amount` fent que es puguin produir les transaccions.

La constructora és una funció especial que s'executa durant la creació del contracte i que no es pot cridar després. En aquest cas, emmagatzema permanentment l'adreça de la persona que crea el contracte.

Les funcions que poden utilitzar els usuaris i també altres contractes per realitzar transaccions son `mint` i `send` .

La funció `mint` envia una quantitat de monedes noves a una altre adreça. Però, per la línia `require(msg.sender == minter);` això només es podrà produir en el cas que qui cridi la funció sigui també el creador del contracte.

El funcionament de la paraula: `error` és bastant semblant a `require` , però a diferència d'aquesta, es permet posar nom als errors i retornar dades adicionals per a que sigui més fàcil identificar què és el que ha passat quan ha saltat error.

Per acabar, la funció `send` pot ser usada per a tothom, per a enviar monedes (si es que té la quantitat suficient que vol enviar). Sino, saltarà l'error que correspon.

9. FUTUR DEL LLENGUATGE

Per acabar, és important destacar que Solidity és un llenguatge que li espera una evolució molt bona. No tan sols per la gran comunitat que té, i perquè rep actualitzacions cada molt poc temps, sinó perquè és un llenguatge que està totalment lligat a la blockchain Ethereum.

Ethereum actualment és de les blockchains més grans que hi ha. La seva criptomoneda, **Ether**, és la segona més gran actualment en el mercat de capitals (la primera és la criptomoneda Bitcoin), i tot apunta que no pararà de créixer, pels grans beneficis que la xarxa blockchain pot aportar.

A més, actualment s'està treballant en una gran actualització de Ethereum, la qual comportarà gran quantitat de beneficis. Aquesta actualitació tindrà el nom de **Ethereum 2.0**, o també és dirà **Serenity**. El principal objectiu que té aquesta actualització és incrementar l'ample de banda de les transaccions a la xarxa de la blockchain: de 15 transaccions per segon a desenes de milers.

10. REFERENCIES

n.d. docs.soliditylang. Language Influences.

n.d. Solidity Programming Language | The Solidity language portal is a comprehensive information page for the Solidity programming language. It features documentation, binaries, blog, resources & more. Solidity.

n.d. Solidity — Solidity 0.8.10 documentation. Solidity.

n.d. Avalanche Docs: What is Avalanche? Avalanche Docs.

n.d. we.trade - more trust. more trade. We.Trade.

Afshar, Vala, and Brett Colbert. 2017. "Ethereum Is The Second Most Valuable Digital Currency, Behind Bitcoin." HuffPost. Ethereum Is The Second Most Valuable Digital Currency, Behind Bitcoin | HuffPost null.

"binance-chain/bsc: A Binance Smart Chain client based on the go-ethereum fork." n.d. GitHub. https://github.com/binance-chain/bsc.

"Blockchain Explained: The difference between blockchain and Bitcoin." n.d. Euromoney. The difference between blockchain and Bitcoin | Euromoney Learning.

Carlson, Ben. 2021. "Ethereum price: Why Ether is taking off." Fortune. There are two very real reasons Ethereum is taking off.

"Ethereum in 2020: Vitalik telegraphs high hopes in Denver." 2020. Fortune. Ethereum, Bitcoin's closest rival, faces its moment of truth.

"ethereum/solidity: Solidity, the Smart Contract Programming Language." n.d. GitHub. GitHub - ethereum/solidity: Solidity, the Smart Contract Programming Language.

"Hyperledger Burrow." n.d. Hyperledger. Hyperledger Burrow.

"Introduction to smart contracts | ethereum.org." n.d. Ethereum.org. Introduction to smart contracts | ethereum.org.

"Introduction to Smart Contracts — Solidity 0.8.10 documentation." n.d. Solidity. Introduction to Smart Contracts — Solidity 0.8.10 documentation.

Laurence, Tiana. 2017. *Blockchain For Dummies*. N.p.: Wiley.

Marr, Bernard. n.d. "What is the Difference Between Blockchain And Bitcoin?" Bernard Marr. What is the Difference Between Blockchain And Bitcoin? | Bernard Marr.

Popper, Nathaniel. 2017. "Move Over, Bitcoin. Ether Is the Digital Currency of the Moment. (Published 2017)." The New York Times. Move Over, Bitcoin. Ether Is the Digital Currency of the Moment. - The New York Times.

"Smart contracts - Simply Explained." 2017. YouTube. Smart contracts - Simply Explained.

"What are smart contracts on blockchain?" n.d. IBM. What are smart contracts on blockchain? | IBM.

"What is Blockchain Technology? - IBM Blockchain." n.d. IBM. What is Blockchain Technology?.