Block-Degree

Damiano Nardi

Contents

1 Introduzione

Si vuole creare una piattaforma pubblica per certificare la laurea degli studenti da parte di tutti i membri della commissione. Per farlo si è scelto di creare uno smart-contract su piattaforma Ethereium che si occupa di tutta la parte logica della certificazione

2 Requisiti

2.1 Creazione di certificati

Solo il Coordinatore del corso può creare nuovi certificati e deve poter inserire

- la commissione sotto forma di indirizzi delle chiavi pubbliche dei docenti
- il codice fiscale, nome e cognome del laureando
- nome e cognome del relatore

2.2 Firma di certificati

Un certificato viene considerato valido solo se tutti i membri della commissione lo hanno firmato.

- Ogni docente può visualizzare la lista di certificati che ha firmato e quelli da firmare
- Il docente può decidere quali certificati firmare
- Chi ha firmato e chi non ha fatto è pubblico

2.3 Visualizzazione dei certificati

Chiunque attraverso il riferimento del certificato può visualizzarlo e può vedere tutte le informazioni contenute al suo interno compreso chi ha firmato e chi non lo ha fatto

3 Implementazione Smart contract

3.1 Solidity

Lo smart contract è implementato in Solidity, Uno smart contract è una raccolta di funzioni e dati (il suo stato) che risiede a un indirizzo specifico sulla blockchain di Ethereum, dove le funzioni implementate possono vedere e modificare lo stato e applicare anche restrizioni a chi può modificare lo stato del contratto.

La struttura dello smart contract si divide in due contratti "DegreeBlock" e "CreateDegree".

CreateDegree viene istanziato quando viene fatto il deploy sulla blockchan mentre

DegreeBlock viene istanziato da CreateDegree quando si crea un nuovo certificato.

vediamoli più nel dettaglio

3.2 DegreeBlock

DegreeBlock ha i seguenti attributi

- **string laurea**; viene memorizzato il nome della laura certifiacata
- **string nomeLaureando**; viene memorizzato il nome e cognome del laureando
- **string codiceFiscaleLaureando** viene memorizzato il codice fiscale del laureando;
- **string nomeRealatore**; viene memorizzato il nome e cognome del relatore
- **address [] commisione**; lista degli indirizzi pubblici della commissione
- **uint256 timestampCreation**; timestamp della creazione
- **mapping (address => bool) hasSigned**; questo mapping ci permette di verificare se un determinato indirizzo di un professore ha firmato il contratto attraverso un metodo. c'è anche un altro metodo che permette di impostare il mapping del docente a true

Poi abbiamo i rispettivi metodi che permettono di accedere e modificare le variabili appena viste, ne vediamo uno di questi: sign() permette al docente di firmare solo se è nella lista commissione e se non lo ha fatto in precedenza

```
function sign() public {
                require(is_in_array(msg.sender, commisione),
                "Il professore non e' nella commisione");

                require(!hasSigned[msg.sender],
                "Il professore ha gia firmato il certificato");
                hasSigned[msg.sender]=true;
}
```

3.3 CreateDegree

CreateDegree ha i seguenti attributi

- **mapping (address => string) nomeCognome** è un mapping per associare ad ogni indirizzo di un docente un nome e cognome sarà compito del docente impostare il suo nome e cognome;
- **mapping (address => bool) isIstazied** è un mapping tra indirizzo contratto e bool e tiene traccia se un determinato contratto DegreeBlock è stato istanziato dal cordinatoreCorso

- **address cordinatoreCorso** viene memorizatto l'indirizzo del coordinatore corso ed l'unico che può chiamare un metodo che istanza un nuovo contratto DegreeBlock

questo è il metodo che permette di istanziare un nuovo certificato

```
        function crea_certificato_di_laurea(
                                        string memory laurea,
                                        string memory nomeLaureando,
                                        string memory codiceFiscaleLaureando,
                                        string memory nomeRealatore,
                                        address[] memory commisione
                                                ) public returns ( address
){

                require(msg.sender == cordinatoreCorso,
                                "Solo il cordinatore del corso
                                puo creare un certificato di laura");

                uint256 timestamp=  block.timestamp;
                DegreeBlock degree= new DegreeBlock(
                                        laurea,
                                        nomeLaureando,
                                        codiceFiscaleLaureando,
                                        nomeRealatore,
                                        timestamp,
                                        commisione
                );

                emit certificato_creato(degree, laurea,timestamp,commisione);
                is_istazied[address(degree)]=true;
                return address(degree);

        }
```

4 Frontend

Il frontend è stato implementato con React-js che permete di creare pagine web dinamiche senza necessità di ricaricare la pagina e permette di avere una gestione dello stato interno della pagina che viene modificata dinamicamente quando questo cambia.

4.1 Interfaccia con la blockchain

4.1.1 metamask

il front-end per interfacciarsi con la blockchain ha bisogno di una estenzione nel browser come per esempio metamask[1] questa si occupa della gestione delle chiavi pubbliche e private e di firmare le transazioni.

4.1.2 intefaccia con lo smartcontract

quando facciamo il deploy del contratto sulla blockchain viene creata anche un file con l'indirizzo e tutte le interfacce delle contratto. Attraverso questo file il frontend riesce ad chiamare le funzione definite nello smartcontract.

5 Deploy

Il deploy del contratto è stato fatto sulla blockchain di test ropsten dove è possibile ottenere token gratuitamente. Il frontend invece è un sito statico dove servito da git-hub pages[2] e si interfaccia direttamente con la blockchain ropsten

5.1 Istruzioni per il deploy del contratto su Blockchain di test Ropsten

5.1.1 installazione delle dipendenze

Per prima cosa posizionarsi nella cartella `cd ./smartcontract/degreeblock`
installare truffle `npm install -g truffle`
Eseguire il comando `npm install` per installare tutte le dipendenze

5.1.2 Configurazione dell deploy

1. Aprire il file `truffle-config.js`
2. modificare la variabile MNEMONIC con il propria chiave privata
3. modificare la variabile ETH-NODE con il proprio nodo Ethereum
   nel mio caso userò i nodi Ethereum messi a disposizione da infura[3]

5.1.3 Deploy del contratto

1. eseguire il comando `truffle deploy --network ropsten`
2. copiare l'interfaccia del contratto nel frontend `cp cp -r ./build/contracs/ ../../frontend/degree-block/src/`

Nell'interfaccia del contratto ci sono tutti i nomi dei metodi e delle varibili del contratto oltre che l'indirzzo in cui è stato istanziato, L'interfaccia permette al front-end implementato in javascript di iterfacciarsi con il contratto e chiamare i suoi metodi.

[1] <https://metamask.io/>
[2] <https://naddi96.github.io/degree-certificate-but-on-blockchain/>
[3] <https://infura.io/>

5.2 Istruzioni per il deploy frontend

5.2.1 installazione delle dipendenze

Per prima cosa posizionarsi nella cartella `cd ./frontend/degree-block`
Eseguire il comando `npm install` per installare tutte le dipendenze

5.2.2 Deploy frontend

1. eseguire il comando `npm run build`
2. copiare la cartella `./build/static` nel proprio webserver (es. Apache,NGNX)
3. installare il wallet metamask nel proprio browser e andare sull'indirizzo del webserver.