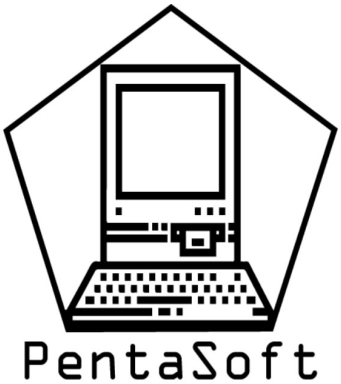

Verbale esterno del 2023-03-07
Progetto Trustify

pentasoftswe@gmail.com

Informazioni sul documento

Responsabile	Pietro Lauriola
Redattori	Marco Rosin
Verificatori	Marco Brugin
Uso	Interno/Esterno
Destinatari	Gruppo *PentaSoft*

Sommario
Vengono riportati gli argomenti discussi durante la riunione del 7 Marzo 2023 con l'azienda Sync Lab S.r.L.

Contenuti

1 Generale

1.1 Informazioni sulla riunione

- **Luogo**: Videochiamata Google Meet;
- **Ora di inizio**: 13:30;
- **Ora di fine**: 14:30;
- **Partecipanti Interni**: Marco Rosin, Pietro Lauriola, Marco Brugin, Luca Marcato, Stefano Meneguzzo, Nicola Lazzarin;
- **Partecipanti Esterni**: Fabio Pallaro, Matteo Galvagni;

1.2 Ordine del giorno

L'incontro effettuato è servito per discutere ed espandere i seguenti argomenti:

- Approfondimento e discussione con la proponente riguardo il funzionamento degli *smart contract*$_{\mathbf{G}}$;
- Consigli su metodologie e strumenti da utilizzare per testare le varie parti del prodotto da sviluppare.

2 Svolgimento

1. **Costi di esercizio della *blockchain***

   D: Come "salvare" le recensioni nella *blockchain*$_{\mathbf{G}}$/*smart contract*$_{\mathbf{G}}$ in modo da poterle recuperare in seguito? È corretto che il costo di ogni transazione sia così alto?

   R: Il costo dipende da più fattori, tra cui la rete su cui si sceglie di sviluppare il contratto, dalla memoria occupata e come viene occupata, da quale tipo di memoria viene utilizzata sulla EVM, dal tipo di operazione ecc. Ovviamente nulla vieta di usare una rete più economica durante lo sviluppo del contratto.

2. **Trasferimento dei *token***

   D: Come si svolge il trasferimento dei *token*$_{\mathbf{G}}$ da un *wallet*$_{\mathbf{G}}$ allo *smart contract*?

   R: Il trasferimento dei *token* è composto di due funzioni principali:
   - `approve()`: il proprietario del *wallet* "autorizza" il nostro contratto a prelevare dal proprio *wallet* un quantitativo di *token* richiesto dallo *smart contract*.
   - `transferTo()`: funzione che effettua il trasferimento dei *token* dal *wallet* dell'utente verso lo *smart contract*.

   NB: È possibile richiedere (e il proprietario del *wallet* può accettare) il trasferimento di una quantità di *token* diversa (possibilmente maggiore) dall'attuale bilancio di un *wallet*. Sarà nostro obbligo controllare che lo *smart contract* prelevi esattamente la quantità di *token* pattuiti e verificare tramite funzione aggiuntiva che il metodo `Approve` sia avvenuto correttamente.

3. **Verifica dello *smart contract***

   D: Avreste suggerimenti/consigli/strumenti da consigliarci per testare lo *smart contract*? Al momento stiamo valutando *Truffle*$_{\mathbf{G}}$.

   R: Gli strumenti di verifica più diffusi per *web3*$_{\mathbf{G}}$ sono *Truffle*$_{\mathbf{G}}$ e *HardHat*$_{\mathbf{G}}$.

   Entrambi forniscono le seguenti funzionalità:

   - Creazione di una cartella "test" nella radice del progetto nella quale inserire i file contenenti il codice dei test;
   - *Dependency Injection*$_{\mathbf{G}}$ automatizzata delle librerie necessarie per l'esecuzione dei test;
   - Salvataggio dello stato della *blockchain* durante l'esecuzione dei test, in modo da poter utilizzare lo stato modificato della *blockchain* come precondizione di altri test.

   Inoltre, a differenza di *HardHat*, *Truffle* può calcolare automaticamente la percentuale di codice coperto dai test.

   NB: Non è richiesto testare il codice del *token* da voi creato in quanto:

   - andrebbero testati solamente i `approve()` e `transferTo()`;
   - se il progetto venisse fornito al pubblico si userebbe uno dei *token* già sviluppato e verificati invece di svilupparne uno proprio.

4. **Verifica della *web app***

   D: Avreste suggerimenti/consigli/strumenti da consigliarci per testare la *web app*$_{\mathbf{G}}$?

   R: Analogamente a quanto detto riguardo lo *Smart Contract*, *Angular*$_{\mathbf{G}}$ fornisce il proprio strumento di *testing* e calcolo automatico della copertura del codice chiamato *Jasmine*$_{\mathbf{G}}$. NB: Ovviamente non dovrete testare il codice di *Metamask*$_{\mathbf{G}}$ iniettato nell'applicazione in quanto molto difficile (se non impossibile) da sostituire con un *mock*$_{\mathbf{G}}$ appropriato.

3 Resoconto

3.1 Tracciamento delle decisioni

ID	Decisione
VE_2022_03_07-1	Pianificazione revisione architettura *smart contract*$_{\mathbf{G}}$ in seguito a feedback ottenuto.
VE_2022_03_07-2	Utilizzo del *framework*$_{\mathbf{G}}$ *Truffle*$_{\mathbf{G}}$ per testare lo *smart contract*.
VE_2022_03_07-3	Utilizzo del *framework Jasmine*$_{\mathbf{G}}$ per testare lo *smart contract*.