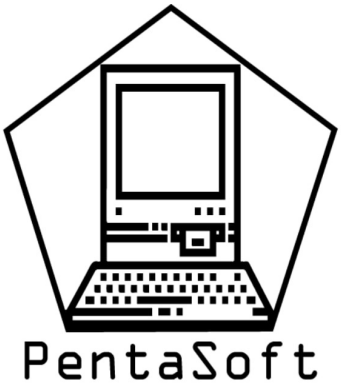

Allegato tecnico
Progetto Trustify

pentasoftswe@gmail.com

Informazioni sul documento

Responsabile	Pietro Lauriola
Redattori	Nicola Lazzarin
Verificatori	
Uso	Esterno
Destinatari	Prof. Tullio Vardanega Prof. Riccardo Cardin
Versione	*v0.0.1*

Sommario

Registro delle Modifiche

Versione	Data	Autore	Verificatore	Descrizione
0.0.1	2023/03/11	Nicola Lazzarin (Progettista)		Creazione struttura documento e stesura Introduzione

Indice

1 Introduzione

1.1 Scopo del documento

La Specifica Architetturale ha lo scopo di fornire una descrizione completa dell'architettura del prodotto software, delle tecnologie utilizzate e dei requisiti richiesti dal team di sviluppo del gruppo PentaSoft durante la fase di progettazione e codifica del prodotto. In particolare, la Specifica Architetturale include diagrammi delle classi per descrivere l'architettura e le funzionalità principali del prodotto, al fine di fornire una panoramica completa del sistema e delle sue interazioni. Inoltre, la Specifica Architetturale include una sezione dedicata ai requisiti soddisfatti dal prodotto, per consentire al team di valutare lo stato di avanzamento del lavoro e garantire il rispetto degli obiettivi prefissati.

1.2 Scopo del prodotto

Scopo del progetto è la realizzazione di una *webapp*$_{\mathbf{G}}$ che permetta di rilasciare e visualizzare recensioni certificate tramite uno *smart contract*$_{\mathbf{G}}$ risiedente in una *blockchain*$_{\mathbf{G}}$ *Ethereum*$_{\mathbf{G}}$ compatibile, al fine di minimizzare la compravendita di recensioni e il *review bombing*$_{\mathbf{G}}$.

1.3 Glossario

Alcuni dei termini utilizzati in questo documento potrebbero generare dei dubbi riguardo al loro significato, al fine di evitare tali ambiguità è necessario dar loro una definizione. Tali termini vengono contassegnati da una G maiuscola finale,se questa non compare in un titolo di sezione, a pedice della parola ed essa non verrà ripetuta più di una volta per paragrafo/sottosezione/sezione onde evitare fastidiose ripetizioni. La loro spiegazione è riportata nel *Glossario v1.0.0*

1.4 Riferimenti

1.4.1 Normativi

- ***Norme di progetto v1.0.0***
- **Regolamento del progetto didattico**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/PD02.pdf

- **Presentazione Capitolato C7 - Trustify**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Progetto/C7.pdf

1.4.2 Informativi

- ***Analisi dei requisiti v3.0.0***
- **Qualità di prodotto - slide T12 del corso di Ingegneria del Software**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T12.pdf

- **Qualità di processo - slide T13 del corso di Ingegneria del Software**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T13.pdf

- **Verifica e validazione: introduzione - slide T14 del corso Ingegneria del Software**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T14.pdf

- **Verifica e validazione: introduzione - slide T15 del corso Ingegneria del Software**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T15.pdf

- **Verifica e validazione: introduzione - slide T16 del corso Ingegneria del Software**:

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T16.pdf

2 Architettura

2.1 Breve descrizione e architettura *smart contract*$_G$ token ERC20 "TCoin"

Questo codice definisce un contratto intelligente (smart contract) sulla piattaforma Ethereum. Il contratto intelligente si chiama "TCoin" e estende il contratto "ERC20" dell'OpenZeppelin, che implementa uno standard per i token Ethereum.

Il contratto "TCoin" ha un costruttore che imposta il nome e il simbolo del token su "TCoin". Inoltre, il contratto ha una funzione pubblica chiamata "drip()" che emette (crea) 100.000 unità del token "TCoin" per l'indirizzo che ha chiamato la funzione. La funzione "drip()" fa ciò utilizzando la funzione "mint()" ereditata dal contratto "ERC20", che crea nuove unità del token e le assegna all'indirizzo specificato come primo parametro della funzione "mint()".

2.2 Breve descrizione *smart contract*$_G$ "Trustify"

Il contratto Trustify è stato scritto nel linguaggio di programmazione Solidity$_G$ e prevede un sistema di recensioni che consente agli utenti di scrivere e leggere recensioni su aziende e/o negozi identificati da un indirizzo *wallet*$_G$. Il contratto Trustify è progettato per utilizzare mappe per tenere traccia delle recensioni degli utenti e delle aziende/negozi che hanno ricevuto recensioni. Inoltre, il contratto richiede agli utenti di depositare token per poter scrivere una recensione su un'azienda/negozio specifico, e include una funzione per verificare se l'utente ha effettivamente depositato i token necessari per scrivere la recensione.

Oltre alla funzione di scrittura il contratto include funzioni per restituire un numero arbitrario (fra 1 e 25) di recensioni di un'azienda/negozio specifico oppure di un utente specifico.

2.3 Architettura *smart contract*$_G$ "Trustify"

2.3.1 Strutture dati

Il contratto definisce tre strutture dati: Review, Company e Customer.
La struttura *Review* rappresenta una singola recensione ed è composta da tre campi:

- **Review**: una stringa che contiene il testo della recensione;
- **Stars**: un valore intero che rappresenta il numero di stelle assegnate alla recensione;
- **HavePayed**: un booleano che indica se l'utente ha eseguito una translazione.

La struttura *Company* rappresenta una singola azienda o esercizio commerciale ed è composta da due campi:

- **allCustomerAddress**: un array di indirizzi Ethereum che contiene gli indirizzi degli utenti che hanno scritto una recensione per questa azienda;
- **reviewMap**: una mappa che associa ad ogni indirizzo di utente la sua recensione;

Definisce anche la struttura *Customer* che rappresenta un singolo utente ed è composta da due campi:

- **allCompanyAddress**: un array di indirizzi Ethereum che contiene gli indirizzi delle aziende o esercizi commerciali che hanno ricevuto una recensione da questo utente;
- **reviewMap**: una mappa che associa ad ogni indirizzo di azienda o esercizio commerciale la recensione dell'utente.

2.3.2 Mappe

Il contratto definisce due mappe che permettono di associare ad ogni indirizzo di azienda o esercizio commerciale la sua struttura dati Company e ad ogni indirizzo di utente la sua struttura dati Customer.

- **companyMap**: una mappa che associa ad ogni indirizzo di azienda o esercizio commerciale la sua struttura dati Company;
- **customerMap**: una mappa che associa ad ogni indirizzo di utente la sua struttura dati Customer.

2.3.3 Dipendenze

Il contratto definisce due dipendenze:

- **IERC20.sol**: l'interfaccia standard per i token ERC20;
- **SafeERC20.sol**: una libreria che definisce metodi sicuri per il trasferimento dei token ERC20;

2.3.4 Funzioni

Il contratto definisce le seguenti funzioni:

- **function CheckTransaction(address companyWalletAddress) private view returns (bool)**:
  Questa funzione è utilizzata internamente dal contratto per verificare se un utente ha già effettuato una transazione con l'azienda specificata tramite l'indirizzo companyWalletAddress. Restituisce true se l'utente ha già effettuato una transazione con l'azienda, false altrimenti.
- **modifier CheckAllowance(uint amount)**:
  Questo è un modifier che viene utilizzato su alcune delle funzioni che richiedono l'approvazione dell'utente per utilizzare il token ERC20. Verifica se l'utente ha dato il permesso al contratto di utilizzare i token ERC20 in una quantità specificata (amount). Se l'utente ha dato il permesso, allora il codice della funzione specifica verrà eseguito, altrimenti verrà restituito un errore.
- **function DepositTokens(address addressToDeposit, uint amount) public CheckAllowance(amount)**:
  Questa funzione consente agli utenti di depositare token ERC20 nel contratto. L'indirizzo del destinatario del deposito viene specificato come parametro addressToDeposit, mentre il numero di token da depositare viene specificato come parametro amount. Prima di effettuare il deposito, viene verificato che l'utente abbia dato il permesso al contratto di utilizzare i token ERC20 tramite il modifier CheckAllowance. Se l'utente ha dato il permesso, i token vengono trasferiti dal portafoglio dell'utente al portafoglio dell'azienda specificata tramite l'indirizzo addressToDeposit.
- **function WriteReview(address companyWalletAddress, string memory review, uint stars, uint amount) public CheckAllowance(amount)**:
  Questa funzione consente agli utenti di scrivere una recensione per un'azienda specificata tramite l'indirizzo addressToReview. La recensione e il numero di stelle sono specificati come parametri review e stars. Prima di scrivere la recensione, viene verificato che l'utente abbia effettuato una transazione con l'azienda tramite la funzione CheckTransaction. Se l'utente ha effettuato una transazione, la recensione viene scritta. Se l'utente ha già scritto una recensione per l'azienda, la vecchia recensione viene sovrascritta con la nuova. Viene anche aggiornato il mapping della mappa customerMap con l'indirizzo dell'azienda e la nuova recensione.

- **function GetNCompanyReview(uint start, uint end, address companyAddress) public view returns (string[] memory, uint8[] memory)**:
  La funzione GetNCompanyReview restituisce un array di stringhe contenente le review scritte da un numero specificato di clienti per una determinata azienda, insieme ad un array di numeri interi che rappresentano il numero di stelle assegnate ad ogni recensione.
  La funzione prende tre parametri in input:

  - **start** è l'indice del primo elemento dell'array di recensioni da restituire
  - **end** è l'indice dell'ultimo elemento dell'array di recensioni da restituire
  - **companyAddress** è l'indirizzo dell'azienda di cui si vogliono visualizzare le recensioni

  La funzione effettua alcuni controlli sulla validità degli indici start e end rispetto alla lunghezza dell'array delle recensioni dell'azienda specificata, e in caso di superamento dell'indice massimo, imposta l'indice massimo come valore massimo di end. Inoltre, viene controllato che l'azienda abbia almeno una recensione.

  Infine, la funzione scorre l'array di recensioni dell'azienda, selezionando le recensioni che si trovano nell'intervallo specificato dagli indici start e end, e salva la descrizione della recensione nell'array reviews e il numero di stelle assegnato nell'array stars. I due array sono poi restituiti come output della funzione.

- **function GetSpecificReview(address addressReviewed) public view returns (string memory, uint8)**:
  Questa funzione, chiamata "GetSpecificReview", è una funzione di tipo "view" che restituisce una tupla contenente una stringa e un numero intero. Questi valori rappresentano rispettivamente una recensione specifica e il numero di stelle assegnato a tale recensione. La funzione accetta un parametro "addressReviewed" che indica l'indirizzo dell'azienda per la quale si desidera ottenere una recensione specifica. La funzione utilizza poi l'indirizzo del chiamante della funzione (msg.sender) per accedere alla recensione specifica dell'azienda. La recensione e il numero di stelle assegnate sono quindi restituiti nella tupla di output. Inoltre, la funzione utilizza anche un'istruzione require per verificare che il chiamante abbia effettivamente rilasciato una recensione per l'azienda specificata. Se il numero di stelle assegnate è zero, viene restituito un messaggio di errore che indica che il chiamante non ha rilasciato alcuna recensione per l'azienda specificata.

- **function GetNMyReview(uint start,uint end) public view returns (string[] memory, uint8[] memory, address[] memory)**:
  La funzione GetNMyReview restituisce un insieme di recensioni fatte da un utente specifico. Prende in input due parametri start e end che rappresentano rispettivamente l'indice iniziale e finale delle recensioni che si vogliono ottenere. La funzione verifica che l'utente abbia effettuato almeno una recensione, che l'indice di partenza sia minore o uguale alla lunghezza totale delle recensioni e che la differenza tra l'indice finale e quello di partenza non sia superiore a 25. In caso contrario, viene sollevata una eccezione. La funzione restituisce quindi tre array contenenti le stringhe delle recensioni, le stelle assegnate e gli indirizzi delle aziende recensite nell'intervallo specificato.

- **function GetAverageStars(address addressReviewed) public view returns (uint[] memory)**:
  Questa funzione restituisce un array di valori interi contenente tutte le stelle assegnate alle recensioni ricevute da una specifica azienda, identificata dall'indirizzo 'addressReviewed'. La funzione inizia controllando la lunghezza dell'array di indirizzi degli utenti che hanno lasciato recensioni per l'azienda. Se la lunghezza è zero, significa che l'azienda non ha ricevuto alcuna recensione e

la funzione genera un errore. Altrimenti, la funzione inizializza un array di interi con la stessa lunghezza dell'array di indirizzi degli utenti, quindi, mediante un ciclo for, riempie l'array di stelle assegnate a ogni recensione. Infine, restituisce l'array di stelle.

2.3.5 Diagramma delle classi

2.3.6 *API-REST*$_G$ Trustify

- La classe "TrustifyContractReader" è una classe che permette di ottenere le recensioni di una determinata azienda attraverso l'uso di una smart contract Ethereum. In particolare, è possibile creare un'istanza della classe "TrustifyContractReader" passando come argomenti l'indirizzo dell'azienda, la posizione iniziale e finale dell'intervallo di recensioni da recuperare. La classe utilizza la libreria *Web3j* per interagire con la *blockchain Ethereum* e la libreria *Tuples*$_G$ per gestire i risultati delle transazioni. Il metodo "getReviews()" è quello che effettivamente esegue la chiamata alla smart contract e restituisce una lista di oggetti "Review".

- La classe "Review" rappresenta una recensione con le seguenti proprietà: una stringa "text" che rappresenta il testo della recensione e un intero "stars" che rappresenta il numero di stelle attribuite alla recensione. Il costruttore della classe accetta due parametri, una stringa "text" e un intero "stars", e inizializza le proprietà della classe con questi valori.

- La classe "ErrorMessage" rappresenta un messaggio di errore con la proprietà "message" che contiene il messaggio di errore. Viene usata per far sapere che errore è avvenuto durante l'esecuzione di una richiesta HTTP ritorando l'errore stesso al posto del JSON.

- La classe "AppExceptionHandler" è una classe di gestione delle eccezioni, estende "ResponseEntityExceptionHandler" e definisce un metodo che gestisce tutte le eccezioni lanciate dall'applicazione. In particolare, il metodo restituisce un oggetto "ErrorMessage" che contiene il messaggio di errore.

- La classe "SmartContractReaderController" è una classe di controller REST che definisce un'API per accedere alle recensioni tramite un'interfaccia HTTP. Il metodo "SmartContractReader" della classe si occupa di chiamare il metodo "getReviews()" della classe "TrustifyContractReader" per ottenere le recensioni.

- La classe "TrustifyRestApplication" è la classe di avvio dell'applicazione Spring Boot e contiene il metodo "main" che avvia l'applicazione.