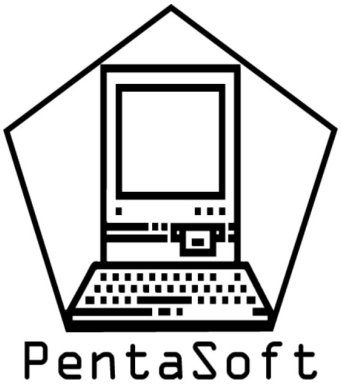


Specifica tecnica
Progetto Trustify

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 tecnica 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 tecnica 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 tecnica 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**:

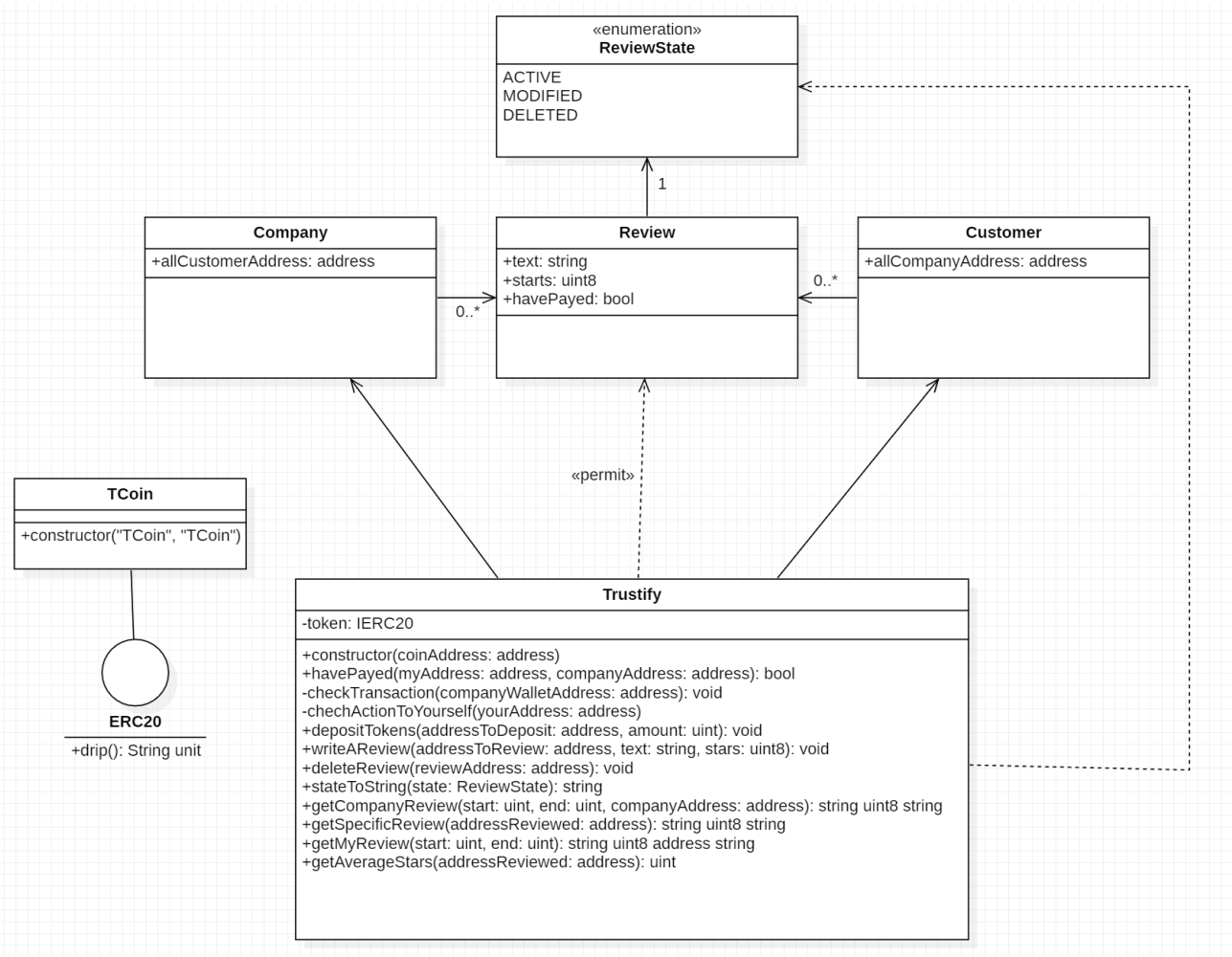
  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T16.pdf

2 Architettura

In questo paragrafo viene spiegata l'architettura e il funzionamento dei vari componenti del sistema *Trustify*.

2.1 Smart contracts

2.1.1 Diagramma delle classi



2.1.2 *TCoin*: breve descrizione e architettura

Il contratto *TCoin* è stato scritto nel linguaggio di programmazione *Solidity*$_{\mathbf{G}}$ ed estende il contratto "ERC20" della libreria *OpenZeppelin*, che implementa uno standard per i token Ethereum.

Il contratto oltre ad avere un contruttore che imposta il nome del token e il simbolo con la stringa "Trustify" possiede un'unica funzione pubblica chiamata "drip()" che emette (crea) 100.000 unità del token "TCoin" per l'indirizzo chiamante. 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()". Questo contratto "ERC20" viene usato come valuta per il pagamento delle recensioni, in quanto è possibile trasferire i token da un indirizzo ad un altro.

2.1.3 *Trustify*: breve descrizione

Il contratto *Trustify* è stato scritto nel linguaggio di programmazione *Solidity*$_{\mathbf{G}}$ 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*$_{\mathbf{G}}$. Il contratto è progettato per utilizzare delle mappe per tenere traccia delle recensioni degli utenti e delle aziende/negozi che hanno ricevuto recensioni. Inoltre, il contratto richiede agli utenti, come pagamento, il deposito di un token (nel nostro caso il token *TCoin*) per poter poi rilasciare una recensione. Oltre alla funzione di scrittura delle recensioni il contratto include funzioni di lettura di un'azienda/negozio specifico utilizzando range che vanno dalla singola recensione fino a massimo 25 per volta.

2.1.4 *Trustify*: architettura

2.1.4.1 Enumerazione Il contratto definisce un'enumerazione che rappresenta lo stato di una recensione.

- **ACTIVE**: indica che la recensione è attiva e la si può visualizzare in blockchain;
- **MODIFIED**: indica che la recensione è stata modificata almeno una volta;
- **DELETED**: indica che la recensione è stata cancellata.

Questi stati servono per far capire se le recensioni sono state modificate, eliminate oppure se sono state lasciate inalterate dall'utente che le ha scritte.

2.1.4.2 Strutture dati Il contratto usa tre strutture dati: Review, Company e Customer.
La struttura *Review* rappresenta una singola recensione ed è composta da quattro 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.
- **state**: rappresenta lo stato della recensione, può essere ACTIVE, MODIFIED o DELETED.

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.

2.1.4.3 Mappe Il contratto definisce due mappe private 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.

2.1.4.4 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;

2.1.4.5 Metodi Il contratto definisce le seguenti funzioni:

- **function havePayed(address myAddress, address companyAddress) public view returns (bool)** :
  La funzione havePayed è una funzione di sola lettura che controlla se un utente specifico ha pagato per lasciare una recensione per una determinata azienda. Prende in input l'indirizzo dell'utente (myAddress) e l'indirizzo dell'azienda (companyAddress) e restituisce un valore booleano true se l'utente ha pagato per la recensione, false altrimenti. La funzione legge lo stato della recensione dell'utente all'interno della mappa delle recensioni dell'azienda e controlla il valore booleano havePayed. Se il valore è true, l'utente ha pagato per la recensione, altrimenti non ha pagato. La funzione non modifica lo stato della mappa delle recensioni dell'azienda o della mappa delle recensioni dell'utente.
- **modifier CheckTransaction(address companyWalletAddress)**:
  La funzione **CheckTransaction** viene usata come modificatore all'interno delle funzioni *WriteAReview* e *DeleteReview* per verificare che l'utente abbia effettuato una transazione verso l'indirizzo dell'azienda a cui sta scrivendo o cancellando una recensione. In particolare, la funzione **CheckTransaction** controlla che l'utente abbia già effettuato un pagamento (trasferimento di token) all'azienda corrispondente, salvato nel mapping *reviewMap* della *Company* relativa all'indirizzo dell'azienda e all'interno del mapping *customerMap* della *Customer* relativa all'indirizzo dell'utente che ha effettuato la transazione. Se l'utente non ha ancora effettuato una transazione, la funzione emette un messaggio di errore.
- **modifier CheckActionToYourself(address yourAddress)**:
  Il modificatore CheckActionToYourself impone che l'azione richiesta non possa essere eseguita dall'indirizzo chiamante stesso. In particolare, viene effettuato un controllo sulla condizione che l'indirizzo passato come parametro yourAddress non coincida con l'indirizzo chiamante msg.sender. Nel caso dovesse coincidere restituisce un errore.

  Il modificatore viene utilizzato in due funzioni del contratto Trustify:

  - **DepositTokens**: permette di depositare token in un determinato indirizzo passato come parametro. L'indirizzo non può coincidere con l'indirizzo chiamante della funzione.
  - **WriteAReview**: permette di scrivere una recensione su un indirizzo aziendale passato come parametro. L'indirizzo non può coincidere con l'indirizzo chiamante della funzione.

- **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 DeleteReview(address reviewAddress) public CheckActionToYourself(reviewAddress)**
  La funzione DeleteReview viene utilizzata per eliminare una recensione. Questo viene fatto impostando lo stato della recensione come ReviewState.DELETED. In particolare, la funzione aggiorna lo stato della recensione dell'utente chiamante e lo stato della recensione dell'azienda a cui è stata lasciata la recensione. La funzione non elimina effettivamente la recensione dal contratto, ma imposta solo il suo stato come "eliminato" e quindi non verrà più considerata nelle funzioni che restituiscono le recensioni attive.

- **function StateToString(ReviewState state) private pure returns (string memory)** :
  La funzione StateToString è una funzione privata (che può essere chiamata solo all'interno del contratto) che prende in input un valore dell'enum ReviewState e restituisce una stringa che rappresenta lo stato corrispondente. Se il valore dell'enum è:

  - ACTIVE, restituisce la stringa "ACTIVE".
  - MODIFIED, restituisce la stringa "MODIFIED".
  - DELETED, restituisce la stringa "DELETED".

  Questa funzione viene utilizzata all'interno del contratto per convertire i valori dell'enum ReviewState in stringhe leggibili per l'utente.

- **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 GetCompanyReview(uint start, uint end, address companyAddress) public view returns (string[] memory, uint8[] memory)**:
  La funzione GetCompanyReview 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 e allo stato delle recensioni. 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.

- **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 GetMyReview(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 quattro array contenenti le stringhe delle recensioni, le stelle assegnate, lo stato delle recensioni 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.

2.2 Web-app$_G$

2.2.1 Breve descrizione

Trustify è una *webapp*$_G$ creata utilizzando *Angular*$_G$, uno dei framework open source più utilizzati per lo sviluppo di applicazioni web moderne. Inoltre, il team di sviluppo ha scelto di utilizzare *TypeScript* come linguaggio principale per la scrittura del codice, poiché questo linguaggio fornisce una tipizzazione statica e altre caratteristiche avanzate che aiutano a rendere il codice più facile da mantenere e da scalare.

L'obiettivo principale della *webapp Trustify* è quello di fornire agli utenti un'esperienza di interazione intuitiva e facile con il sistema decentralizzato *Trustify*. In particolare, la piattaforma consente di visualizzare le recensioni di un'azienda, fornendo un quadro completo e trasparente delle esperienze degli utenti con quella specifica attività commerciale. Gli utenti possono anche scrivere e pubblicare recensioni per un'azienda, condividendo le loro opinioni e raccomandazioni.

2.2.2 Angular service

- **WalletService** La classe WalletService definisce un servizio per la gestione della connessione con il wallet MetaMask e la comunicazione con la blockchain. Il servizio espone metodi per verificare se MetaMask è installato, se l'utente è connesso, per connettersi a MetaMask e passare alla rete Sepolia. Inoltre, il servizio permette di recuperare l'indirizzo dell'account corrente e l'URL del provider HTTP di Infura per la rete Sepolia.

  Metodi principali:

  - **getAccount()** è un metodo asincrono che restituisce l'indirizzo dell'account corrente. L'indirizzo dell'account viene utilizzato per inviare e ricevere transazioni sulla blockchain.
  - **isInstalled()** è un metodo asincrono che controlla se il plugin MetaMask è installato nel browser dell'utente poiché l'interazione con MetaMask è necessaria per gestire la connessione con la blockchain.
  - **isWalletConnected()** è un metodo asincrono che restituisce true se l'utente è attualmente connesso a MetaMask.
  - **connectToMetamask()** è un metodo asincrono che apre la finestra di dialogo di connessione di MetaMask e, se l'utente accetta la connessione, imposta l'indirizzo dell'account corrente e il provider *Web3*. Questo metodo viene utilizzato per connettersi a MetaMask e impostare le variabili di stato del servizio.
  - **switchNetwork()** è un metodo asincrono che permette di cambiare rete in cui si trova il wallet MetaMask, in questo caso viene utilizzato per passare alla rete Sepolia. Il metodo controlla prima se la rete è già presente in MetaMask, altrimenti la aggiunge.
  - **connect()** è un metodo asincrono che chiama il metodo *connectToMetamask()* e il metodo *switchNetwork()* per connettersi a MetaMask e passare alla rete Sepolia.

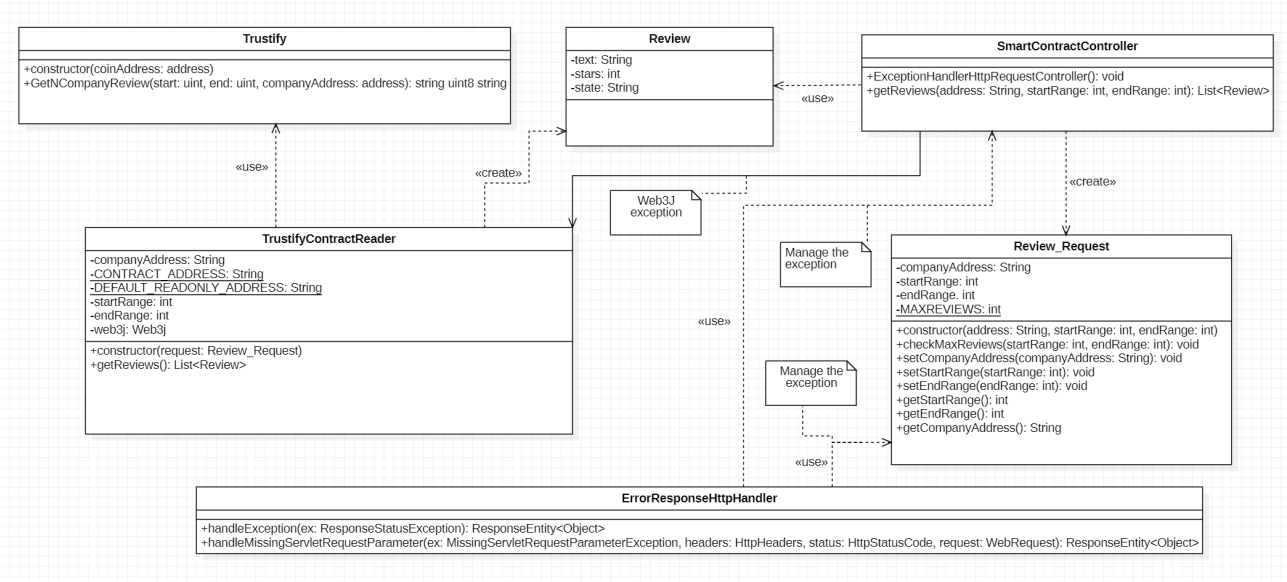
- **Web3Service** Questa classe, chiamata Web3Service, rappresenta un servizio Angular che fornisce una serie di funzionalità per interagire con la blockchain Ethereum attraverso la libreria Web3.js. In particolare, la classe gestisce la connessione con il wallet dell'utente e fornisce metodi per effettuare transazioni, accedere agli smart contract, e recuperare informazioni dalla blockchain.

  Metodi principali:

  - **pullTCoin** è un metodo asincrono che invia una transazione allo smart contract TCoin per richiedere l'invio di token.
  - **getTokenBalance** è un metodo asincrono che restituisce il saldo di token TCoin dell'account con cui il servizio comunica.
  - **approveTokens** è un metodo asincrono che richiede l'autorizzazione per la spesa di un determinato numero di token *TCoin* allo smart contract *TCoin*.
  - **depositTokens** è un metodo asincrono che invia una transazione alla smart contract Trustify per depositare un determinato numero di token TCoin.
  - **writeAReview** è un metodo asincrono che invia una transazione alla smart contract Trustify per scrivere una recensione per un'azienda.
  - **getCompanyReview** è un metodo asincrono che restituisce un array contenente le recensioni delle aziende tra un intervallo di indici.
  - **getSpecificReview** è un metodo asincrono che restituisce la recensione e il punteggio in stelle associati a un'azienda specifica.
  - **getMyReview** è un metodo asincrono che restituisce la recensione e il punteggio in stelle dell'utente corrente per una determinata azienda.
  - **deleteReview** è un metodo asincrono che invia una transazione allo *smart contract Trustify* per cancellare la recensione dell'utente corrente per una determinata azienda.

2.3 API-REST$_G$

2.3.1 Diagramma delle classi



2.3.2 Descrizione delle classi

- 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.

2.4 Design pattern utilizzati

Per il progetto *Trustify* sono stati utilizzati i seguenti design pattern:

- **Dependency Injection**: è stato utilizzato per iniettare le dipendenze tra i vari componenti dell'applicazione.