Laurea Magistrale in Informatica - Università degli studi di Salerno
Corso di *Sicurezza dei Dati* -
Professori Alfredo De Santis, Christiancarmine Esposito

Documentazione progetto - Chain of Art

Autori

Ciano Marco
matr. 0522501674
m.ciano4@studenti.unisa.it

De Martino Angela
matr. 0522501589
a.demartino68@studenti.unisa.it

Sommario

1. Introduzione

Nel panorama emergente delle applicazioni decentralizzate (Dapp [2]), “Chain of Art” si posiziona come un progetto pionieristico che fonde l’innovazione tecnologica della blockchain con il mondo dell’arte. Il progetto "Chain of Art" è un ecosistema che unisce l'arte digitale, la tecnologia blockchain e il Web 3.0 per creare una piattaforma innovativa di acquisto e vendita di opere d'arte digitali. Questa combinazione offre agli artisti un modo decentralizzato per esporre e vendere le proprie opere, mentre fornisce agli acquirenti una sicurezza e una trasparenza garantite dalla tecnologia blockchain.

Il sito web interagisce con lo smart contract Ethereum chiamato "PaintContract". Questo contratto gestisce la collezione di quadri digitali, tracciando la disponibilità di ogni opera, gestendo le transazioni di acquisto e mantenendo un registro degli acquisti effettuati da ciascun utente.

Gli utenti possono navigare tra le opere, visualizzare i dettagli dei quadri e acquistarli tramite l'interfaccia del sito web. Le transazioni di acquisto sono eseguite sulla blockchain Ethereum, garantendo trasparenza e immutabilità.

In questo paper, esploreremo come “Chain of Art” sfrutta la tecnologia blockchain per rivoluzionare il mercato dell’arte, creando un nuovo paradigma in cui la bellezza artistica si intreccia con la generosità umanitaria. Analizzeremo l’architettura della piattaforma, il suo modello di business, le strategie di coinvolgimento della comunità artistica e dei benefattori, e l’impatto sociale che il progetto si prefigge di realizzare. Attraverso “Chain of Art”, si apre un nuovo capitolo nel mondo dell’arte, dove ogni transazione diventa un atto di sostegno a cause più grandi, dimostrando il potere dell’arte come forza per il cambiamento positivo nella società.

2. Tecnologie utilizzate

Il cuore di questo progetto è alimentato da una suite di strumenti tecnologici avanzati, tra cui Ganache, MetaMask, Truffle, GitHub e Google Drive, ciascuno svolgendo un ruolo fondamentale nel suo funzionamento e sviluppo.

Ganache è un pezzo chiave del puzzle, fornendo un ambiente di sviluppo blockchain locale per testare smart contracts e transazioni in un contesto sicuro e controllato, simulando la rete Ethereum.

MetaMask agisce come un interfaccia essenziale, un portafoglio criptovalute basato su browser che consente agli utenti di interagire con la DApp in modo sicuro e decentralizzato. È fondamentale per la gestione delle identità degli utenti e per le transazioni di acquisto.

Il framework di sviluppo Truffle offre una suite di strumenti per lo sviluppo, il test e la distribuzione di smart contracts, garantendo che siano robusti, sicuri e ben integrati nel network Ethereum.

GitHub gioca un ruolo vitale nella collaborazione tra gli sviluppatori e nella gestione del versioning[5] del software, mantenendo il codice sorgente di “Chain of Art” accessibile, trasparente e costantemente aggiornato.

Infine, Google Drive è lo strumento scelto per la gestione documentale e la collaborazione tra i membri del team, consentendo la condivisione e la modifica efficace di documenti e piani di progetto.

3. Struttura del progetto

https://bit.ly/3vk7432

Il progetto "Chain Of Art" è organizzato in una struttura di cartelle e file che facilita lo sviluppo e la manutenzione del codice. Di seguito è fornita una descrizione dettagliata della struttura del progetto:

Directory Principali

build/: Contiene i file compilati del contratto intelligente. Questi file JSON includono l'ABI[1] e l'indirizzo del contratto, essenziali per l'interazione front-end con il contratto.

contracts/: Questa directory ospita i file sorgente dello smart contract scritti in Solidity. Qui risiede la logica di business principale del progetto.

migrations/: Contiene gli script di migrazione usati da Truffle per distribuire i contratti intelligenti sulla blockchain.

node_modules/: Directory generata automaticamente contenente tutte le dipendenze del progetto installate tramite NPM (Node Package Manager).

src/: La cartella più importante per lo sviluppo front-end. Include tutti i file HTML, CSS, JavaScript e le immagini necessarie per l'interfaccia utente.

test/: Contiene gli script di test per i contratti intelligenti, permettendo di eseguire test automatizzati per assicurare la correttezza del codice del contratto.

File Principali

truffle-config.js: Il file di configurazione per Truffle, utilizzato per impostare le reti di blockchain, i compilatori e altre opzioni di configurazione.

package.json e **package-lock.json**: Questi file gestiscono le dipendenze del progetto e assicurano una coerenza nell'ambiente di sviluppo.

Directory src/

All'interno della directory src/, si trovano le seguenti sottodirectory e file:

src/index.html: Il file principale dell'interfaccia utente, che funge da punto di ingresso per l'applicazione web.

src/js/: Contiene i file JavaScript, inclusi app.js (che gestisce la logica principale dell'interfaccia utente e l'interazione con il contratto), web3.min.js (libreria per interagire con Ethereum) e truffle-contract.js (per lavorare con i contratti Truffle).

src/css/: Ospita i fogli di stile CSS per la personalizzazione dell'aspetto dell'interfaccia utente.

src/fonts/ e **src/img**/: Contengono rispettivamente i font e le immagini utilizzati nell'interfaccia utente.

src/paints.json: Un file JSON che potrebbe essere usato per memorizzare i dati dei quadri, come parte della logica dell'applicazione.

Questa struttura consente una separazione chiara tra la logica del contratto intelligente e l'interfaccia utente, facilitando lo sviluppo e la manutenzione.

4. Setup e configurazione

Per configurare e avviare il progetto “Chain Of Art”, segui i passaggi dettagliati qui sotto.

Prerequisiti

Prima di procedere, assicurati di avere installato:

- **Node.js** e **npm**: Necessari per gestire le dipendenze JavaScript del progetto.
- **Truffle**: Utilizzato per lo sviluppo, il test e la distribuzione dei contratti intelligenti Solidity.
- **Ganache**: Una blockchain personale per lo sviluppo Ethereum, utilizzata per testare il progetto in un ambiente locale.

Installazione delle Dipendenze

1. Clonazione del Progetto: Se il progetto è ospitato su un repository Git, clonalo nel tuo sistema. Altrimenti, assicurati di avere tutti i file del progetto estratti dalla cartella ZIP.
2. Installazione delle Dipendenze Node: Apri un terminale nella directory principale del progetto e esegui il comando **npm install package.json**. Questo comando installerà tutte le dipendenze elencate nel file package.json.

Configurazione di Truffle e Ganache

1. Avvio di Ganache: Avvia Ganache (GUI o CLI). Prendi nota dell’indirizzo e della porta mostrati nell’interfaccia di Ganache, che di solito è http://localhost:7545.
2. Configurazione di Truffle: Assicurati che nel file truffle-config.js le impostazioni di rete siano configurate per corrispondere a quelle di Ganache. Questo garantirà che Truffle possa connettersi alla tua blockchain locale.

Distribuzione dei Contratti Intelligenti

1. Compilazione dei Contratti: Nella directory principale del progetto, esegui il comando **truffle compile** per compilare i contratti intelligenti.
2. Migrazione dei Contratti: Utilizza il comando **truffle migrate --network** development per distribuire i contratti sulla blockchain Ganache.

Avvio dell’Interfaccia Utente

1. Visualizzazione dell’Interfaccia Utente: Apri il file src/index.html nel tuo browser per interagire con l’applicazione. Se il progetto include un server di sviluppo (ad esempio, configurato nel package.json), puoi avviarlo eseguendo npm start.

Test

Esegui i test dei contratti intelligenti con il comando truffle test nella directory principale del progetto. Questo comando eseguirà tutti gli script di test presenti nella cartella test.

5. Focus su smart contract

Il cuore del progetto “Chain Of Art” è rappresentato dagli smart contract scritti in Solidity. Questi contratti risiedono nella directory **contracts**/ del progetto e sono responsabili della logica di business principale, come la gestione degli acquisti e la proprietà delle opere d’arte digitali. Di seguito è fornita una descrizione dettagliata del funzionamento e delle funzionalità principali dello smart contract.

Descrizione dello Smart Contract

- Nome del Contratto: PaintContract
- Versione Solidity: pragma solidity ^0.8.0;

Funzioni Principali dello Smart Contract

1. **createPaint(uint id, string memory title, string memory img, string memory artist, string memory price)**: Crea un nuovo quadro nella collezione con i dettagli forniti e verifica se il quadro creato con lo stesso id non esiste già.

1. **loadPaints()**: Carica inizialmente una serie di quadri usando la funzione createPaint. Tale funzione viene utilizzata per inizializzare il contratto con un insieme predefinito di quadri.

1. **buyPaint(uint paintId)**: Consente all’utente di acquistare un quadro specifico, Verificando che il quadro sia disponibile e che l’utente non abbia acquistato già lo stesso quadro. Inoltre, aggiorna lo stato del quadro come venduto, registra l’acquisto ed emette un evento.

2. **PurchasedStatus(address buyer, uint paintId)**: Funzione di supporto che verifica se un utente ha già acquistato un determinato quadro.

3. **getPurchasedPaints(address buyer)**: Restituisce un array di strutture Paint che rappresentano i quadri acquistati da un utente specifico.

4. **getPaintDetails(uint paintId)**: Restituisce i dettagli di un quadro dato il suo Id.

Eventi

- **PurchasedPaint**: Questo evento viene emesso quando un’opera d’arte viene acquistata, registrando l’acquirente e l’ID dell’opera.

Sicurezza e Considerazioni

- Validazioni: Il contratto include controlli per evitare acquisti doppi e per assicurarsi che vengano inviati fondi adeguati per l’acquisto.

Deploy e Interazione

- Distribuzione: Il contratto viene distribuito sulla blockchain attraverso gli script di migrazione presenti nella cartella migrations/.
- Interazione: L'interazione con il contratto avviene tramite l'interfaccia utente del progetto, che utilizza Web3.js per inviare transazioni e richiamare le funzioni dello smart contract.

6. Front-end

L'interfaccia utente del progetto "Chain Of Art" offre agli utenti la possibilità di interagire con il sistema di acquisto delle opere d'arte. Realizzata con HTML, CSS e JavaScript, l'interfaccia è progettata per essere intuitiva e accessibile, consentendo agli utenti di visualizzare facilmente le opere disponibili e di procedere con gli acquisti.

7. Interazione Web3

Nel progetto “Chain Of Art”, la libreria Web3.js gioca un ruolo fondamentale nell’interazione tra l’interfaccia utente (front-end) e la blockchain Ethereum. Il file web3.min.js, situato nella directory src/js/, è il motore che permette all’applicazione di comunicare con lo smart contract sulla blockchain.

Funzioni Chiave di Web3.js:

Connessione con Ethereum: Web3.js si connette al provider Ethereum, come MetaMask o un nodo locale Ganache. Questo permette all’applicazione di interagire con la blockchain, inviando transazioni e richieste di lettura.

Interazione con lo Smart Contract: Utilizzando l’ABI (Application Binary Interface) fornita dal file PaintContract.json, Web3.js consente all’applicazione di richiamare le funzioni dello smart contract, come l’acquisto di opere d’arte o la verifica della proprietà.

Gestione delle Transazioni: La libreria gestisce l’invio di transazioni alla blockchain, inclusa la gestione del gas e la conversione dei valori in Ether.

Integrazione nel File app.js:

Inizializzazione di Web3: Nel file app.js, Web3.js viene inizializzato e configurato per utilizzare il provider Ethereum disponibile. Questo passaggio è essenziale per garantire che l’applicazione possa effettuare chiamate allo smart contract.

Richiamo delle Funzioni del Contratto: app.js include la logica per interagire con le funzioni dello smart contract tramite Web3.js, consentendo agli utenti di eseguire azioni come l’acquisto di quadri e la visualizzazione delle informazioni di proprietà.

8. Testing

Il testing è un aspetto vitale nel ciclo di vita dello sviluppo dello smart contract nel progetto "Chain Of Art". I test assicurano che lo smart contract funzioni come previsto e che sia sicuro da eventuali vulnerabilità.

Test dello Smart Contract:

Script di Test: Situati nella directory test/, gli script di test sono scritti per verificare vari aspetti e funzionalità dello smart contract, come la corretta assegnazione della proprietà e il funzionamento delle transazioni.

Esecuzione dei Test con Truffle: Utilizzando Truffle, una suite di strumenti per lo sviluppo di Ethereum, i test possono essere eseguiti eseguendo **truffle test** dalla riga di comando. Questo comando esegue tutti gli script di test e fornisce un report dei risultati.

Importanza dei Test: I test aiutano a scoprire bug, problemi di logica o potenziali vulnerabilità nello smart contract. Sono una parte fondamentale per garantire la stabilità e la sicurezza del contratto prima della sua distribuzione su una rete pubblica Ethereum.

9. Considerazioni e conclusioni

Il progetto “Chain Of Art” unisce arte e tecnologia attraverso l’utilizzo della blockchain, offrendo una piattaforma innovativa per l’acquisto e la vendita di arte digitale. Con un’interfaccia utente intuitiva e un solido smart contract, il progetto ha già dimostrato il suo potenziale nel democratizzare l’accesso all’arte digitale. Guardando al futuro, ci sono diverse aree entusiasmanti per l’espansione e il miglioramento. La piattaforma potrebbe beneficiare dell’introduzione di sistemi di aste e di rivendita, rendendo il mercato dell’arte più dinamico e accessibile. Allo stesso tempo, miglioramenti nell’interfaccia utente e l’implementazione di funzionalità di valutazione e social sharing[4] potrebbero aumentare significativamente l’engagement degli utenti[3] e la visibilità degli artisti. Mantenendo un focus sulla sicurezza e l’innovazione tecnologica, “Chain Of Art” è ben posizionato per continuare a crescere e affermarsi come punto di riferimento nel panorama dell’arte digitale.

Dizionario

1. ABI: (Application Binary Interface): rappresenta l'interfaccia tra due programmi o componenti di software, consentendo loro di comunicare tra loro.
2. DApp: applicazione decentralizzata.
3. Engagement degli utenti: si riferisce al coinvolgimento e all'interazione degli utenti con l'applicazione o il sito web.
4. Valutazione e social sharing: riferito alle possibili funzionalità future della piattaforma "Chain Of Art" in termini di valutazioni delle opere d'arte e condivisione su social media.
5. Versioning: controllo delle versioni.