

Requisiti Block Chat

Damiano Nardi, Arianna Patrizi, Simone Bruno

June 22, 2021

1 Introduzione

Gli Smart contracts, come funzionalità aggiuntiva alla blockchain, hanno recentemente ricevuto maggiore attenzione. Sono programmi eseguibili la cui istanza e stato sono archiviati in blockchain. Pertanto, i Smart contracts e la blockchain consentono un protocollo affidabile, tracciabile e irreversibile senza la necessità di terze parti fidate che generalmente costituiscono un singolo punto di errore. Se un utente crea e distribuisce uno smart contract, altri saranno in grado di interagire con esso, mentre la blockchain sottostante garantisce un'esecuzione affidabile. Nello specifico sono stati introdotti nuovi standard per gli smart contract come l'ERC-721 [1] che ha trovato ampia applicazione soprattutto nel mondo dell'arte digitale.

1.1 La nostra idea

L'ERC-721 [1] è stato usato fino ad ora come modello per gestire gli attestati di proprietà sulla blockchain. In questo progetto invece si è utilizzato come base per costruire un meccanismo di autorizzazione rispetto a determinate azioni che l'utente può compiere sia nella blockchain che all'esterno (nel nostro backend).

2 Descrizione

Si vuole creare un social su blockchain dove una celebrità può creare un proprio NFT che può essere acquistato da più persone. Colui che lo acquista può inviare il primo messaggio (sulla blockchain con garanzia di risposta) instaurando un canale di comunicazione con la sua celebrità preferita. Ogni persona che acquista un NFT da una celebrità ha un NFT unico in quanto esso viene creato con un numero identificativo di creazione. Da qui possiamo progettare anche un sistema di compravendita di questi NFT tra utenti che hanno il desiderio di collezionarli: Gotta Chat 'Em All.

2.1 Obiettivi

- Permettere alle celebrità di creare i loro nft impostando determinati vincoli;

- Permettere alle celebrità di ottenere parte dei guadagni generati dall'acquisto degli nft da loro creati;
- Instaurare un canale di comunicazione con chi ha comprato un nft creato da una celebrità;
- Impedire il ricevimento continuo/eccessivo di messaggi da parte della celebrità a sua discrezione;
- Permettere agli utenti un refund dei soldi spesi in caso di mancata risposta da parte della celebrità;
- Permettere lo scambio e la vendita degli nft comprati da parte degli utenti;
- Permettere ai creatori del sistema (noi autori) di ricevere le tasse raccolte;
- interfaccia user-friendly e funzionale.

2.2 Flussi di interazione

In questa sezione analizzeremo come i vari componenti del progetto interagiscono tra di loro nel corso dello svolgimento delle operazioni che si è in grado di eseguire. Per ogni operazione, si suppone di essere connessi al sito con Metamask o software simili.

2.2.1 Creazione Nft

Quando un utente vuole creare un proprio nft, inserisce i dati necessari all'interno di un form:

The screenshot shows the 'Crea Nft' form in the BlockChat application. The form is a white card with rounded corners. It includes the following fields and values:

- Nome Nft:** A text input field containing 'prova'.
- Data Scadenza:** A date picker showing '10 / 11 / 2021'.
- Minuti di blocco per messaggio:** A dropdown menu showing 'nessun blocco di default'.
- Limite Messaggi:** A dropdown menu showing 'nessun blocco di default'.
- Numero di nft comprabili:** A dropdown menu showing 'nessun limite'.
- Costo:** Two input fields. The first contains '1000000000000000000' and the second contains '1'.

A 'Crea Nft' button is located at the bottom of the form.

Figure 1: Alcuni campi sono obbligatori, ad esempio il nome.

ed infine effettua la transazione su blockchain, pagando il costo di transazione, calcolato tramite un'apposita funzione:

$$CostoCreazione = (costo/3) + ((costo * (tempoValidita)/100)$$

Dove "costo" è il prezzo che dovranno pagare gli utenti per acquistare l'nft, mentre "tempoValidita" è il numero di giorni per il quale è valido. Prima di richiedere la transazione all'utente, il sistema controlla se effettivamente il costoCreazione permette all'utente di avere un guadagno sufficiente:

$$GuadagnoMassimo = (costo - ((costo * 5/100))) * limiteMint$$

Dove limiteMint è il massimo numero di nft acquistabili. Per poter creare l' nft, è necessario che il guadagno potenziale sia superiore al prezzo per la creazione.

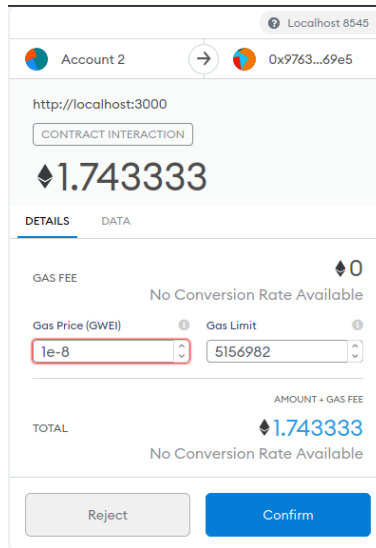


Figure 2: Il costo comprende quello della creazione del token e della transazione.

2.2.2 Acquisto Nft di una celebrità

Un utente che vuole acquistare un nft di un VIP, accede alla pagina contenente la lista dei NFT in vendita:

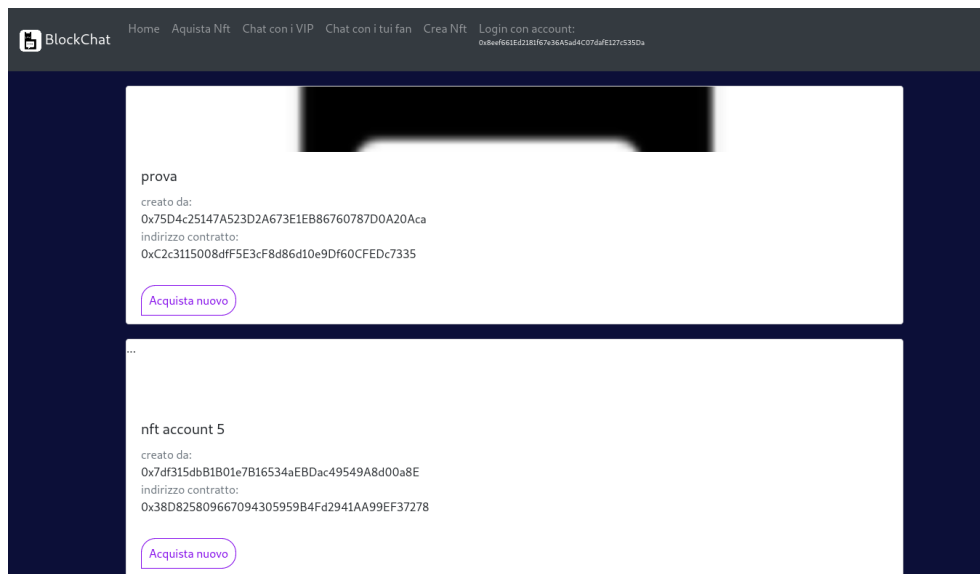


Figure 3: Lista dei token acquistabili.

E selezionando l'nft da acquistare, inserisce il primo messaggio(il quale sarà inviato su blockchain) da inviare alla celebrità.



Figure 4: L'invio del primo messaggio su blockchain garantisce la garanzia di risposta da parte del VIP.

Infine l'utente pagherà con una transazione su blockchain il costo dell'nft che vuole acquistare.

localhost:8545

Account 6 → 0xC2c3...7335

http://localhost:3000

CONTRACT INTERACTION

1

DETAILS DATA

GAS FEE 0
No Conversion Rate Available

Gas Price (GWEI) 1e-8 Gas Limit 309429

AMOUNT + GAS FEE

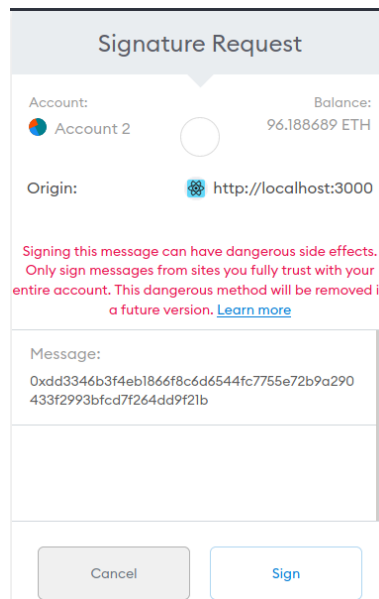
TOTAL 1
No Conversion Rate Available

Reject Confirm



Figure 5: l'utente paga solo il costo del token.


2.2.3 Invio di messaggi

Per l'invio dei messaggi, il sistema è diviso in due tipologie di chat, la chat con i VIP e quella con i fan. In entrambi i casi, le chat sono gestite su backend e per poter vedere/scrivere i messaggi(escluso il primo, il quale si trova su blockchain, quindi sempre visibile) è necessario effettuare un login, il quale consiste in una firma digitale.



Signature Request

Account:  Account 2  Balance: 96.188689 ETH

Origin:  http://localhost:3000

Signing this message can have dangerous side effects.
Only sign messages from sites you fully trust with your
entire account. This dangerous method will be removed in
a future version. [Learn more](#)

Message:
0xdd3346b3f4eb1866f8c6d6544fc7755e72b9a290
433f2993bfcd7f264dd9f21b

Figure 6: Firma digitale necessario per il login su backend.

Le due pagine con la lista delle chat sono molto simili, entrambe mostrano i vari token creati(in caso di chat con i fan) o posseduti(chat con i vip):

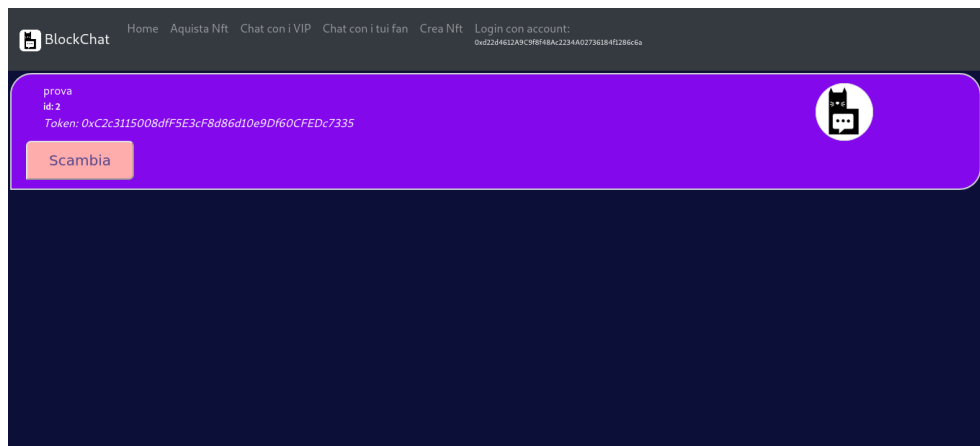


Figure 7: Lista delle chat con i vip di cui l'utente possiede il token.

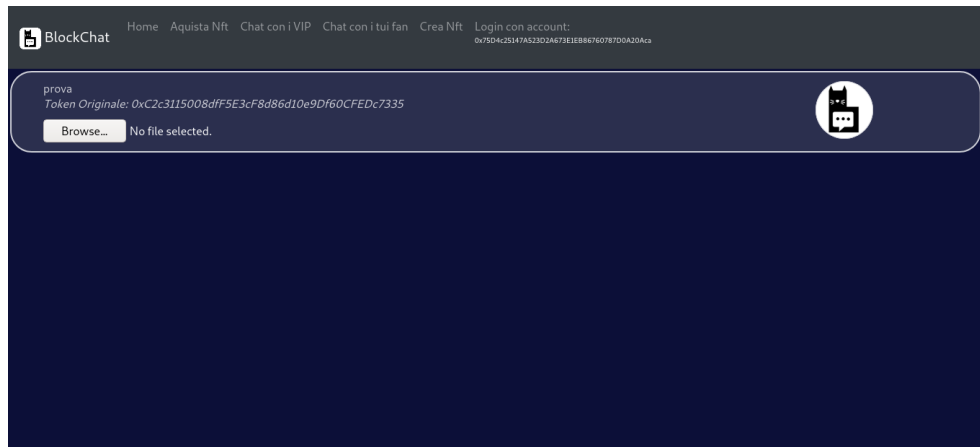


Figure 8: Lista dei proprio token creati, cliccando un token si accede alle chat relative a quel token.

Selezionando un nft, viene visualizzata la lista delle chat, dalla quale è possibile scegliere una chat e iniziare a messaggiare.

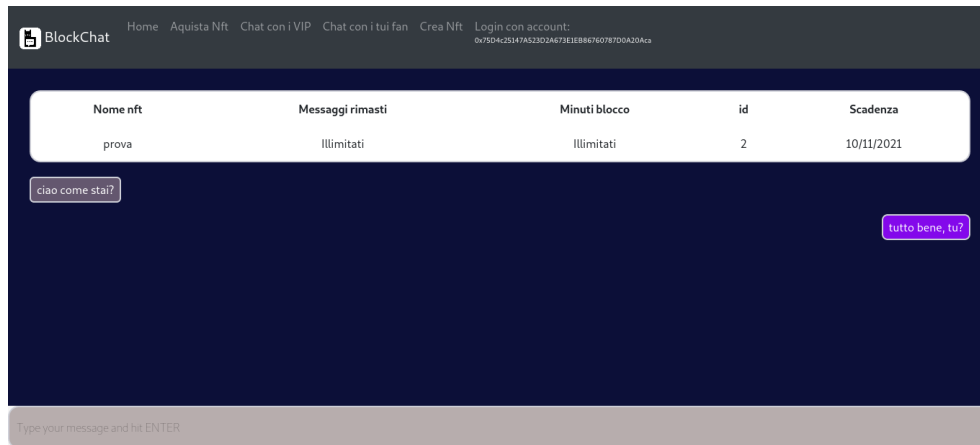


Figure 9: Esempio di chat.

2.2.4 Accredito dei ricavi alla celebrità

Per garantire la risposta al primo messaggio da parte del vip, i soldi dell'acquisto di un token nft non sono trasferiti immediatamente al vip, ma sono inviati al contratto. Saranno inviati successivamente, quando la celebrità avrà risposto al primo messaggio inviato dal fan, con una tassa del 5% che sarà mantenuta dal sistema.

2.2.5 Scambio e compravendita di nft

Quando un utente vuole scambiare un suo nft con un altro utente, seleziona l'apposito bottone sulla lista delle chat con i VIP e inserisce l'indirizzo del destinatario del nft da scambiare. Successivamente effettua la transazione su blockchain ed il token viene ceduto all'altro utente.

3 Implementazione della soluzione

3.1 Architettura

L'architettura proposta prevede tre entità principali il frontend, la blockchain, e il backend, queste entità cooperano tra di loro per garantire autenticazione, sicurezza, e verificabilità.

3.1.1 Frontend

Il front end interagisce con il wallet dell'utente, con il backend, e con la blockchain attraverso i metodi dello smart contract. Nello specifico si occupa di:

- **Visualizzazione:** prelevare i dati dalla blockchain usando i metodi dello smart contract e di visualizzarli all'utente. Prelevare la chat dal backend e visualizzarla
- **Inoltrare transazioni** Inoltrare le transazioni previste dal contratto quando un utente accede a determinate funzionalità del sistema come ad esempio la creazione di un nuovo nft, l'acquisto, lo scambio, ecc, ...
- **Loggarsi nel backend** Per il fare il login nel backend il frontend firma con la chiave privata nel wallet dell'utente (interfacendosi con esso) una stringa random generata dal backend ed invia la firma indietro al backend che risponderà con una set-cookie.

3.1.2 Blockchain

la Blockchain acceduta attraverso i metodi dello smart contract si occupa di:

- **Memorizzazione** memorizzazione delle variabili relative all'nft creato come ad esempio l'indirizzo di chi lo ha acquistato, creato, e i vari campi che il creatore ha inserito nella fase di creazione dell'nft
- **Controllo dei vincoli** Controllo minuzioso dei vincoli specialmente in ogni operazione che modifica le variabili al suo interno o nelle operazioni di pagamento e in caso annullare l'operazione chiamata.
- **Pagamento al creatore** Pagamento del creatore solo quando questo conferma la lettura del primo messaggio

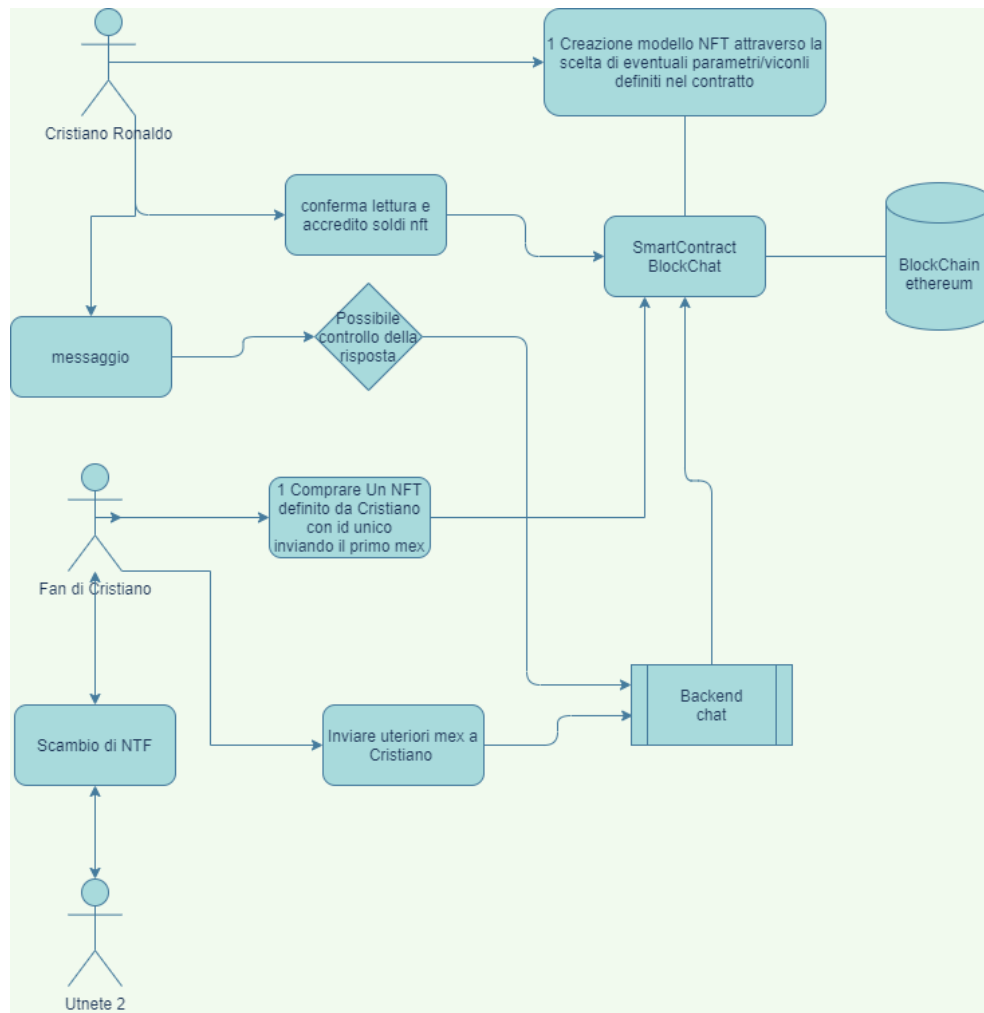


Figure 10: Funzionamento generale del sistema.

- **Pagamento ai proprietari del sistema** Pagamento ai proprietari del sistema della fee di creazione di un nuovo nft e della fee del 5% del costo dell'nft quando un utente lo acquista (ma accreditata solo quando il creatore conferma la lettura del messaggio dell'utente)

3.1.3 Backend

Il backend permette di lo scambio di messaggi tra la celebrità che crea l'nft e il fan che lo acquista. Per fare ciò si mettono a disposizione i seguenti servizi attraverso una REST-API:

- **login** il server deve poter riconoscere se l'utente possiede effettivamente la sua chiave, per questo richiede all'utente di firmare una stringa generata dal backend per rilasciare il cookie
- **memorizzazione** Il backend memorizza i messaggi che vengono scambiati controllando i necessari vincoli di tempo, numero, e proprietà. Questo lo fa interfacciandosi con la blockchain e controllando se l'utente ha fatto il login
- **get dei messaggi** Il backend restituisce quando richiesto i messaggi di una conversazione facendo i dovuti controlli di proprietà interfacciandosi con la blockchain e controllando se l'utente ha fatto il login

3.2 Linguaggi utilizzati

3.2.1 Smart contract: Solidity

Per sviluppare lo smartcontract su blockchain ethereum è stato usato il linguaggio Solidity [2] che è lo standard di fatto per lo sviluppo di smart contract su ethereum. Scendendo più nel dettaglio il nostro contratto è definito da due componenti:

- **BlockChat.sol** che viene istanziato una sola volta (da chi lo mette sulla blockchain) e mette a disposizione una serie di metodi per il get e di istanziazione del contratto NFT_MODEL.sol con i parametri inseriti dalla celebrità.
- **NFT_MODEL.sol** questo contratto definisce il modello dell'nft che crea la celebrità (mediante determinate variabili come nome_modello, pub_key_creatore, costo etc...) ed eredita tutti i metodi dello standard ERC-721 [1] e ne mette a disposizione ulteriori implementati da noi volti al suo utilizzo come per esempio reclamo() , confermaRisposta(), compraNft(), e i vari getter.

3.2.2 Frontend: ReactJS

Per sviluppare il Frontend abbiamo scelto di usare il framework ReactJS [3] in quanto

- **divisone in componenti** ci ha permesso di dividere le varie pagine in componenti separati
- **Gestione dello stato** la gestione dello stato dei vari componenti permette ad ogni cambio di stato di aggiornare efficientemente solamente le parti della UI che dipendono da esso.
- **esperienza utente** ReactJS ci permette di fornire all'utente una esperienza utente continua e senza ricaricamenti di pagina molto più simile ad un'app che ad un classico sito web

Componenti principali tutti i componenti si trovano nella cartella /frontend/src/components/ e vengono istanziati in base all'url della pagina dal file /frontend/src/App.js che è il main. I componenti principali sono i seguenti:

- **Homepage.jsx** si occupa di visualizzare la landing page
- **CreateNft.jsx** visualizza il form di creazione dell'nft facendo i dovuti controlli e permette la creazione di un nuovo nft mediante un pulsante
- **BuyNft.jsx** preleva le informazioni dell'nft che si vuole acquistare mostrandole e permette di acquistarlo attraverso un pulsante
- **Navigation.jsx** visualizza la navbar che permette di navigare all'interno della applicazione.
- **ChatListBuyer.jsx** e **ChatListCreator.jsx** visualizzano la lista delle chat il primo degli nft che si è comprati il secondo di quelli che si è creati
- **Chat.jsx** visualizza la chat di un nft riconoscendo se si è creatore o compratore
- **requestsAPI.js** Vi sono tutti i metodi per comunicare con il backend

3.2.3 Backend: python Flask

Per la semplicità di utilizzo e velocità di sviluppo lato backend abbiamo scelto di usare python con la libreria per il webserver Flask [4]. tutto il backend è racchiuso nel file /backend/back.py dove sono definiti gli end-point della rest-api

- **Rest-api** il back end mette a disposizione i seguenti endpoints

```
( '/api/prelogin ', methods=["GET"] )
( '/api/login ', methods=["POST"] )
( '/api/sendmex ', methods=["POST"] )
( '/api/getmex ', methods=["POST"] )
( '/api/upload_img ', methods=["POST"] )
```

per eseguire le operazioni citate prima di

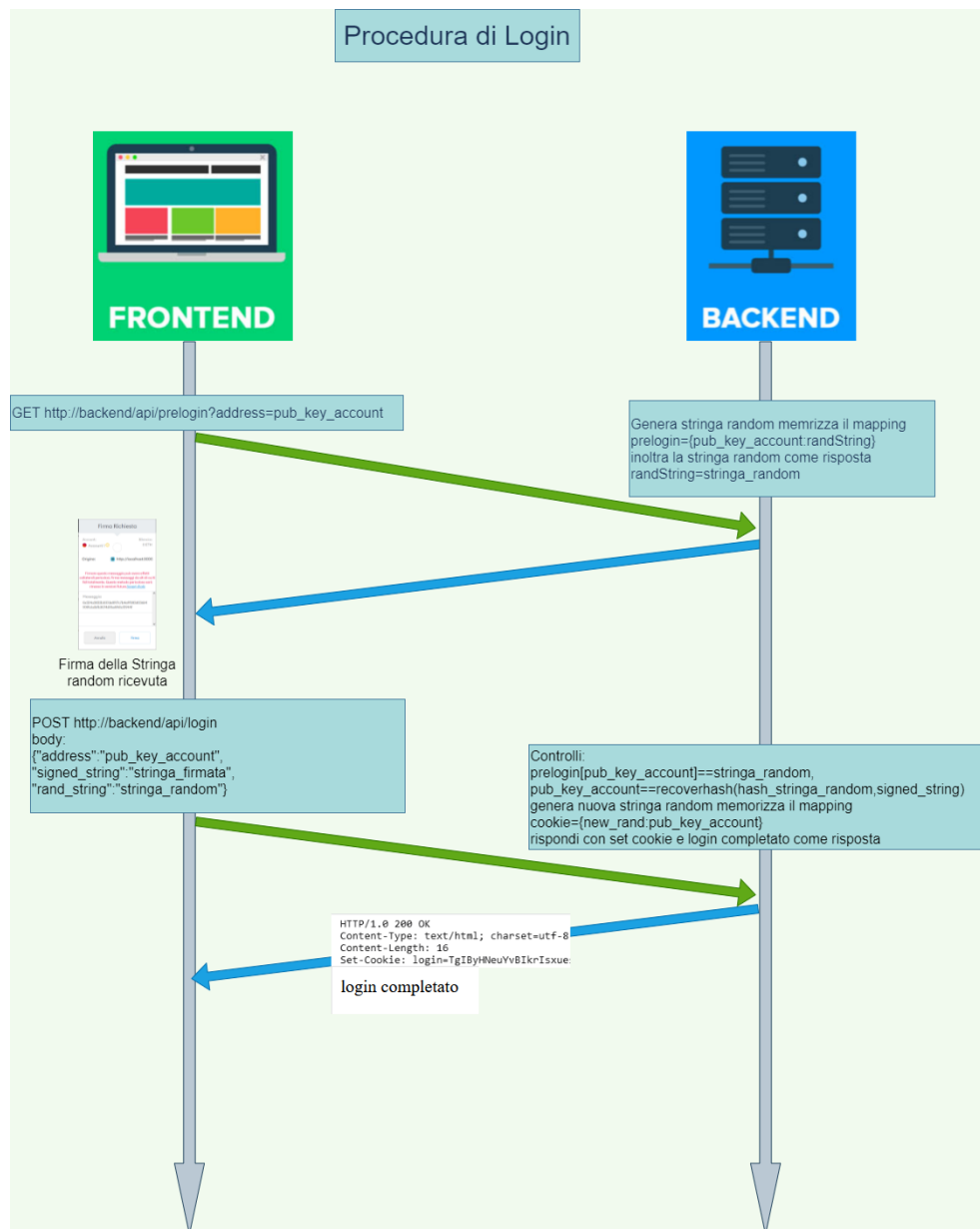


Figure 11: Funzionamento Login

- **Login** il login viene effettuato attraverso gli endpoint `'/api/prelogin'` e `'/api/login'`, il primo restituisce un stringa random che poi sarà restituita firmata dal frontend chiamando la seconda. A questo punto il backend memorizza una nuova stringa random associata alla chiave pubblica dell'utente ritorna una set-cookie con quest'ultima al frontend
- **get e send di messaggi** `'/api/sendmex'` `'/api/getmex'` queste servono rispettivamente per l'invio e la ricezione dei messaggi entrambe richiedono di aver effettuato precedentemente un login e che il richiamante sia il creatore o l'acquirente dell'nft
- **upload immagine** `'/api/upload.img'` permette di fare l upload di un immagine al creatore di un modello nft rispetto a un modello nft specifico. Anche questa richiede il login

3.3 Architettura Deploy

Per il deploy abbiamo usato due container docker [5] nella stessa rete interna docker, avviati attraverso docker-compose su questi due container ci girano rispettivamente:

- **NGNX** [6] è l'unico container che espone le porte 80,443 all'esterno della rete Docker. Questo container si occupa di tre compiti molto importanti:
 1. l'hosting del frontend
 2. reverse proxy al container con il backend
 3. fornire https per incrementare la sicurezza del sistema contro le intercettazioni (es: rubare il cookie del login)
- **Backend** In questo container non ci sono porte esposte all'esterno. Le comunicazioni che avvengono sono tra lui e il reverse proxy e la lettura della blockchain

3.4 Manuale di installazione

Sul repository gitHub [7] nei file README.md ci sono le istruzioni per installare ed avviare il progetto sia in un ambiente locale (`./README.md`) che in un ambiente di produzione (`./deploy-ropsten/README.md`)

4 Conclusioni

4.1 Risultati conseguiti

Abbiamo creato un prodotto funzionante in cui facciamo cooperare un sistema distribuito la blockchain, a uno centralizzato il webserver. Questo ci ha permesso di prendere il meglio tra i due mondi:

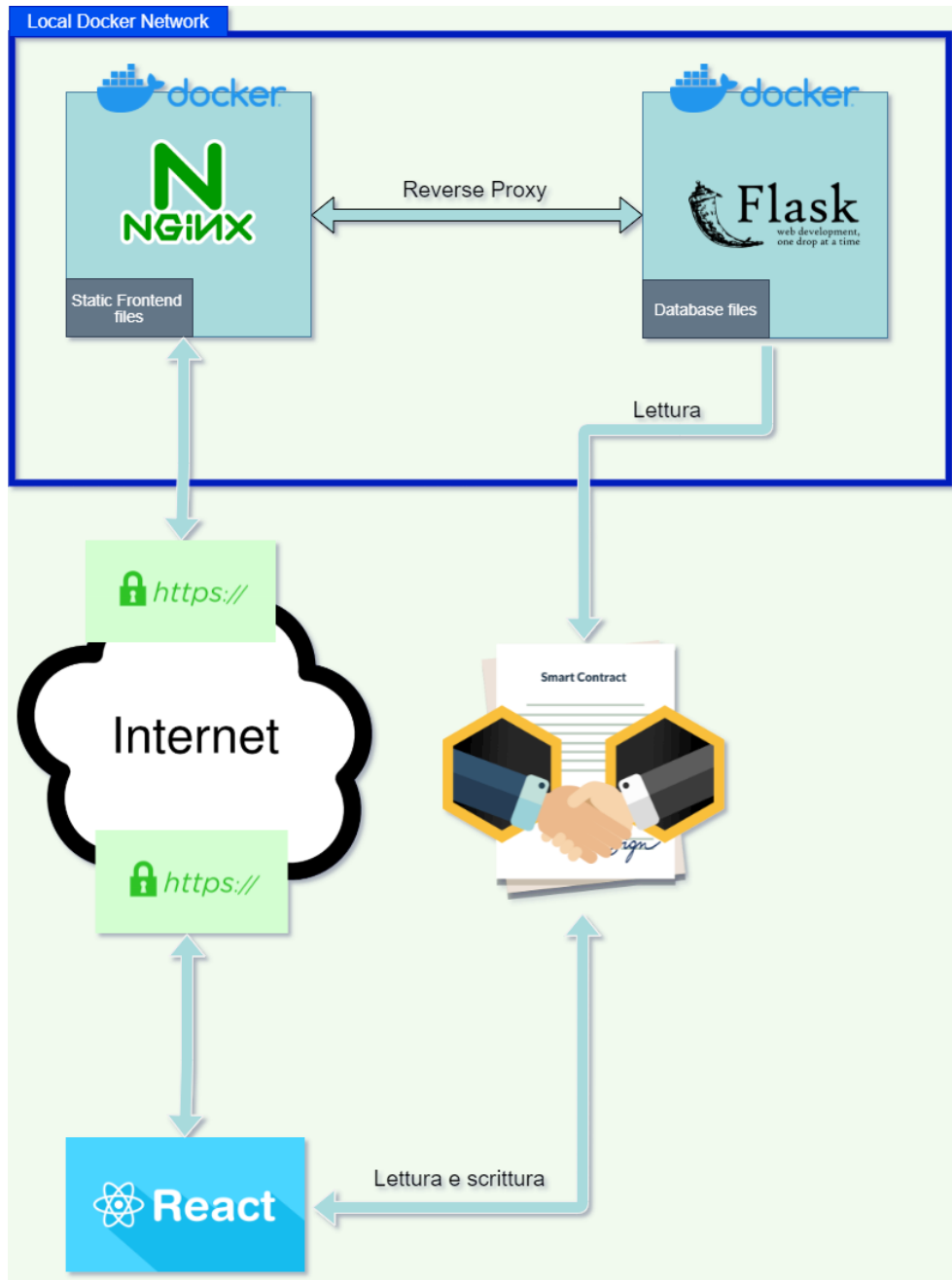


Figure 12: Vista dall'alto della architettura usata per il deploy

- "proprietà" e il pagamento sono modellati nello smart-contract
- il canale di comunicazione è modellato attraverso una classica architettura client-server dove per scambiarsi messaggi non vi è il costo del gas della transazione.

4.2 Osservazioni sulla dimensione dello smart contract

La dimensione massima per uno smart contract nella blockchain ethereum è **24576 bytes**, questo limite ci ha costretto a fare molta economia nel codice accorpare molte funzioni e controlli. Inoltre abbiamo dovuto togliere la parte di compravendita (che c'è ma commentata) in quanto faceva sfiorare il limite di 24576 bytes in nostro contratto attualmente pesa **24470.5 bytes** e con l'aggiunta anche di una sola funzione supera il limite consentito.

4.3 Sviluppi possibili

Eventuali miglioramenti futuri sono:

- L'aggiunta della funzionalità della segnalazione di un utente per comportamento non consono alle normative del social, soprattutto per una risposta non adeguata da parte della celebrità al primo messaggio del fan;
- Ridurre ulteriormente la dimensione del contratto implementando solo una parte dell'erc-721 per fare spazio ai metodi per la compravendita;
- Miglioramenti estetici;

References

- [1] Erc-721. <http://erc721.org/>.
- [2] Solidity 0.8.0 documentation. <https://docs.soliditylang.org/en/v0.8.0/>.
- [3] Reactjs una libreria javascript per creare interfacce utente. <https://it.reactjs.org/>.
- [4] Welcome to flask flask documentation (2.0.x). <https://flask.palletsprojects.com/en/2.0.x/>.
- [5] Empowering app development for developers docker. <https://www.docker.com/>.
- [6] Nginx high performance load balancer, web server, & reverse proxy. <https://www.nginx.com/>.
- [7] naddi96/block-chat-project. <https://github.com/naddi96/Block-Chat-Project>.