



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

# **Analisi di blockchain e sviluppo di un applicativo nel linguaggio di programmazione Solidity**

RELATORE

**Prof. Dario Di Nucci**

Università degli Studi di Salerno

CANDIDATO

**Luigi Allocca**

Matricola: 0512106979

Anno Accademico 2022-2023



## **Sommario**

La blockchain è una nuova tecnologia che permette di sviluppare applicazioni decentralizzate, effettuare transazioni immutabili e rimuovere la presenza di intermediari. L'interesse per questa tecnologia è arrivato anche in Europa, dove è stata fondata l'European Blockchain Partnership al fine di creare un'infrastruttura digitale europea nel settore pubblico. Il presente lavoro di ricerca vuole evidenziare i benefici derivanti dall'utilizzo di questa nuova tecnologia, mostrare come sviluppare applicazioni decentralizzate e fornire un esempio di applicazione decentralizzata sviluppata in Solidity.

<b>Indice</b>	<b>ii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Motivazioni e Obiettivi . . . . .	1
1.2 Risultati . . . . .	1
1.3 Struttura della tesi . . . . .	2
<b>2 Blockchain e Smart Contract</b>	<b>3</b>
2.1 Blockchain e Smart Contracts . . . . .	3
2.2 Protocolli di consenso . . . . .	5
2.3 Tipi di Blockchain più noti . . . . .	6
2.4 Decentralized Applications . . . . .	7
2.5 Sviluppo di Blockchain tramite Ethereum . . . . .	7
<b>3 Applicativo in Solidity</b>	<b>9</b>
3.1 Requisiti funzionali e non funzionali . . . . .	9
3.2 Use cases e scenari . . . . .	10
3.3 Diagramma delle classi . . . . .	12
3.4 Mockups . . . . .	14
3.5 Implementazione . . . . .	15
3.5.1 Creazione e deploy dello smart contract . . . . .	16
3.5.2 Creazione, pubblicazione e visualizzazione degli NFT da React . . . . .	20
3.6 Testing . . . . .	24

---

<b>4</b>	<b>Tecnologie utilizzate</b>	<b>27</b>
4.1	The Application Binary Interface (ABI) . . . . .	27
4.2	VSCode e Solidity . . . . .	27
4.3	Truffle . . . . .	27
4.4	Infura . . . . .	28
4.5	IPFS . . . . .	28
4.6	Goerli test network . . . . .	28
4.7	React JS e Node.js . . . . .	28
4.8	Moralis . . . . .	29
4.9	Librerie e frameworks utilizzati . . . . .	29
4.10	OpenSea . . . . .	29
<b>5</b>	<b>Conclusioni</b>	<b>30</b>
	<b>Ringraziamenti</b>	<b>31</b>

### 1.1 Motivazioni e Obiettivi

Lo scopo questo lavoro di tesi è lo studio della blockchain e lo sviluppo di un' applicativo nel linguaggio di programmazione Solidity. In particolare ho provato a migliorare l'attuale sistema del commercio dei beni digitali fornendo trasparenza, basse commissioni, transazioni rapide e sicure. Al fine di garantire tutto ciò ho avuto bisogno di un' infrastruttura che mi permettesse di ottenere tutto questo, e per questo ho scelto la blockchain. In particolare ho sviluppato una Dapp, cioè un'applicazione decentralizzata che permetta agli studenti di comprare e vendere appunti. Sfruttando i vantaggi della blockchain, oltre ai benefici descritti in precedenza, verrà tutelata la proprietà intellettuale degli utenti attraverso dei contratti, e dato che non ci sono intermediari nella transazione, l'acquirente dovrà soltanto pagare una piccola commissione, piuttosto che una percentuale extra come siamo abituati negli attuali sistemi di pagamento. Tutto questo sarà possibile grazie alla tecnologia NFT, cioè atti di proprietà e certificati di autenticità degli appunti ed agli smart contract, che andranno a gestire la fase di caricamento (minting) dell'NFT.

### 1.2 Risultati

La Dapp che ho realizzato permette di caricare, comprare e vendere appunti, i quali saranno caratterizzati da: autore, materia ed un titolo. Gli appunti saranno degli NFT, per la

precisione dei token ERC-721, il quale smart contract è stato sviluppato nel linguaggio di programmazione Solidity e poi deployato su Goerli, una testnet di Ethereum con all'utilizzo di Truffle. Per creare l'applicazione web sono stati utilizzati React Js e Node JS. Infine, per gestire il sistema di acquisto e vendita degli NFT mi sono affidato ad OpenSea, una piattaforma riconosciuta di compra vendita di NFT. Infine, per la fase di testing è stato utilizzato Selenium, dove sono stati svolti test di sistema, per i quali sono stati definiti obbiettivi riguardo cosa testare ed i risultati.

### **1.3 Struttura della tesi**

La tesi è strutturata come segue. Il Capitolo 2 descrive la blockchain e gli smart contract e approfondisce i concetti chiave di questi argomenti. Nel Capitolo 3 si descrivono le fasi di sviluppo dell'applicazione, ovvero le fasi di analisi, implementazione e testing, andando a motivare le scelte fatte. Successivamente il Capitolo 4 è dedicato alle tecnologie utilizzate, focalizzandosi sulle tecnologie e i framework utilizzati durante lo sviluppo. Infine vi sono le conclusioni, dove vi si riassumono i risultati ottenuti e si elencano gli sviluppi futuri del sistema.

#### 2.1 Blockchain e Smart Contracts

La blockchain è una catena di registri che forniscono l'archiviazione dei dati, dunque delle transazioni oppure di vere e proprie applicazioni Chiap et al. [2019].

In questa catena è possibile aggiungere nuovi blocchi in maniera sequenziale, ma non è possibile eliminarli e/o modificarli, grazie all'utilizzo della crittografia e i protocolli di consenso che appunto garantiscono sicurezza e immutabilità (Chiap et al. [2019]).

Uno smart contract è "un protocollo di transazione digitale che esegue i termini di contratto" Chiap et al. [2019], il quale obiettivo è quello di controllare se i vincoli di un contratto vengono soddisfatti in maniera automatica, senza avere il rischio di controparte Chiap et al. [2019].

Un esempio più evidente di smart contract sono le criptovalute, ma possono esserci anche sistemi più complessi che associano gli smart contract al concetto di proprietà di beni fisici o astratti (smart property) o al concetto di identità (smart identity) Chiap et al. [2019].

Possiamo dunque vedere gli smart contract come programmi generici, che hanno le caratteristiche dei contratti per come li conosciamo, che vengono eseguiti sulla blockchain. Gli accordi in questo modo non necessitano della legge, in quanto il contratto eseguirà in automatico il controllo per verificare se le condizioni del contratto sono soddisfatte e nel momento in cui le condizioni vengono soddisfatte, lo smart contract esegue automaticamente



delle azioni specifiche, per esempio trasferire denaro o la proprietà di un bene (Chiap et al. [2019]).

I principali vantaggi che offre la blockchain sono: decentralizzazione, disintermediazione, trasparenza, verificabilità, immutabilità e programmabilità dei trasferimenti. Per via di queste caratteristiche, la blockchain può essere utilizzata ad esempio per migliorare il sistema dei documenti personali e delle certificazioni, oppure gestire informazioni finanziarie e renderle sicure e trasparenti. Molte delle aziende più importanti al mondo utilizzano in qualche modo strumenti basati su blockchain e parte di questi sviluppano soluzioni basate su questa tecnologia (cry [2022]).

**Hashing e criptazione.** Le informazioni che vengono scambiate su una rete subiscono un processo di criptazione, cioè vengono codificate al fine di garantire l'accesso alle informazioni originali solo alle persone autorizzate (Chiap et al. [2019]). Due nodi per comunicare su un canale non sicuro come internet, devono criptare il messaggio utilizzando un algoritmo a chiave pubblica in modo da nascondere il messaggio a persone non autorizzate e garantire la confidenzialità del messaggio. Tuttavia se un messaggio criptato non può essere letto da persone non autorizzate, queste possono modificare il contenuto del messaggio. Al fine di evitare che ciò accada, la criptazione viene affiancata ad altre tecniche, come l'hashing.

**Firma digitale.** Le firme digitali si ottengono unendo la crittografia a chiave pubblica all'hashing, e servono a dimostrare l'identità di qualcuno. La firma digitale provvede:

- **Autenticazione:** Ogni utente è in possesso di una chiave privata, grazie alla quale si può identificare univocamente il mittente di un messaggio.
- **Integrità:** Ogni modifica ad un messaggio inviato ne modifica la firma, dunque è facilmente verificabile che è stato modificato. Questo è possibile perché si utilizza un algoritmo di Hashing.
- **Non ripudio:** Se un utente firma un messaggio, non può negare di averlo fatto in un secondo momento.

**Indirizzi.** L'anonimato è una delle qualità principali della blockchain (Wang et al. [2020]), infatti, non esistono profili utente, bensì indirizzi che rappresentano la destinazione di una transazione e vengono utilizzati per trasferire asset digitali.

Gli indirizzi tuttavia non sono prettamente utilizzati per i profili utente, ma anche per identificare i contratti. Ad esempio il contratto di un NFT ha un indirizzo attraverso il quale si possono eseguire transazioni, ma questo soltanto se ci stiamo interfacciando con una blockchain multipurpose come Ethereum. Se possediamo l'indirizzo di un contratto possiamo vedere lo storico delle transazioni eseguite su quel contratto e gli indirizzi che hanno partecipato alla transazione.

## 2.2 Protocolli di consenso

In una blockchain non c'è un' autorità centrale e di conseguenza nasce l'esigenza di raggiungere un'accordo tra i nodi sullo stato corretto della blockchain. Quest'accordo è definito dal **consenso**, il quale è un accordo generale tra i nodi di una rete, dove ognuno ha parte del potere decisionale. La blockchain, al fine di incentivare tutti i nodi a raggiungere un accordo, utilizza economia, matematica e teoria dei giochi (Chiap et al. [2019]).

Il consenso deve essere raggiunto anche in una situazione dove ci sono nodi malevoli, e per fare ciò sono stati progettati due diversi algoritmi per risolvere il problema, il Proof of Work (PoW) e il Proof of Stake (PoS), dove ognuno ha delle proprie varianti. I nodi che partecipano attivamente al processo di consenso sono chiamati **miner**, i quali fanno mining. Il mining è il processo attraverso il quale le transazioni vengono validate, aggregate in blocchi e aggiunte alla blockchain (Chiap et al. [2019]).

**Proof of Work (PoW).** Il Proof of Work è un protocollo di consenso utilizzato per raggiungere il consenso distribuito. Il PoW consiste nel trovare un numero che richiede molta computazione, il quale una volta trovato, diventa facile per gli altri nodi verificarne la correttezza. In questo sistema, un nodo è considerato valido solo se contiene una soluzione valida (Chiap et al. [2019]).

**Proof of Stake (PoS).** Un altro protocollo per raggiungere il consenso è il Proof of Stake (POS), il quale scopo è uguale a quello del PoW, ma a differenza di quest'ultimo i blocchi verranno validati da nodi scelti in anticipo in base alla quantità di criptovalute in possesso, definita come **stake** (Chiap et al. [2019]).

**Conferme.** Come già detto, le informazioni sulla blockchain come le transazioni sono raggruppate in blocchi i quali vengono aggiunti in modo progressivo come in una lista concatenata. Una transazione prima di essere inserita in un blocco deve superare una fase di

verifica. Quando una transazione supera la fase di verifica, essa ha **1 conferma**, ed aumenta il numero di conferme man mano che altri blocchi vengono creati (Chiap et al. [2019]).

## 2.3 Tipi di Blockchain più noti

Attualmente esistono quattro tipi di reti blockchain:

- **Pubbliche:** In questo tipo di blockchain chiunque può partecipare, e tutti hanno gli stessi diritti di lettura, modifica e convalida della blockchain. Viene utilizzata soprattutto nell'ambito delle criptovalute.
- **Private:** Per partecipare ad una blockchain privata invece bisogna avere il consenso dell'autorità che decide anche i permessi nella rete. Viene utilizzata soprattutto in ambito aziendale, e si dice che è una blockchain parzialmente decentralizzata per via delle restrizioni.
- **Ibride:** Un compromesso tra le reti pubbliche e quelle private, sono le blockchain ibride in modo da controllare l'accesso a dati sensibili, lasciando altri dati pubblici, dove ognuno ha diritto di lettura.
- **Di consorzio:** Quando delle aziende o delle comunità lavorano per uno scopo comune, utilizzano una rete blockchain di consorzio, dove ogni azienda o comunità ha il compito di mantenere la blockchain sicura, garantendo che alcuni dati siano accessibili soltanto ad utenti autorizzati.

Tra le più note blockchain troviamo: Hyperledger Fabric, Ethereum, Corda e Quorum. Tutte queste blockchain vengono utilizzate per motivi eterogenei, ad esempio un'azienda potrebbe voler utilizzare Hyperledger per sviluppare una Dapp privata in modo semplice e veloce, oppure se delle aziende vogliono negoziare in modo da proteggere a pieno la privacy possono optare per Corda. In questo lavoro di tesi però ci sarà un focus su Ethereum, una piattaforma blockchain general purpose e che quindi lascia agli sviluppatori una grande libertà, soprattutto grazie al concetto di **smart contract** i quali vengono eseguiti in modo decentralizzato, continuo e senza censure (Chiap et al. [2019]). In oltre gli sviluppatori che decidono di creare token sulla blockchain di Ethereum non dovranno ricreare una blockchain e i token possono essere conservati su un qualsiasi wallet compatibile con Ethereum, riducendo in modo importante l'effort che lo sviluppatore dovrà impiegare.

## 2.4 Decentralized Applications

Le app decentralizzate (Dapp) sono programmi informatici che vengono eseguiti su un sistema informatico distribuito, come una rete blockchain. A differenza dell'architettura client-server che alimenta la maggior parte delle app su Internet, le Dapp integrate con una rete blockchain possono eseguire una logica di app che è garantita come trasparente, verificabile e immutabile (Johnson et al. [2019]).

## 2.5 Sviluppo di Blockchain tramite Ethereum

Come già detto in precedenza, questo lavoro di tesi si focalizza sullo sviluppo di uno smart contract in **solidity**, il quale verrà successivamente deployato sulla blockchain di Ethereum. Solidity appunto, è un linguaggio ad alto livello, compilato in bytecode EVM.

Il linguaggio più importante e ampiamente adottato attualmente è Solidity. Esso è un linguaggio di programmazione Turing-completo di alto livello con una sintassi simile a JavaScript, è tipizzato staticamente, supporta l'ereditarietà e il polimorfismo, nonché librerie e tipi complessi definiti dall'utente. Quando si utilizza Solidity per lo sviluppo di contratti, questi sono strutturati in modo simile alle classi dei linguaggi di programmazione orientati agli oggetti. Il codice del contratto consiste in variabili e funzioni che le leggono e le modificano, come nella programmazione imperativa tradizionale (Wohrer and Zdun [2018]).

**Ethereum Virtual Machine (EVM).** Quando si parla di sviluppo di smart contracts sulla blockchain di Ethereum, è di fondamentale importanza parlare della Ethereum Virtual Machine e in che modo viene utilizzata. In sintesi, la EVM è un computer virtuale (e di conseguenza non fisico), aperto e distribuito e gli smart contracts sono i suoi programmi. I programmi in una EVM vengono eseguiti in modo sequenziale, ed utilizza come struttura dati una pila, dove i contenuti vengono eseguiti in modalità LIFO (Last in First out) (Hirai [2017]). La documentazione in inglese che descrive le specifiche della EVM si chiama Yellow Paper.

L'EVM è un automa a **stati** finiti, dove ad eccezione di alcuni parametri globali, la maggior parte degli stati sono contenuti in account. Gli stati sono una sequenza di byte e lo stato di un account trattiene informazioni riguardo codice, memoria, nonce e bilancio (Hirai [2017]).

Una EVM supporta le transazioni, le quali possono essere istanziate da un account esterno che vuole creare un contratto o chiamare un account. Questa transazione sarà deterministica e visibile pubblicamente (Hirai [2017]).

Quando si istanzia una transazione tra un account e un altro, indipendentemente se interno o esterno, l'account che viene chiamato riceve il bilancio, mentre l'account che istanzia la transazione deve pagare una fee, detta GAS fee (Hirai [2017]).

**GAS.** Lo STARTGAS ed il GASPRICE sono campi fondamentali per prevenire il degrado di Ethereum (Jani [2017]). Esso infatti può subire danno per via di loop accidentali e non o altri sprechi computazionali derivanti dal codice di chi scrive gli **smart contracts**. Per questo ogni transazione è tenuta a stabilire un limite al numero di passi computazionali di esecuzione del codice che può utilizzare (Jani [2017]). Il "**gas**" è l'unità fondamentale di calcolo e un passo computazionale costa generalmente 1 gas, ma può aumentare con l'aumentare dei dati memorizzati o con l'aumento dei passi computazionali (Jani [2017]). Inoltre c'è da pagare una tassa di 5 gas per ogni byte di dati della transazione al fine di pagare in modo proporzionale alle risorse che si sfruttano durante la transazione (Jani [2017]).

**Token e loro standardizzazione in ERC-721.** Gli smart contract sono collegati al concetto di Token, i quali rappresentano una risorsa all'interno di una blockchain (Chiap et al. [2019]). In questo lavoro di tesi approfondirò il concetto di NFT (Non fungible Token). Il concetto di NFT deriva originariamente da uno standard di Ethereum, che mira a distinguere i token con segni distintivi. Questo tipo di token può essere legato a proprietà virtuali o digitali come loro identificativo univoco. Con gli NFT, tutte le proprietà contrassegnate possono essere liberamente scambiate con valori personalizzati in base ai loro attributi. Gli NFT hanno stimolato notevolmente la prosperità dei mercati delle applicazioni decentralizzate (DApp) (Wang et al. [2021]).

Lo standard ERC-721 standardizza i token non fungibili definendo un'interfaccia API per implementare uno smart contract sulla blockchain di Ethereum e imponendo le funzionalità che tale smart contract deve fornire. Le funzionalità riguardano principalmente il trasferimento di token da un conto a un altro, il recupero del saldo corrente di un portafoglio e il recupero della proprietà di uno specifico token (Casale-Brunet et al. [2021]).

Gli NFT sono tutti diversi tra loro, e le caratteristiche sono specificate nei metadati. I metadati contengono informazioni sull'NFT, riguardo il proprietario, il file, il nome ed altri attributi che sono definiti dal creatore del contratto, i quali possono variare.

Il mio lavoro di tesi si concentra sull'analisi di Blockchain e sullo sviluppo di un applicativo nel linguaggio di programmazione Solidity. L'applicativo in particolare è un sistema che permette agli utenti di caricare, vendere e comprare appunti.

### 3.1 Requisiti funzionali e non funzionali

Nella fase di analisi sono stati individuati i requisiti funzionali ed i requisiti non funzionali. Tra i requisiti funzionali troviamo:

- **RF1:** Un utente con un wallet può caricare i propri appunti;
- **RF2:** Un utente qualsiasi può vedere la lista degli appunti caricati dagli altri studenti;
- **RF3:** Un utente con un wallet può acquistare degli appunti;
- **RF4:** Un utente può visualizzare tutti gli appunti di cui è in possesso.

Mentre i requisiti non funzionali sono:

- **RNF1:** La registrazione delle transazioni deve essere trasparente, quindi sicura e verificabile da tutti;
- **RNF2:** I costi delle transazioni deve essere basso;
- **RNF3:** I documenti caricati devono essere verificati e verificabili;

- **RNF4:** Dato che si tratta di un sistema di transazioni, c'è il bisogno di minimizzare la fiducia tra i sistemi di rete.

Proprio a fronte di questi requisiti non funzionali, si è optato per l'utilizzo della Blockchain, in quanto questi requisiti sono soddisfatti dalla tecnologia stessa e una soluzione basata su un modello client-server per questo tipo di sistemi sarebbe molto più complessa.

### 3.2 Use cases e scenari

Tra i casi d'uso dell'applicazione sono stati individuati: Caricamento degli appunti, Visualizzazione degli appunti, Visualizzazione anteprima degli appunti, Acquisto degli appunti e Vendita degli appunti, come è possibile vedere in figura 3.1.

Nome:	Caricamento degli appunti
ID:	UC01
Partecipante:	Utente
Flusso degli eventi	<ol style="list-style-type: none"> <li>1. L'utente naviga nella pagina "Carica NFT" attraverso il link.</li> <li>2. Inserisce gli appunti in formato PDF e compila le caselle di testo.</li> <li>3. Clicca sul pulsante "Carica".</li> <li>4. Paga la transazione e carica l' NFT.</li> </ol>
Condizione d'entrata:	
Condizione d'uscita:	Il caricamento è andato a buon fine oppure c'è stato un errore che verrà notificato all' utente.

Nome:	Visualizzazione degli appunti
ID:	UC02
Partecipante:	Utente
Flusso degli eventi	<ol style="list-style-type: none"> <li>1. L'utente naviga nella pagina "NFTs" attraverso il link.</li> <li>2. L'utente visualizza gli appunti caricati dagli studenti.</li> </ol>
Condizione d'entrata:	
Condizione d'uscita:	L'utente visualizza gli appunti.

Nome:	Visualizzazione anteprima degli appunti
ID:	UC03
Partecipante:	Utente
Flusso degli eventi	1. Include UC02. 2. L'utente sceglie gli appunti da visualizzare e clicca su "Visualizza anteprima"
Condizione d'entrata:	
Condizione d'uscita:	L'utente visualizza l'anteprima gli appunti.

Nome:	Acquisto degli appunti
ID:	UC04
Partecipante:	Utente
Flusso degli eventi	1. Include UC02. 2. L'utente sceglie gli appunti da acquistare e clicca su "Visualizza su OpenSea" 3. Acquista l'NFT.
Condizione d'entrata:	
Condizione d'uscita:	L'utente acquista gli appunti.

Nome:	Vendita degli appunti
ID:	UC05
Partecipante:	Utente
Flusso degli eventi	1. Include UC01. 2. L'utente accede su OpenSea 3. Sceglie un prezzo e vende gli appunti.
Condizione d'entrata:	
Condizione d'uscita:	L'utente mette in vendita gli appunti.

Per esaminare meglio l'interazione tra utente e sistema è stato stilato un use case diagram (vedere Figura 3.1), comprendente i seguenti task: (i) Caricamento degli appunti, (ii) Visualizzazione degli appunti, (iii) Acquisto degli appunti, (iv) Vendita degli appunti, (v) Visualizza appunti.



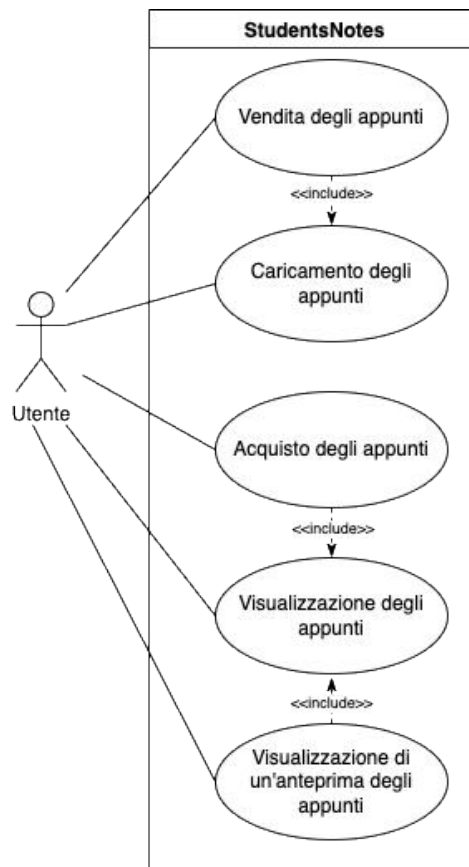


Figura 3.1: Use case diagram

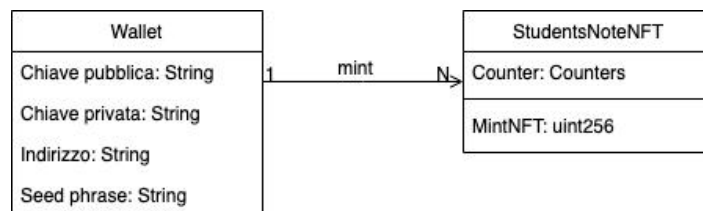
### 3.3 Diagramma delle classi

Al fine di ingegnerizzare il software ho stilato un diagramma degli oggetti, che mostra chi sono le classi che andranno ad interagire tra loro durante l'interazione col software, come mostrato in figura 3.2. Come si può vedere non si prevede una classe Utente o Persona, in quanto uno dei vantaggi della blockchain è l'anonimato, permesso grazie al concetto di wallet. Il wallet fornisce un'identità digitale sulla blockchain che non possiede informazioni personali del proprietario, ma soltanto una chiave privata, una chiave pubblica, un indirizzo e una seed phrase. Questi saranno rispettivamente:

- **Chiave segreta:** Utilizzata per effettuare il login e firmare transazioni.
- **Chiave pubblica:** Generata dalla chiave privata attraverso l'algoritmo ECDSA-512.
- **Indirizzo:** Generato dalla chiave pubblica attraverso l'algoritmo RipeMD 160, si ottiene infine l'indirizzo pubblico, che indica la "posizione" del singolo wallet sul registro della blockchain.

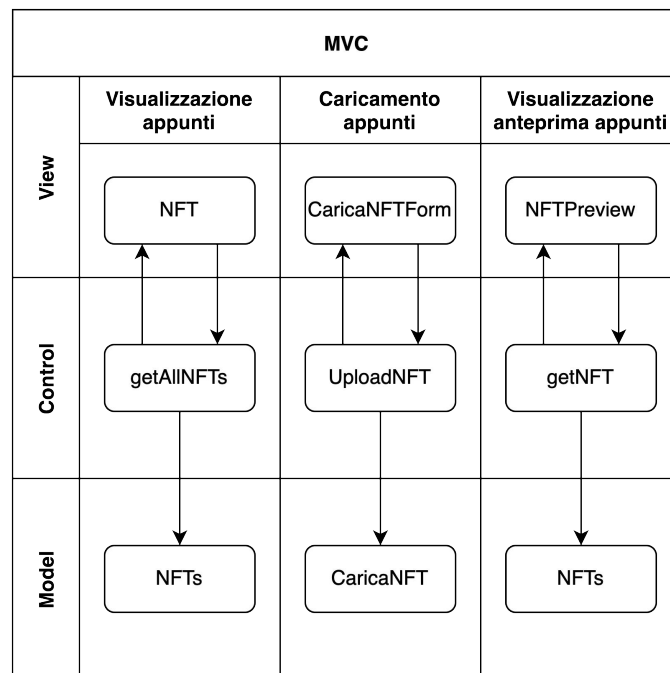
- **Seed phrase:** Una sequenza di 12-24 parole che permette l'accesso a tutte le chiavi ed indirizzi del wallet.

Mentre la classe StudentsNoteNFT conterrà un counter per identificare univocamente ogni token ed un metodo mint, per creare un token su una determinata blockchain al fine di renderlo acquistabile da altri utenti (blo [2022]).



**Figura 3.2:** Class diagram

Inoltre, è stato scelto di utilizzare il pattern architetturale MVC come mostrato in figura 3.3, in quanto fornisce la separazione dei componenti e la parallelizzazione della programmazione. Uno sviluppatore potrebbe lavorare sulla view mentre un altro sul model. Ciò rende lo sviluppo più veloce e manutenibile anche in sviluppi futuri.

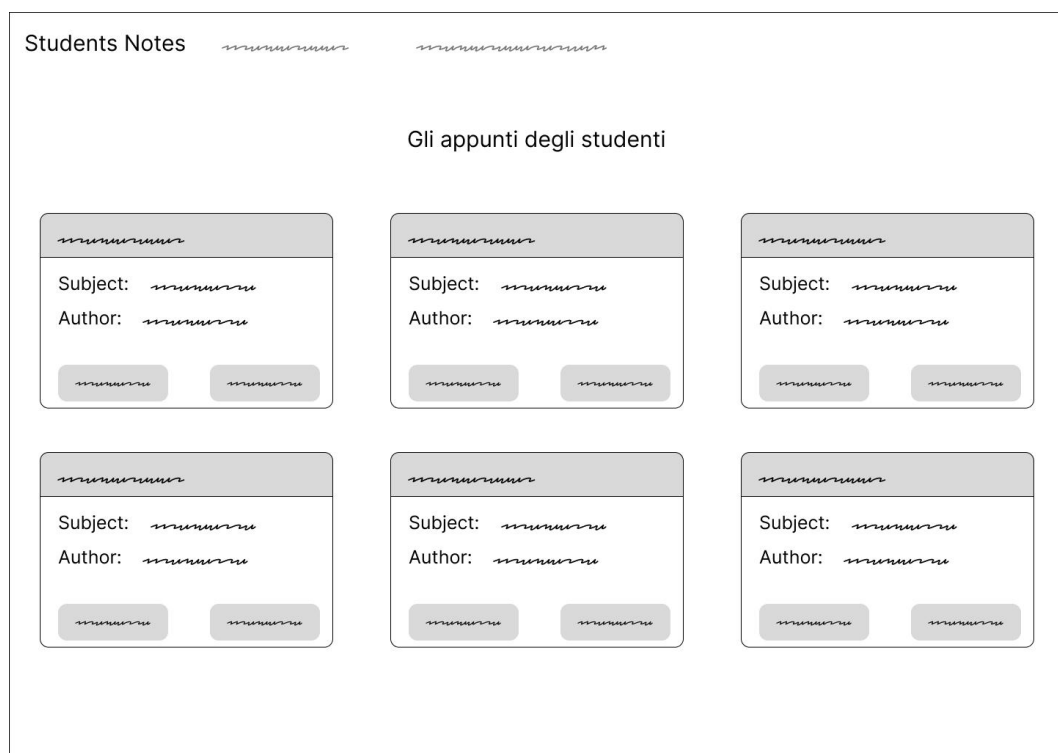


**Figura 3.3:** Model View Controller

## 3.4 Mockups

Per avere una chiara idea di come dovrà essere il front-end dell'applicativo, sono stati creati dei mock-up, i quali sono serviti anche per avere un'idea più chiara di quali devono essere i dati disponibili all'utente e in che modo dovranno poi essere mostrati. In particolare, per la fase della creazione dei mock-up, si è proceduto suddividendo l'attività in tre fasi: creazione del Lo-Fi prototype, creazione del Mi-fi prototype e creazione dell'Hi-Fi prototype.

**Lo-Fi prototype.** I Lo-fi prototype sono una rappresentazione di uno o più wireframe che permettono di raccogliere rapidamente feedback e suggerimenti da parte degli utenti, e nel mio caso ho utilizzato Figma per crearli. Nel mio caso, ho creato un Lo-fi prototype per la home page, come è possibile vedere in figura 3.4 e uno per la pagina per caricare gli appunti in figura 3.5.



**Figura 3.4:** Lo-fi prototype della home page

**Mi-Fi prototype.** Il mi-fi prototype è un prototipo con delle aree cliccabili che presentano le interazioni e le potenzialità di navigazione di un'applicazione. Grazie a un prototipo di media fedeltà, diventa facile presentare le trame e validare l'interazione. Questi mi-fi prototype sono stati costruiti sulla base delle user stories viste in precedenza. I miei mi-fi prototype

The image shows a lo-fi prototype of a web page titled "Students Notes". The page has a header with the title and two wavy lines representing a logo or decorative element. Below the header, there are four input fields with labels: "Carica documento", "Titolo", "Autore", and "Materia". Each label is followed by a gray rectangular box representing the input field. At the bottom of the form, there is a rounded rectangular button with a wavy line icon, representing a submit or upload button.

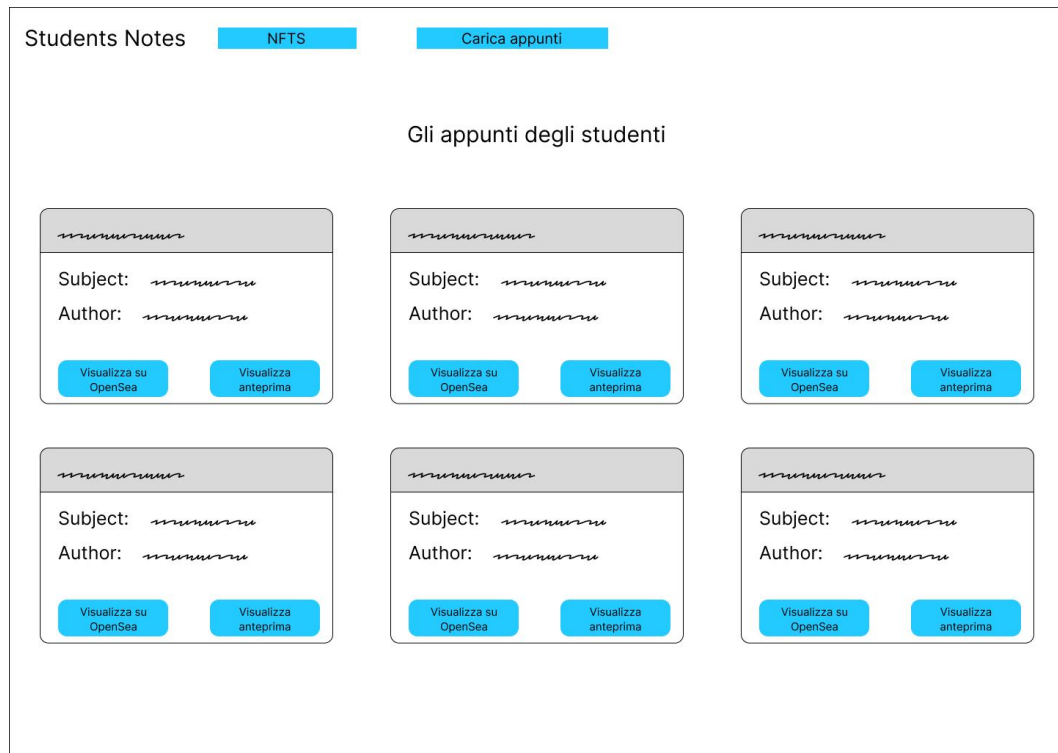
**Figura 3.5:** Lo-fi prototype della pagina per caricare gli appunti

hanno evoluto il lo-fi prototype mettendo in chiaro il testo dei pulsanti, ed evidenziano le aree cliccabili dall'utente. Di seguito verranno illustrati il mi-fi prototype della home page come è possibile vedere in figura 3.6 e della pagina per caricare gli appunti in figura 3.7.

**Hi-Fi prototype.** Gli hi-fi prototype sono prototipi con i quali l'utente può interagire come i mi-fi, ma a differenza di questi ultimi, sono molto vicini a quello che sarà prodotto finale, con la maggior parte degli componenti necessari sviluppati e integrati. Questi prototipi saranno poi utilizzati nelle fasi successive per testare l'usabilità, la quale è fondamentale affinché chiunque possa utilizzare il sistema a prescindere dalle disabilità. In questa fase ho progettato l'hi-fi della home come è possibile vedere in figura 3.8 e della pagina per caricare gli appunti in figura 3.9.

## 3.5 Implementazione

Per l'implementazione si è utilizzato Truffle per compilare e deployare sulla blockchain gli smart contract, React JS per sviluppare il front-end ed infine Node JS per chiamare le API. Il mio lavoro è diviso in due parti, la prima parte è dedicata alla scrittura ed al deploy dello smart contract, la seconda invece è dedicata allo sviluppo del frontend della Dapp. Per fare



**Figura 3.6:** Mi-fi prototype della home page

ciò ho seguito il tutorial ufficiale di Infura il quale è un sistema che permette alle dApp di processare informazioni sulla blockchain di Ethereum senza avere un full node installato sul proprio device. (Inf [2022a]), infatti prima di iniziare sarà necessario creare un progetto IPFS, cioè un progetto per operare su IPFS, un protocollo peer-to-peer e open source su Infura.

### 3.5.1 Creazione e deploy dello smart contract

Prima di iniziare dobbiamo installare tutte le dipendenze con gli appositi comandi da terminale:

- **Truffle:** `npm install -g truffle`, cioè ambiente di sviluppo che lavora sulla blockchain di Ethereum.
- **Dotenv:** `npm install dotenv`, per salvare i valori di configurazione;

Per prima cosa ho creato una cartella dove inizializzare il progetto con Truffle, che andrà a creare tutti le cartelle necessariie per il progetto. Poi ho installato OpenZeppelin, cioè una libreria che nasce con lo scopo di fornire un set di Smart Contracts Open Source per utilizzare codice sicuro, evitando di reinventare la ruota e commettere errori, con il comando `npm install -save-dev @openzeppelin/contracts`, che aiuta a minimizzare i

Students Notes    NFTS    Carica appunti

Carica documento

Titolo

Autore

Materia

Carica

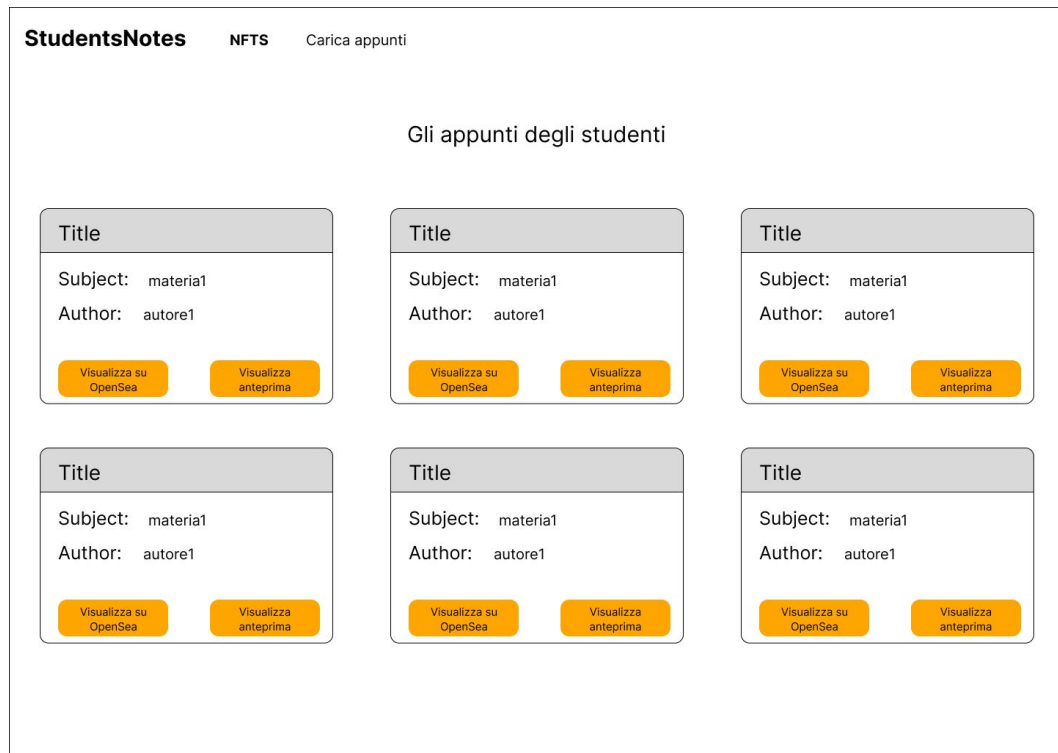
**Figura 3.7:** Mi-fi prototype della pagina per caricare gli appunti

rischi utilizzando librerie testate di smart contract per Ethereum e altre blockchain (Ope [2022]). Include le implementazioni più utilizzate degli standard ERC, come ERC-20 (Eth [2022a]) ed ERC-721 (Eth [2022b] Ope [2022]), di cui ho utilizzato l'implementazione per lo standard ERC-721 (Eth [2022b]).

Lo smart contract scritto in Solidity è un Ownable, cioè fornisce un meccanismo d'accesso e di controllo base, dove il proprietario ha accesso a delle funzioni esclusive, ed un ERC721URIStorage che offre un modo per immagazzinare i metadati del token. Come parametro ha un Counter che ha il ruolo di assegnare un id univoco ad ogni NFT caricato sulla blockchain, ed una funzione di mint per caricare il token su una determinata blockchain al fine di renderlo acquistabile da altri utenti (blo [2022]).

---

```
//Contract based on
//[https://docs.openzeppelin.com/contracts/4.x/erc721]
//([https://docs.openzeppelin.com/contracts/4.x/erc721])
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
```



**Figura 3.8:** Hi-fi prototype della home page

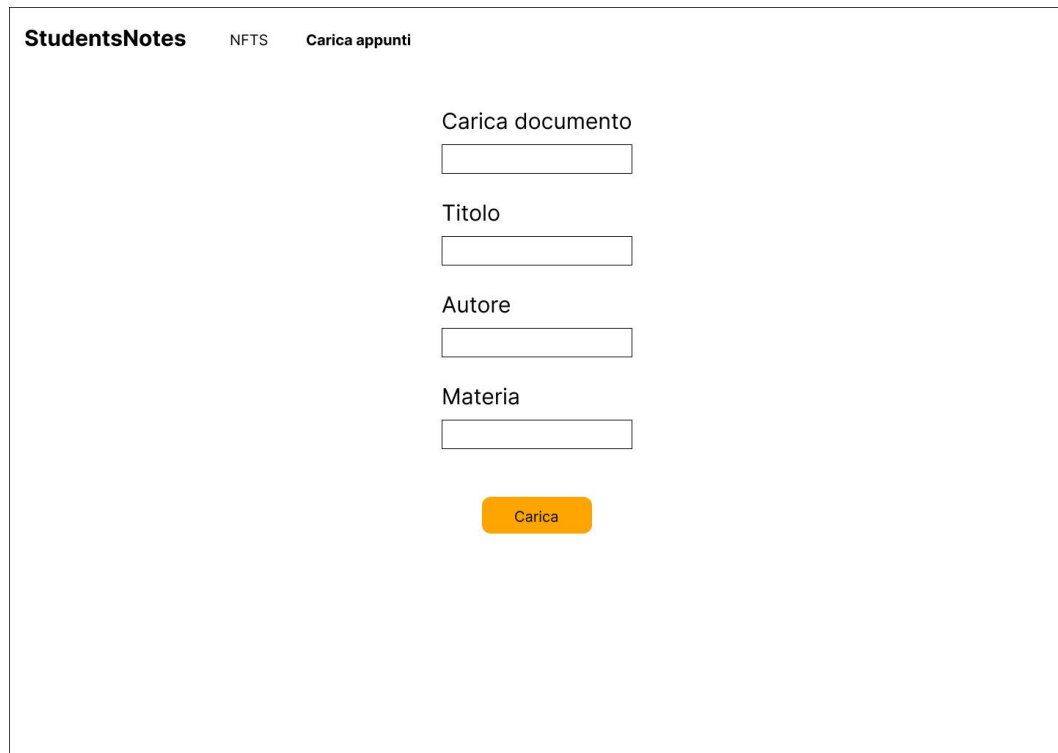
```
contract StudentNoteNFT is ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("Student note", "NFT") {}

    function mintNFT(address recipient, string memory tokenURI)
        public
        returns (uint256)
    {
        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _safeMint(recipient, newItemId);
        _setTokenURI(newItemId, tokenURI);

        return newItemId;
    }
}
```



The image shows a web form titled 'StudentsNotes' with a sub-header 'Carica appunti'. The form contains four text input fields with labels: 'Carica documento', 'Titolo', 'Autore', and 'Materia'. Below these fields is an orange button labeled 'Carica'.

**Figura 3.9:** Hi-fi prototype della pagina per caricare gli appunti

Per deployare lo smart contract ho creato un file `.env` nella directory principale del progetto è l'ho configurato come segue:

---

```
INFURA_API_KEY = "wss://goerli.infura.io/ws/v3/<Your-API-Key>"
MNEMONIC = "<Your-MetaMask-Secret-Recovery-Phrase>"
```

---

Una volta creato lo smart contract, ho configurato il file `truffle-config.json` al fine di utilizzare l'`hdwallet-provider`, utilizzato per firmare transazioni per indirizzi derivati da un mnemonico di 12 o 24 parole. In particolare bisognerà pagare le gas fees e deployare lo smart contract sulla testnet Goerli, cioè una rete di Ethereum. Il file `truffle-config.json` l'ho configurato come segue:

---

```
require('dotenv').config()
const HDWalletProvider = require('@truffle/hdwallet-provider')
const { INFURA_API_KEY, MNEMONIC } = process.env;

module.exports = {
  networks: {
    development: {
      host: '127.0.0.1',
      port: 8545,
```



```
    network_id: '*'
  },
  goerli: {
    provider: () => new HDWalletProvider(MNEMONIC, INFURA_API_KEY),
    network_id: 5,
    gas: 5500000,
  }
},
compilers: {
  solc: {
    version: '^0.8.0'
  }
}
};
```

---

Quindi ho compilato il contratto. Per deployare il contratto su una rete, ho definito un deployment script.

---

```
const NFT_Contract = artifacts.require("StudentNoteNFT");

module.exports = function(deployer) {
  deployer.deploy(NFT_Contract);
};
```

---

Infine ho deployato lo script pagando le gas fees per la transazione. A questo punto ho salvato l'indirizzo dello smart contract, per poi interagirci con la Dapp.

### 3.5.2 Creazione, pubblicazione e visualizzazione degli NFT da React

La Dapp è stata sviluppando utilizzando React Js e Node JS. È proprio quest ultimo che andremo ad approfondire per primo, perché ci permetterà di visualizzare tutti gli NFT mintati dal nostro smart contract.

**Progetto Node JS.** Per prima cosa ho creato un folder, dove ho inizializzato il mio progetto Node ed ho installato i seguenti framwork:

- **Express:** npm install express –save, per gestire le richieste get ad un determinato URL, nel mio caso http://localhost:3000;
- **Moralis:** npm install moralis, che fornisce le API per ottenere gli NFT di un determinato contratto;

- **CORS:** npm install cors, che permette di allentare la sicurezza applicata a un'API, rendendola accessibile.

---

```
const express = require("express");
const Moralis = require("moralis").default;
const app = express();
const cors = require("cors");
const port = 3000;
app.use(cors());
app.use(express.json());
Moralis.start({
  apiKey: <MORALIS-API-Key>,
}).then(() => {
  app.listen(port, () => {
    console.log(`Listening for API Calls`);
  });
});
app.get("/allNft", async (req, res) => {
  try {
    const { query } = req;
    let NFTs;
    if (query.cursor) {
      NFTs = await Moralis.EvmApi.nft.getContractNFTs({
        address: "0x29B1094261a2591B83B092c5CB82aE128d5534a4",
        chain: "0x5", // GOERLI
        cursor: query.cursor,
        limit: 20,
      });
    } else {
      NFTs = await Moralis.EvmApi.nft.getContractNFTs({
        address: "0x29B1094261a2591B83B092c5CB82aE128d5534a4",
        chain: "0x5", // GOERLI
        limit: 20,
      });
    }
    const result = NFTs.raw;
    return res.status(200).json({ result });
  } catch (e) {
    console.log(e);
    console.log("something went wrong");
    return res.status(400).json();
  }
});
```

```
}  
});
```

---

Una volta che ho creato il servizio di API, ho programmato il frontend. Per la user interface ho utilizzato React-Bootstrap, che fornisce i mezzi per fare delle UI con componenti già creati e modellabili a piacimento. L'app si compone principalmente di due pagine: Lista degli NFT e Carica NFT. Non si prevede una pagina di login in quanto sulla blockchain non esistono profili utente.

**Browse NFT.** Questa pagina dispone del metodo `fetchNFTs`, il quale viene invocato al caricamento della pagina e alla pressione del tasto per caricare più risultati. Esso fa una richiesta di tipo GET a Node, il quale restituirà la lista degli NFT collegati al mio contratto.

---

```
async function fetchNFTs () {  
  let res;  
  if (cursor) {  
    res = await axios.get(`http://localhost:3000/allNft`, {  
      params: { cursor: cursor },  
    });  
  } else {  
    res = await axios.get(`http://localhost:3000/allNft`);  
  }  
  let n = NFTs;  
  setNFTs(n.concat(res.data.result.result));  
  setCursor(res.data.result.cursor);  
}
```

---

**Crea NFT.** Questa pagina disporrà di due metodi: uno per recuperare il file contenente gli appunti ed uno per caricare l'NFT. Il metodo per caricare il file è il seguente:

---

```
const retrieveFile = (e) => {  
  const data = e.target.files[0];  
  const reader = new window.FileReader();  
  console.log(reader)  
  reader.readAsArrayBuffer(data);  
  reader.onload = () => {  
    console.log("Buffer data: ", Buffer.from(reader.result, "Uint8Array"));  
  }  
  reader.onloadend = () => {  
    setFile(Buffer.from(reader.result, "Uint8Array"));  
  }  
}
```

```

    }
    e.preventDefault();
  }

```

---

Il metodo per caricare l’NFT invece è diviso in più parti:

---

```

// Controllo se tutti i campi sono stati riempiti correttamente.
// Carico l'NFT ed i suoi metadati su IPFS.
const created = await client.add(file);
const url = `https://students-notes.infura-ipfs.io/ipfs/\${created.path}`;
if(!urlArr.some(u => u === url) ){
  setUrlArr(prev => [...prev, url]);
  var updated_nft_data = {
    "name": title,
    "description": "",
    "image": url,
    "attributes": [
      {
        "trait_type": "Author",
        "value": author
      },
      {
        "trait_type": "Subject",
        "value": subject,
      }
    ]
  }
  const toSend = JSON.stringify(updated_nft_data)
  const created2 = await client.add(toSend);
  const metadataurl =
    `https://students-notes.infura-ipfs.io/ipfs/\${created2.path}`;

  // Richiedo il wallet per pagare la transazione.
  // Access our wallet inside of our dapp
  const web3 = new Web3(Web3.givenProvider);
  await web3.eth.net.getNetworkType()
  // Contract address of the deployed smart contract
  const storageContract = new web3.eth.Contract(StudentNote,
    web3.utils.toChecksumAddress("0x29B1094261a2591B83B092c5CB82a
E128d5534a4"));
  const accounts = await window.ethereum.enable();
  const account = accounts[0];

```

---

```

// Get permission to access user funds to pay for gas fees
const gas = await storageContract.methods.mintNFT(account,
    metadataurl).estimateGas();
await storageContract.methods.mintNFT(account, metadataurl).send({
    from: account,
    gas,
})
// Verifico se la transazione sia andata a buon fine
.on('transactionHash', (txhash) => {
    setOutput(`Mining transaction ...`)
    setContractUrl(`https://goerli.etherscan.io/tx/${txhash}`)
})
.on('error', function(error){
    setOutput(`Errore: ${error}`)
})
.then(function(receipt){
    setOutput(`Success!! The NFT has been minted and mined in block
        \${receipt.blockNumber}`)
    // Success, you've minted the NFT. The transaction is now on chain!
});
}

```

---

## 3.6 Testing

Per eseguire i test è stato utilizzato Selenium, per la precisione Selenium IDE, l'espansione per Chrome. Il testing è stato un testing di sistema, ed è stato eseguito su tutti i requisiti funzionali individuati nella prima fase. Di seguito sono elencati tutti i test eseguiti con annessi risultati. L'ultimo in particolare, è il test che ha verificato che l'integrazione tra le varie componenti è andata a buon fine.

**Caricamento dell'NFT.** In questo test ho verificato che il caricamento dell'NFT andasse a buon fine, compilando i campi che andranno poi a definire i metadati dell'NFT. Il risultato è stato positivo in quanto è stato possibile caricare l'NFT con i dati in input com'è possibile vedere in figura 3.10.

**Visualizzare un' anteprima degli appunti.** In questo test è stato verificato che l'utente visualizzi correttamente un' anteprima gli appunti cliccando il pulsante "Visualizza antepri-

```
Running 'carica nft'
1. open on http://localhost:3001/ OK
2. setWindowSize on 1440x804 OK
3. click on linkText=Carica NFT OK
4. mouseOver on linkText=Carica NFT OK
5. mouseOut on linkText=Carica NFT OK
6. click on name=data OK
7. type on name=data with value C:\fakepath\3_creazione_meeting.pdf OK
8. type on css=.input:nth-child(4) > input with value Luigi Allocca OK
9. type on css=.input:nth-child(6) > input with value IUMUS OK
10. type on css=.input:nth-child(8) > input with value Creazione meeting OK
11. sendKeys on css=.input:nth-child(8) > input with value ${KEY_ENTER} OK
```

**Figura 3.10:** Testing caricamento NFT

ma". Il risultato è stato un successo, in quanto la pagina alla quale sono stato indirizzato, corrispondeva con quella aspettata com'è possibile vedere in figura 3.11.

```
Running 'visualizza anteprima appunti'
1. open on http://localhost:3001/ OK
2. setWindowSize on 1440x804 OK
3. Trying to find css=.col:nth-child(2) .btn:nth-child(2)... OK
4. assertTitle on QmaW3YYyaNEwJybdjERRm7C78CZwjdwvKTH2rsTH3H3orV (100x100) OK
'visualizza anteprima appunti' completed successfully
```

**Figura 3.11:** Testing visualizzazione anteprima appunti

**Visualizzare l’NFT su OpenSea.** In questo test è stato verificato che l’utente visualizzi l’NFT su OpenSea tramite il pulsante "Visualizza su OpenSea". Il risultato è stato un successo, in quanto la pagina alla quale sono stato indirizzato, corrispondeva con quella aspettata com'è possibile vedere in figura 3.12.

```
Running 'visualizza nft su opensea'
1. open on http://localhost:3001/ OK
2. setWindowSize on 1440x804 OK
3. Trying to find css=.col:nth-child(2) .btn:nth-child(1)... OK
4. assertTitle on 1 - My meme NFT - Jj40rhXxHE | OpenSea OK
'visualizza nft su opensea' completed successfully
```

**Figura 3.12:** Testing visualizzazione NFT su OpenSea

**Caricamento NFT e visualizzazione in home page.** In questa fase di testing sono stati testati i task di caricamento dell’NFT e di visualizzazione dell’anteprima degli appunti. Il risultato è stato un successo, in quanto dopo aver caricato con successo l’NFT, esso era visibile

nella pagina NFTs, ed una volta cliccato su "Visualizza anteprima", sono stato riportato alla pagina attesa (vedere la figura 3.13).

```
Running 'caricamento e visualizzazione dell' NFT'
1. open on http://localhost:3001/ OK
2. setWindowSize on 1440x804 OK
3. click on linkText=Carica NFT OK
4. click on name=data OK
5. type on name=data with value C:\fakepath\3_creazione_meeting.pdf OK
6. type on css=.input:nth-child(4) > input with value Luigi Allocca OK
7. type on css=.input:nth-child(6) > input with value IUMUS - Vitiello OK
8. type on css=.input:nth-child(8) > input with value Creazione meeting - MetaClass OK
9. click on css=.btn OK
10. click on linkText=NFTs OK
11. Trying to find css=.btn:nth-child(3)... OK
12. Trying to find css=.col:nth-child(19) .btn:nth-child(2)... OK
13. assertTitle OK
'caricamento e visualizzazione dell' NFT' completed successfully
```

**Figura 3.13:** Testing caricamento e visualizzazione NFT

## 4.1 The Application Binary Interface (ABI)

Per eseguire una funzione di un programma compilato in EVM è necessario conoscere i nomi e i tipi precisi associati alle operazioni. L'ABI definisce i metodi e le strutture utilizzate per interagire con il contratto ed indica al chiamante della funzione di codificare le informazioni necessarie, come le firme delle funzioni e le dichiarazioni delle variabili, in un formato che l'EVM può comprendere per chiamare la funzione in bytecode (Qui [2022]).

## 4.2 VSCode e Solidity

Per scrivere lo smart contract in Solidity e creare il progetto, come editor è stato utilizzato VS Code. Questa scelta è stata dovuta dal fatto che Truffle offre un servizio CLI molto vantaggioso se utilizzato con VSCode, in quanto ad esempio permette di creare un progetto con un semplice comando "truffle init", oppure si può deployare uno smart contract con un comando.

## 4.3 Truffle

Truffle è un ambiente di sviluppo, un framework di test e una pipeline di asset di livello mondiale per le blockchain che utilizzano la macchina virtuale di Ethereum (EVM), con l'obiettivo di semplificare la vita degli sviluppatori (kal [2022]).



Nel mio caso in particolare è stato utile per creare il progetto di uno smart contract, compilarlo e deployarlo successivamente sulla testnet di Goerli, una fork di Ethereum utilizzata prettamente per il testing.

## 4.4 Infura

Infura mi ha permesso di creare l'applicazione in fase di test con un accesso semplice e affidabile a Ethereum e IPFS. (Inf [2022b]). In particolare nella mia DApp ho utilizzato Infura come tramite, per caricare gli NFT e i loro metadati su IPFS. Questa soluzione mi ha permesso di ridurre notevolmente i tempi di sviluppo garantendo un'ottima qualità in termini anche di esperienza utente al momento dell'utilizzo del sistema.

## 4.5 IPFS

IPFS è un sistema distribuito per l'archiviazione e l'accesso a file, siti web, applicazioni e dati (IPF [2022]). Nella mia Dapp, ho utilizzato IPFS per caricare gli NFT ed i metadati associati, attraverso Infura. Questa soluzione è risultata molto produttiva, in quanto il tempo di sviluppo e la difficoltà dell'implementazione sono stati molto ridotti.

## 4.6 Goerli test network

Goerli è una testnet web3 che ho utilizzato per testare l'applicazione. (alc [2022]).

## 4.7 React JS e Node.js

Per il front-end della mia Dapp ho utilizzato React JS, mentre per la chiamata alle API ho utilizzato Node.js. Da notare come Node.js di solito è utilizzato come back-end delle applicazioni, ma nel nostro caso non esiste un vero e proprio back-end visto che stiamo parlando di un'applicazione decentralizzata.

React.js è un framework e una libreria JavaScript open-source sviluppata da Facebook (hub [2022]).

Node.js è stato utilizzato invece per chiamare le API di Moralis.

## 4.8 Moralis

Moralis è un Development Workflow utilizzato lo sviluppo Web3 che fornisce una suite completa di API per tutto ciò che riguarda la blockchain (Mor [2022]). Nel caso della mia Dapp, Morailis è stato utilizzato per mostrare nella schermata principale, i gli attributi dell’NFT, quali il titolo, l’autore, e la materia. Questa informazioni sono state estrapolate dal JSON contenenti i metadata presente su IPFS. Moralis è stato utilizzato da Node.js per fare delle chiamate ed ottenere le informazioni da mostrare all’utente. Successivamente sarà mostrato come è stato utilizzato nel codice.

## 4.9 Librerie e frameworks utilizzati

Tra le librerie utilizzate troviamo:

- **Axios:** Come riportato dal sito di codebots: Axios è un client HTTP per JavaScript basato su promise. È in grado di effettuare richieste HTTP dal browser e di gestire la trasformazione dei dati di richiesta e risposta (cod [2022]). Nella mia soluzione Axios è stato utilizzato per effettuare le chiamate a node ed ottenere una risposta con un codice e ud un messaggio. Successivamente sarà disponibile lo script che lo utilizza.
- **Express JS:** Express è un framework per applicazioni web node js che offre ampie funzionalità per la creazione di applicazioni web a pagina singola, multipagina e ibride e mobile (sim [2022]).

## 4.10 OpenSea

OpenSea è un marketplace di NFT. Gli appunti potranno dunque essere acquistati e venduti dagli utenti attraverso questa piattaforma garantendo la sicurezza delle transazioni e la possibilità di raggiungere più utenti interessati.

L'obiettivo della tesi è analizzare la blockchain e sviluppare un applicativo nel linguaggio di programmazione Solidity. Il risultato del mio lavoro di tesi è una Dapp funzionante, che funge da marketplace di appunti di studenti, dove gli utenti possono agire sia venditore che acquirente. Gli sviluppi futuri prevedono la possibilità di acquistare e vendere gli appunti con una criptovaluta per l'acquisto e la vendita degli appunti direttamente dall'applicazione ed un miglioramento dell'interfaccia utente. Non si esclude inoltre il passaggio ad un'altra blockchain come Solana (<https://solana.com/>) per transazioni più veloci ed economiche.

---

## Ringraziamenti

---

Ringrazio la mia famiglia che mi è sempre stata vicina, sia nei momenti belli che in quelli difficili, ed ha sempre creduto in me nonostante tutto.

Un ringraziamento va anche alla famiglia della mia ragazza, che mi ha sempre sostenuto e mi ha dato consigli che mi hanno accompagnato per questi tre anni.

Ringrazio il mio amico Francesco che più che un amico è stato un punto di riferimento, una persona sulla quale ho sempre potuto contare.

Ringrazio il mio relatore Dario Di Nucci, perché è sempre stato disponibile e mi ha guidato per tutto il periodo della scrittura della tesi.

Un ringraziamento speciale va alla mia ragazza, che è stata sempre al mio fianco in ogni momento.

*"To W.W. My Star, My Perfect Silence."*

---

## Bibliografia

---

- Gianluca Chiap, Jacopo Ranalli, and Raffaele Bianchi. *Blockchain. Tecnologia e applicazioni per il business: Tutto ciò che serve per entrare nella nuova rivoluzione digitale*. HOEPLI EDITORE, 2019. (Citato alle pagine 3, 4, 5, 6 e 8)
- cryptonomist aziende che utilizzano blockchain. <https://cryptonomist.ch/2022/03/19/quali-grandi-aziende-utilizzano-blockchain-crypto/>, 2022. Accessed: 2022-11-25. (Citato a pagina 4)
- Mengjiao Wang, Hikaru Ichijo, and Bob Xiao. Cryptocurrency address clustering and labeling. *arXiv preprint arXiv:2003.13399*, 2020. (Citato a pagina 4)
- Matthew Johnson, Michael Jones, Mark Shervy, Joel T Dudley, Noah Zimmerman, et al. Building a secure biomedical data sharing decentralized app (dapp): tutorial. *Journal of medical Internet research*, 21(10):e13601, 2019. (Citato a pagina 7)
- Maximilian Wohrer and Uwe Zdun. Smart contracts: security patterns in the ethereum ecosystem and solidity. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 2–8, 2018. doi: 10.1109/IWBOSE.2018.8327565. (Citato a pagina 7)
- Yoichi Hirai. Defining the ethereum virtual machine for interactive theorem provers. In *International Conference on Financial Cryptography and Data Security*, pages 520–535. Springer, 2017. (Citato alle pagine 7 e 8)
- Shailak Jani. An overview of ethereum & its comparison with bitcoin. *Int. J. Sci. Eng. Res*, 10(8):1–6, 2017. (Citato a pagina 8)

- Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. Non-fungible token (nft): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*, 2021. (Citato a pagina 8)
- Simone Casale-Brunet, Paolo Ribeca, P Doyle, and Marco Mattavelli. Networks of ethereum non-fungible tokens: a graph-based analysis of the erc-721 ecosystem. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 188–195. IEEE, 2021. (Citato a pagina 8)
- blockchainftitalia mint. <https://blockchainftitalia.com/mint-nft/#:~:text=Mint>, 2022. Accessed: 2022-11-25. (Citato alle pagine 13 e 17)
- Infura upload nft metadata to ipfs. <https://docs.infura.io/infura/tutorials/ethereum/create-an-nft-using-truffle/upload-nft-metadata-to-ipfs>, 2022a. Accessed: 2022-11-22. (Citato a pagina 16)
- OpenZeppelin contracts. <https://docs.infura.io/infura/tutorials/ethereum/create-an-nft-using-truffle/upload-nft-metadata-to-ipfs>, 2022. Accessed: 2022-11-22. (Citato a pagina 17)
- Ethereum token erc-20. <https://ethereum.org/it/developers/docs/standards/tokens/erc-20/>, 2022a. Accessed: 2022-11-25. (Citato a pagina 17)
- Ethereum token erc-721. <https://ethereum.org/it/developers/docs/standards/tokens/erc-721/>, 2022b. Accessed: 2022-11-25. (Citato a pagina 17)
- QuickNode what is abi. <https://www.quicknode.com/guides/smart-contract-development/what-is-an-abi>, 2022. Accessed: 2022-11-21. (Citato a pagina 27)
- kaleido truffle description. <https://www.kaleido.io/blockchain-platform/truffle#:~:text=Truffle>, 2022. Accessed: 2022-11-21. (Citato a pagina 27)
- Infura description. <https://www.infura.io/faq/general>, 2022b. Accessed: 2022-11-21. (Citato a pagina 28)
- IPFS what is ipfs. <https://docs.ipfs.tech/concepts/what-is-ipfs/>, 2022. Accessed: 2022-11-21. (Citato a pagina 28)
- alchemy aziende che utilizzano blockchain. <https://www.alchemy.com/overviews/goerli-faucet#:~:text=Goerli>, 2022. Accessed: 2022-11-25. (Citato a pagina 28)

hubspot what is react js. <https://blog.hubspot.com/website/react-js>, 2022.  
Accessed: 2022-11-21. (Citato a pagina 28)

Moralis what is. <https://moralis.zendesk.com/hc/en-us/articles/4651400933010-What-is-Moralis->, 2022. Accessed: 2022-11-21. (Citato a pagina 29)

codebots what is axios. <https://codebots.com/docs/what-is-axios#:~:text=Axios>, 2022. Accessed: 2022-11-21. (Citato a pagina 29)

simplilearn what is express. <https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js#:~:text=Express>, 2022. Accessed: 2022-11-21. (Citato a pagina 29)