

**Università degli Studi di Padova**

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA "

CORSO DI LAUREA IN INFORMATICA



**Analisi e sviluppo front-end di una web  
application in ambito blockchain/NFT**

*Tesi di laurea*

*Relatore*

Prof. Gilberto Filè

*Laureando*

Margherita Mitillo

---

ANNO ACCADEMICO 2020-2021



Chi non ha visto il calar della notte non giuri d'inoltrarsi nelle tenebre.

— John Ronald Reuel Tolkien

Dedicato a Daniele, Alessandra, Gianni e Luciano



# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Margherita Mitillo presso l'azienda Sync Lab S.r.l. supervisionato e coordinato dal tutor aziendale Fabio Scettro. Lo scopo principale era lo sviluppo di maschere per il front end (ovvero interfacce utente), dell'applicazione NFTLab tramite il framework Vue.js.

Oltre a questo era previsto lo studio delle tecnologie coinvolte per la progettazione e la codifica del prodotto. Infine era richiesta l'implementazione di tali interfacce e la stesura di un documento tecnico che raccolga la descrizione di ciò che è stato implementato.



*“Certe strade, è meglio intraprenderle che rifiutarle, anche se il loro esito è oscuro”*

— John Ronald Reuel Tolkien

# Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Gilberto Filè, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.*

*Ho desiderio di ringraziare il mio tutor aziendale Fabio Scettro per avermi seguito durante il periodo di stage.*

*Ringrazio con tanto affetto il mio ragazzo Daniele per avermi supportato e aver sempre creduto in me durante tutto periodo di studi. Senza di lui probabilmente non sarei arrivata fino a qui.*

*Desidero ringraziare con affetto i miei genitori Alessandra, Luciano e Gianni per il sostegno, e per essermi stati vicini in ogni momento durante gli anni di studio.*

*Ho desiderio di ringraziare poi i miei amici, in particolare Alberto, per i bellissimi anni passati assieme.*

*Padova, Luglio 2021*

Margherita Mitillo





# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Organizzazione del testo . . . . .	1
1.2	Regole tipografiche . . . . .	1
<b>2</b>	<b>Descrizione dello stage</b>	<b>3</b>
2.1	L'azienda . . . . .	3
2.1.1	La storia . . . . .	3
2.1.2	I prodotti . . . . .	3
2.2	L'idea dello stage . . . . .	4
2.3	Organizzazione del lavoro . . . . .	5
2.3.1	Modello di sviluppo . . . . .	5
2.3.2	Organizzazione del lavoro . . . . .	6
<b>3</b>	<b>Nozioni apprese</b>	<b>9</b>
3.1	Organizzazione dello studio . . . . .	9
3.2	Nozioni principali . . . . .	9
3.2.1	Javascript . . . . .	9
3.2.2	Vue.js . . . . .	9
3.3	Nozioni secondarie . . . . .	10
3.3.1	Java . . . . .	10
3.3.2	Concetti web . . . . .	10
3.3.3	Spring . . . . .	11
3.3.4	Blockchain . . . . .	13
3.3.5	Ethereum . . . . .	13
3.3.6	NFT . . . . .	13
<b>4</b>	<b>Analisi dei requisiti</b>	<b>15</b>
4.1	Casi d'uso . . . . .	15
4.1.1	Attori dei casi d'uso . . . . .	15
4.2	Tracciamento dei requisiti . . . . .	19
<b>5</b>	<b>Progettazione e codifica</b>	<b>21</b>
5.1	Tecnologie e strumenti . . . . .	21
5.1.1	Tecnologie . . . . .	21
5.1.2	Strumenti . . . . .	23
5.2	Ciclo di vita del software . . . . .	25
5.3	Progettazione . . . . .	25
5.4	Design Pattern utilizzati . . . . .	25
5.5	Codifica . . . . .	25

<b>6</b>	<b>Verifica e validazione</b>	<b>27</b>
<b>7</b>	<b>Conclusioni</b>	<b>29</b>
7.1	Consuntivo finale . . . . .	29
7.2	Raggiungimento degli obiettivi . . . . .	29
7.3	Conoscenze acquisite . . . . .	29
7.4	Valutazione personale . . . . .	29
<b>A</b>	<b>Appendice A</b>	<b>31</b>
	<b>Glossario</b>	<b>33</b>
	<b>Acronimi</b>	<b>35</b>
	<b>Bibliografia</b>	<b>37</b>

# Elenco delle figure

2.1	Logo di Sync Lab . . . . .	3
2.2	Logo di NFTLab . . . . .	4
2.3	Metodo Scrum . . . . .	6
2.4	Spezzone del report giornaliero . . . . .	7
4.1	Attori dei casi d'uso . . . . .	15
4.2	U.C1.1 Inserimento dei dati per la registrazione . . . . .	16
4.3	Use Case - UC0: Scenario principale . . . . .	18
5.1	Logo di Vue.js . . . . .	21
5.2	Funzionalità di Vuetify . . . . .	23

# Elenco delle tabelle

4.1	Tabella del tracciamento dei requisiti funzionali . . . . .	19
4.2	Tabella del tracciamento dei requisiti qualitativi . . . . .	19
4.3	Tabella del tracciamento dei requisiti di vincolo . . . . .	19



# Capitolo 1

## Introduzione

Lo scopo di questo progetto di stage è studiare il `frameworkG` Vue.js per implementare parte delle componenti di `front endG` di un'applicazione web in ambito Blockchain e `NFTG` chiamata NFTLab.

Tramite questa applicazione sarà possibile creare un proprio profilo per poter compiere diverse azioni come vendere le proprie opere multimediali, modificare alcuni dati di tali opere, visualizzare opere di altri utenti e comprare le stesse da essi.

Per realizzare questi aspetti sono state implementate diverse maschere, ovvero interfacce utente, in collaborazione con gli altri colleghi stagisti.

Infatti un aspetto importante di questo progetto è la collaborazione con gli altri componenti del gruppo per integrare il proprio lavoro con quello degli altri, in particolare per il corretto funzionamento della web application è necessaria l'integrazione tra `front end` e back end.

### 1.1 Organizzazione del testo

**Il secondo capitolo** approfondisce la descrizione dello stage, l'organizzazione del lavoro e il progetto da sviluppare;

**Il terzo capitolo** approfondisce le nozioni apprese nel periodo di stage;

**Il quarto capitolo** approfondisce il processo di analisi dei requisiti e il tracciamento di essi;

**Il quinto capitolo** approfondisce i processi di progettazione e codifica descrivendo anche le tecnologie e gli strumenti utilizzati;

**Il sesto capitolo** approfondisce i processi di validazione e verifica del codice prodotto;

**Il settimo capitolo** descrive le conclusioni tratte alla fine del periodo di stage.

### 1.2 Regole tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- \* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;

- \* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola<sub>G</sub>*;
- \* per le successive occorrenze dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*;
- \* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

## Capitolo 2

# Descrizione dello stage

*In questo capitolo verrà descritto in dettaglio l'idea dello stage, l'azienda dove è stato svolto e come è stato organizzato il lavoro.*

### 2.1 L'azienda

#### 2.1.1 La storia

Sync Lab S.r.l. è un'azienda di consulenza informatica nata nel 2002 nella sede di Napoli e trasformatasi molto velocemente nel corso degli anni in *System Integrator* grazie ad un processo di maturazione delle competenze tecnologiche. L'azienda supporta le esigenze di innovazione dei clienti offrendo soluzioni IT in ambito *Business Consultancy*, *Project Financing* e *IT Consultancy*.



**Figura 2.1:** Logo di Sync Lab

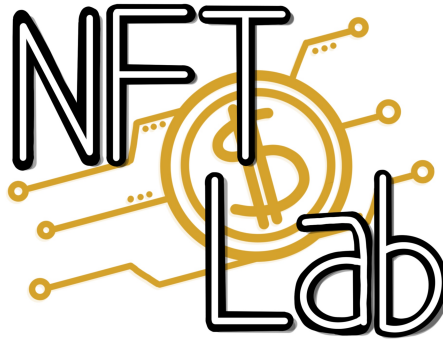
Ad oggi l'azienda conta un organico di circa 200 dipendenti ed è riuscita a coprire tutto il territorio nazionale fino ad ottenere un totale di cinque sedi a Roma, Napoli, Verona, Padova e Milano. L'azienda propone sul mercato prodotti software ed attraverso essi ha gradualmente conquistato significativamente fette di mercato nei seguenti settori: mobile, videosorveglianza e sicurezza delle infrastrutture informatiche aziendali. Il loro obiettivo è quello di supportare il cliente nella Realizzazione, Messa in Opera e *Governance* di soluzioni IT, sia dal punto di vista Tecnologico, sia nel Governo del Cambiamento Organizzativo.

#### 2.1.2 I prodotti

I prodotti più noti dell'azienda sono i seguenti:

- \* **SynClinic**: sistema informativo sanitario per la gestione dei processi clinici ed amministrativi di ospedali, cliniche e case di cura. Lo scopo è diventare uno strumento indispensabile di supporto per il personale nella gestione del rischio clinico di ogni paziente;
- \* **DPS4**: web application per gestire gli adempimenti della privacy GDPR, acronimo per *General Data Protection Regulation*, con lo scopo di rafforzare la privacy dei dati dei cittadini;
- \* **StreamLog**: sistema in grado di effettuare in modo semplice ed efficace il controllo degli accessi degli utenti ai sistemi;
- \* **StreamCrusher**: software in grado di analizzare i dati che aiuta ad essere informati adeguatamente su quando bisogna prendere decisioni di business, ad identificare criticità e riorganizzare il sistema;
- \* **Wave**: acronimo per *Wide Area Videosurveillance Environment*, è un prodotto software nato per ottenere un'integrazione sinergica tra i mondi della sorveglianza e dei Sistemi Informativi Territoriali (GIS) in modo da ottenere una copertura totale del territorio;
- \* **Seastream**: piattaforma in grado di migliorare la sicurezza, l'efficienza e il processo di innovazione in campo marittimo.

## 2.2 L'idea dello stage



**Figura 2.2:** Logo di NFTLab

Il progetto da svolgere proposto dall'azienda per lo stage è una web application chiamata NFTLab legata all'ambito blockchain e [NFT](#), due argomenti che rendono contemporaneo ed interessante il progetto di stage.

In particolare l'utente potrà visualizzare le opere multimediali in vendita, fare l'accesso o la registrazione al sito. Dopo essere acceduto, l'utente potrà caricare le proprie



opere e decidere se metterle in vendita o meno. Nella pagina del profilo personale sarà possibile per l'utente modificare i propri dati personali o i dati delle opere caricate. Oltre a queste azioni l'utente potrà sfogliare il catalogo delle opere presenti e decidere di comprarne a sua volta.

Essendo un progetto legato al concetto di blockchain e [Non-Fungible Token \(NFT\)](#) le azioni di compravendita verranno effettuate tramite lo scambio di una moneta virtuale detta Ethereum. Un utente nel momento in cui carica un'opera diventa l'autore ed il proprietario di quest'ultima, ma dopo la vendita l'acquirente diventerà il nuovo proprietario e solo l'utente che ha caricato l'opera sarà l'autore.

Per realizzare questi aspetti sono state implementate diverse maschere, ovvero interfacce utente, tramite il framework Vue.js, uno strumento utile allo sviluppo di web application reattive e a pagina singola.

Parte del percorso di stage è stata riservata ad un periodo di studio delle tecnologie utili allo sviluppo delle maschere e legate al back end, in particolare è stato studiato il framework Spring. FRASE CHE NON MI PIACE

## 2.3 Organizzazione del lavoro

### 2.3.1 Modello di sviluppo

L'azienda adotta il modello di sviluppo Agile che si basa sull'interazione continua con gli [stakeholder<sub>G</sub>](#) e predilige:

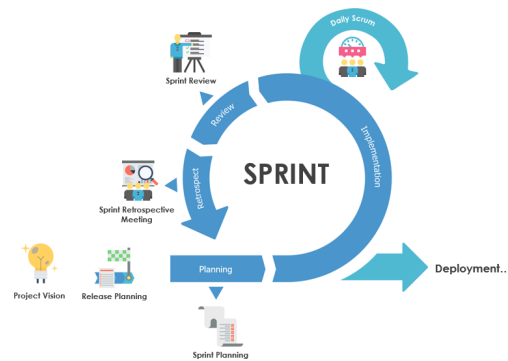
- \* gli individui e le loro interazioni piuttosto che i processi e gli strumenti;
- \* il software funzionante piuttosto che una documentazione esaustiva;
- \* la collaborazione con il cliente piuttosto che la negoziazione dei contratti;
- \* essere propositivi verso il cambiamento piuttosto che rifiutarlo.

L'idea di questo modello non si basa quindi su uno sviluppo sequenziale ma sul concetto di rivedere di continuo le specifiche adeguandole durante l'avanzamento dello sviluppo software. In questo modo si possono apportare agilmente modifiche mediante metodologie iterative ed incremental con un continuo scambio di pareri ed informazioni tra gli sviluppatori ed il committente. Il progetto di stage è stato portato avanti utilizzando il metodo [Scrum<sub>G</sub>](#).

Scrum è il metodo più diffuso e prevede di dividere il progetto in blocchi di lavoro detti sprint, alla fine dei quali si crea un incremento del prodotto software. Esso prevede vari meeting per controllare il lavoro svolto ed organizzare il lavoro da svolgere nel prossimo periodo. Inoltre nel corso dello sviluppo l'azienda organizza diversi meeting con gli [stakeholder](#) e i membri del progetto per valutare l'andamento dello sprint.

Gli incontri legati a questo modello di sviluppo che vengono svolti di solito sono:

- \* sprint planning meeting: tenuto ad inizio sprint per poter organizzare il lavoro;
- \* daily scrum: riunione giornaliera del team di sviluppo per monitorare l'andamento dello sprint;
- \* sprint review: tenuto a fine sprint per valutare i risultati ottenuti e quali cambiamenti apportare per lo sprint successivo.



**Figura 2.3:** Metodo Scrum

La situazione di emergenza sanitaria ancora non conclusa mi ha portato a vivere un'esperienza lavorativa diversa dai colleghi degli anni precedenti, infatti ha fatto sì che lo stage fosse organizzato in modalità mista: in parte da remoto ed in parte in azienda (di solito uno o due giorni alla settimana). A causa di ciò non ho potuto vivere appieno questo modello di sviluppo vista l'impossibilità di essere sempre presente in azienda. Per questo motivo il giorno concordato con i colleghi stagisti ed il tutor aziendale per andare in azienda è stato sfruttato per fare un daily scrum alternativo data la sua cadenza settimanale.

Nonostante l'emergenza sanitaria ancora in corso ho potuto comunque constatare l'efficacia e la funzionalità di questo modello di sviluppo, soprattutto per un progetto software in evoluzione quale è stato quello a cui ho lavorato.

### 2.3.2 Organizzazione del lavoro

#### Trello

Trello è una piattaforma online tramite la quale si può gestire i progetti in modo semplice, gratuito e flessibile. Questa piattaforma risulta essere molto utile per l'organizzazione e la gestione del workflow online.

Tramite questa piattaforma è possibile rappresentare le attività da svolgere in schede inserite sotto apposite colonne di avanzamento. In base a quanto si è riusciti a realizzare, è possibile spostare manualmente o automaticamente lo stato di avanzamento di ogni scheda. DA FINIRE Per tenere traccia delle attività da svolgere è stato deciso di utilizzare la piattaforma online gestionale Trello. Tramite essa l'azienda ha creato due bacheche:

- \* personale: disponibile solo allo stagista ed al tutor aziendale, dove sono stati inseriti i compiti decisi nel piano di lavoro suddivisi per settimana;
- \* collettiva: disponibile a tutti gli stagisti che lavorano allo stesso progetto e ai loro rispettivi tutor aziendali, dove sono state inserite le prime idee e funzionalità legate al progetto.

Nella bacheca personale sono presenti diverse schede di numero pari alle settimane di stage da svolgere mentre nella bacheca collettiva il numero delle schede dipende dal numero di funzionalità e servizi da sviluppare.

### Report giornaliero

#### DA FINIRE

Oltre a tenere traccia delle attività tramite Trello, l'azienda ha deciso di far scrivere allo stagista un file Excel su Drive come report giornaliero delle attività svolte. Il report era suddiviso in quattro colonne:

- \* data: indica il giorno in è stata fatta un'attività in formato AAAA-MM-GG;
- \* descrizione: breve descrizione dell'attività svolta dal tirocinante compilata alla fine di ogni giornata lavorativa;
- \* nota: campo dove il tutor aziendale può aggiungere note di consiglio riguardo all'attività svolta, inoltre in questo campo vengono specificati i giorni festivi;
- \* spunta: casella di spunta dove il tutor indica di aver visionato l'attività svolta.

Giorno	Attività	Nota	Conferma del tutor
2021-05-03	Organizzazione lavoro, ripasso Java e Servlet		ok
2021-05-04	Ripasso del concetto di JSON, inizio dello studio del pattern REST	per approfondire Rest vs Restful: <a href="https://martinfowler.com/articles/richardsonMaturityModel.html">https://martinfowler.com/articles/richardsonMaturityModel.html</a>	ok
2021-05-05	Continuazione studio del pattern REST		ok
2021-05-06	Studiato Rest vs Restful, principi del REST e best practice delle REST API	Consiglio: se non l'hai mai visto dai un'occhiata a postman	ok
2021-05-07	Concluso lo studio di REST con esercizi pratici e ripasso generale delle conoscenze apprese nel corso della settimana		ok

**Figura 2.4:** Spezzone del report giornaliero

### Discord

Per comunicare con l'intero gruppo di stagisti l'azienda ha proposto di utilizzare Discord, una piattaforma messaggistica dove noi stagisti potevamo comunicare sia con il personale aziendale che singolarmente con il proprio tutor per consigli, delucidazioni o motivi organizzativi. DA FINIRE



## Capitolo 3

# Nozioni apprese

*In questo capitolo verranno elencate e spiegate brevemente le nozioni apprese durante questo percorso di stage.*

### 3.1 Organizzazione dello studio

Le ore complessive dello stage possono essere considerate suddivise in 3 periodi:

- \* periodo di studio della durata di circa 160 ore;
- \* periodo di sviluppo della durata di circa 120 ore;
- \* periodo di convalidazione finale della durata di circa 40 ore.

Nelle sezioni qui di seguito verranno elencate e spiegate le nozioni apprese nel periodo di studio.

Per chiarezza sono state suddivise in principali e secondarie, non per dare meno importanza ad alcuni argomenti studiati ma per suddividerli in funzione dello sviluppo del progetto. Infatti le prime mi sono maggiormente servite per poter sviluppare le maschere, mentre le seconde mi sono servite per conoscenza generale e migliore collaborazione con i colleghi addetti al reparto back end.

### 3.2 Nozioni principali

#### 3.2.1 Javascript

Javascript è un linguaggio di programmazione orientato agli oggetti e agli eventi comunemente utilizzato per la programmazione Web lato client. La sua enorme diffusione è dovuta al fiorire di numerose librerie nate per semplificare la programmazione sul browser. DA FINIRE

#### 3.2.2 Vue.js

Vue.js è un framework javascript open-source per la configurazione di interfacce utente e single-page application.

*Questo framework verrà spiegato nel dettaglio nel capitolo riguardante la Progettazione e codifica*

## 3.3 Nozioni secondarie

### 3.3.1 Java

Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica che si appoggia sull'omonima piattaforma software di esecuzione. Java nasce dall'esigenza di risolvere due problemi concreti comuni tra i linguaggi orientati ad oggetti: garantire maggiore semplicità nella scrittura e gestione del codice, e permettere la realizzazione di programmi slegati da un'architettura precisa.

Il primo problema fu risolto utilizzando un sistema basato sulla gestione della memoria detto *garbage collector*, liberando così il programmatore dall'onere della gestione della memoria. Il secondo problema, invece, fu risolto facendo in modo che i programmi non fossero compilati in codice macchina ma in una sorta di codice intermedio che dovrà poi essere eseguito non dall'hardware direttamente ma dalla macchina virtuale detta JVM, ovvero *Java Virtual Machine*.

### 3.3.2 Concetti web

Riguardo a questo argomento sono stati ripassati in dettaglio due argomenti: il concetto di Servlet e il concetto di REST.

#### Servlet

Una Servlet è un oggetto scritto in linguaggio Java in grado di gestire le richieste generate da uno o più client attraverso scambi di messaggi tra il server ed i client stessi. La servlet può utilizzare le Java API per implementare le diverse funzionalità e le specifiche servlet API per mettere a disposizione un'interfaccia standard per gestire le comunicazioni tra il web client e la servlet. E' importante sapere che le servlet non hanno delle GUI.

I vantaggi della Servlet sono:

- \* *efficienza*: la Servlet viene istanziata e caricata una sola volta, alla prima invocazione, mentre le chiamate successive sono gestite chiamando nuovi thread;
- \* *portabilità*: le servlet possono essere facilmente programmate e "portate" in diverse piattaforme;
- \* *persistenza*: dopo essere stata caricata in memoria la servlet rimane anche nelle successive richieste;
- \* *gestione delle sessioni*: grazie alle servlet si riesce a superare la limitazione dei protocolli HTTP senza stati.

#### REST

REST è un insieme di principi architetturali per la progettazione che rendono il Web adatto a realizzare Web Service. I principi sono:

- \* *identificazione delle risorse*: per risorsa si intende qualsiasi elemento in oggetto di elaborazione, ovvero qualsiasi oggetto su cui è possibile effettuare operazioni. Ciascuna risorsa deve essere identificata univocamente, il meccanismo più naturale è il concetto di URI;

- \* *uso esplicito dei metodi HTTP*: serve un meccanismo per indicare le operazioni che si possono fare sulle risorse, per questo motivo si sfruttano i metodi predefiniti del protocollo HTTP;
- \* *risorse autodescrittive*: è opportuno usare i formati più standard possibili per semplificare l'interazione con il client. Il tipo di rappresentazione è indicato nella risposta HTTP;
- \* *collegamenti tra risorse*: le risorse devono essere messe tra di loro in relazione tramite link ipertestuali, è un principio detto HATEOAS
- \* *comunicazione senza stato*: è un principio secondo il quale una richiesta non ha alcuna relazione con le richieste precedenti e successive.

### 3.3.3 Spring

Il framework Spring è una piattaforma Java che fornisce un'infrastruttura di supporto per sviluppatori in Java in modo tale che loro si occupino dalla parte dell'applicazione. I benefici di Spring sono:

- \* dipendenze esplicite ed evidenti grazie alla Dependency Injection;
- \* i contenitori legati all'Inversion of Control tendono ad essere leggeri;
- \* non reinventa nulla, prende quello che è già esistente e lo rende disponibile;
- \* è organizzato in modo modulare quindi il programmatore si preoccupa del pacchetto che gli interessa;
- \* è un model view controller framework ben formato;
- \* fornisce un'interfaccia di gestione delle transizioni.

Di questo framework in particolare ho studiato: Spring Boot, Spring Data, Spring Data JPA e Spring Data REST. Nelle sezioni successive verranno spiegati questi concetti.

#### Spring Boot

Spring Boot è un progetto Spring che ha lo scopo di rendere più semplice lo sviluppo e l'esecuzione delle applicazioni Spring. Un'applicazione richiede molti metadati per la configurazione e può risultare pesante, Spring Boot invece alleggerisce questo meccanismo fornendo una configurazione automatica. Infatti esso è basato su opinioni e convenzioni proprie per avere una configurazione minima con la possibilità di riscriverle se necessario. Alcune caratteristiche principali sono legate ai seguenti aspetti:

- \* *starter dependencies*, ovvero configurazione automatica delle librerie e delle dipendenze;
- \* configurazione automatica di bean e componenti e delle loro relazioni;
- \* *actuator*, per ispezionare un'applicazione in esecuzione.

Spring Boot quindi semplifica la gestione delle dipendenze fornendo e supportando una serie di dipendenze dette *starter*, ovvero dipendenze la cui inclusione implica l'inclusione automatica delle loro dipendenze transitive. La classe detta *Spring Boot*

*Application* esegue l'avvio dell'applicazione etichettando la classe di configurazione, abilitando la scansione e l'identificazione automatica dei componenti, ed infine creando i componenti mancanti necessari alla configurazione dell'applicazione.

Per riassumere i vantaggi offerti da Spring Boot sono:

- \* possibilità di incorporare applicazioni web server per cui non è necessario l'uso di file WAR, acronimo di *Web Application Archive*;
- \* configurazione Maven semplificata;
- \* configurazione automatica se possibile;
- \* fornitura di caratteristiche non funzionali come metriche o configurazioni esternalizzate.

### Spring Data

Spring Data è un progetto ad alto livello di Spring con lo scopo di unificare e facilitare l'accesso a diversi livelli di archiviazione, ovvero database sia relazionali che non relazionali. Spring Data fornisce interfacce generiche per gli aspetti legati a *Crud Repository* e *Paging And Sorting Repository* per ottenere implementazioni specifiche dall'archivio di persistenza. Grazie alle repository basterà scrivere l'interfaccia con i metodi di ricerca definiti in base ad un determinato insieme di convenzioni che possono variare in base al tipo di archiviazione in uso. Sarà Spring a fornire un'implementazione appropriata di tale interfaccia in fase di esecuzione.

### Spring Data JPA

Spring Data JPA aiuta ad implementare facilmente sistemi basati su archiviazione di tipo JPA, framework per il linguaggio di programmazione Java che gestisce la persistenza dei dati in un database relazionale che usano le Java Platform, Standard Edition e Java Enterprise Edition. Questo modulo infatti si occupa del supporto per i livelli di accesso ai dati basati su JPA e rende più semplice la creazione di applicazioni basate su Spring che utilizzano tecnologie di accesso ai dati.

Per molto tempo implementare un livello di accesso ai dati è risultato complicato: era necessario scrivere molto codice per eseguire query semplici, impaginazione e controllo dei dati. Spring Data JPA mira a ridurre lo sforzo all'importo effettivamente necessario. Lo sviluppatore scriverà le interfacce del repository, inclusi i metodi personalizzati, e sarà Spring a fornire l'implementazione automaticamente.

### Spring Data REST

Spring Data REST semplifica la creazione di servizi Web REST basati su repository di Spring Data analizzando il modello di dominio dell'applicazione ed espone le risorse HTTP basate su *hypermedia* per gli aggregati contenuti nel modello.

Spring Data REST definisce automaticamente gli *endpoint* di aggiunta, visualizzazione, rimozione e modifica. Così facendo viene eliminato molto del lavoro manuale solitamente associato a queste attività e rende semplice l'implementazione delle funzionalità [CRUD<sub>G</sub>](#).



### 3.3.4 Blockchain

La Blockchain è una tecnologia che sfrutta le caratteristiche di una rete informatica di nodi e consente di gestire ed aggiornare, in modo univoco e sicuro, un registro contenente dati ed informazioni in maniera aperta, condivisa e distribuita senza la necessità di un'entità centrale di controllo e verifica. Le applicazioni della blockchain sono contraddistinte dalla necessità di decentralizzazione e disintermediazione in modo da evitare banche, istituzioni finanziarie e così via.

Questa tecnologia abilita l'Internet of Value e si basa su un registro distribuito, sistema in cui i nodi di una rete possiedono la medesima copia di un database che modificano e leggono i dati indipendentemente dagli altri nodi. Il registro di queste tecnologie è strutturato come una catena di blocchi contenenti le transazioni ed il consenso è distribuito su tutti i nodi della rete, in questo modo i nodi possono partecipare al processo di validazione delle transazioni da includere nel registro.

### 3.3.5 Ethereum

Ethereum è una piattaforma digitale che permette di costruire una gamma di applicazioni decentralizzate. Queste applicazioni possono includere programmi di sicurezza, sistemi elettorali e metodi di pagamento. Questa piattaforma opera fuori dal mandato delle autorità centrali ed è per questo motivo che opera nel mercato delle criptovalute. Ethereum funziona come piattaforma software basata sulla tecnologia blockchain, simile a quella dei bitcoin perchè conserva le transazioni, ed inoltre permette ai programmatori di costruire applicazioni decentralizzate. Queste applicazioni sono software open source che utilizzano la blockchain e gli smart contract e non hanno bisogno di mediatori.

### 3.3.6 NFT

**NFT**, acronimo per Non Fungible Token, si tratta di token insostituibili che rappresentano la proprietà di un bene, come un'opera d'arte digitale o reale. Un token è un oggetto con un valore particolare e simbolico, in particolare un token sulla blockchain è un'informazione digitale registrata su un registro distribuito. Questa informazione è associata ad un utente specifico e rappresenta un certo tipo di diritto, come proprietà di un oggetto o la ricezione di un pagamento. Gli **NFT** sono diversi dagli altri token perchè sono insostituibili, unici, indivisibili ed è questo il motivo per cui si prestano alla cessione dei diritti di proprietà di opere d'arte.

Quando una persona compra un **NFT** si deve capire che:

- \* non ha comprato l'opera: fisicamente essa rimane sempre in possesso dell'autore;
- \* non ha comprato i diritti d'autore: non ha diritto a riprodurla, basare altre opere su di essa oppure utilizzarla come se l'avesse creata;
- \* non ha comprato l'esclusività della riproduzione o dell'uso.



## Capitolo 4

# Analisi dei requisiti

*In questo capitolo verranno elencati i casi d'uso delle funzionalità implementate con i relativi requisiti ed i loro tracciamento*

### 4.1 Casi d'uso

Per lo studio dei casi d'uso del prodotto creato sono stati creati dei diagrammi. Questi diagrammi, detti appunto dei casi d'uso (in inglese *Use Case Diagram*), sono diagrammi di tipo [Unified Modeling Language \(UML\)](#) dedicati alla descrizione del sistema e delle funzioni o servizi offerti da un esso, e come gli utilizzatori interagiscono con esso. Lo strumento utilizzato per la realizzazione di tali diagrammi è StarUML.

#### 4.1.1 Attori dei casi d'uso

I possibili attori dei casi d'uso analizzati sono i seguenti attori primari:

- \* utente non autenticato
- \* utente autenticato



**Figura 4.1:** Attori dei casi d'uso

## UC1: Registrazione

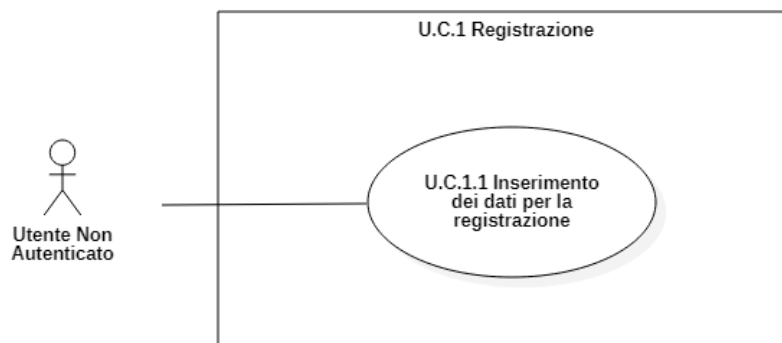
**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è ancora presente nei registri del sistema.

**Descrizione:** L'utente accede all'applicazione e naviga fino alla pagina di registrazione. L'utente inserisce i dati necessari e li conferma e questo porta l'utente a possedere un account nel sistema.

**Postcondizioni:** L'utente risulta presente nei registri del sistema e gli è possibile autenticarsi nella piattaforma.

### UC1.1: Inserimento dei dati per la registrazione



**Figura 4.2:** U.C.1.1 Inserimento dei dati per la registrazione

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non autenticato si trova nella pagina di navigazione.

**Descrizione:** L'utente non autenticato compila i campi nel seguente modo:

- \* Inserimento del nome (U.C1.1.1)
- \* Inserimento del cognome (U.C1.1.2)
- \* Inserimento dell'email (U.C1.1.3)
- \* Inserimento della password (U.C1.1.4)
- \* Inserimento della conferma della password (U.C1.1.5)
- \* Inserimento dell'anno di nascita (U.C1.1.6)
- \* Inserimento dell'ID del Wallet (U.C.1.1.7)

DA SISTEMARE PER IL PUNTO SCOMO.

**Postcondizioni:** L'utente ha completato la compilazione dei campi e può procedere con la registrazione.

**UC1.1.1: Inserimento del nome**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce il proprio nome.

**Postcondizioni:** L'utente ha inserito il nome.

**UC1.1.2: Inserimento del cognome**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce il proprio cognome.

**Postcondizioni:** L'utente ha inserito il cognome.

**UC1.1.3: Inserimento dell'email**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce la sua email.

**Postcondizioni:** L'utente ha inserito l'email.

**UC1.1.4: Inserimento della password**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce la sua password.

**Postcondizioni:** L'utente ha inserito la password.

**UC1.1.5: Inserimento della conferma della password**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce la conferma della password.

**Postcondizioni:** L'utente ha inserito la conferma della password.

**UC1.1.6: Inserimento dell'anno di nascita**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce il proprio anno di nascita.

**Postcondizioni:** L'utente ha inserito l'anno di nascita.

**UC1.1.7: Inserimento dell'ID Wallet**

**Attori Principali:** Utente non autenticato.

**Precondizioni:** L'utente non è autenticato e non ha compilato questo campo.

**Descrizione:** L'utente non autenticato inserisce l'ID Wallet.

**Postcondizioni:** L'utente ha inserito l'ID Wallet.

## UC2: Login

**Attori Principali:** Utente non autenticato.

**Precondizioni:** .

**Descrizione:** .

**Postcondizioni:** .

## UC2.1: Inserimento dei dati per il login

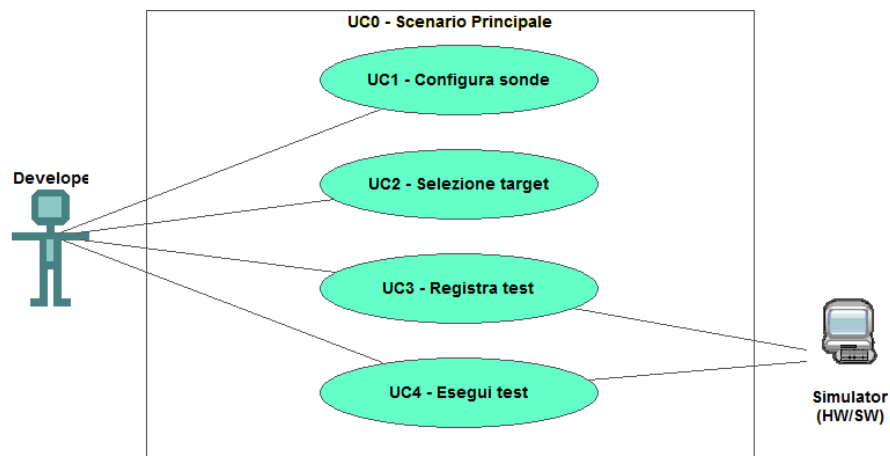
**Attori Principali:** Utente non autenticato.

**Precondizioni:** .

**Descrizione:** .

**Postcondizioni:** .

## UC0: Scenario principale



**Figura 4.3:** Use Case - UC0: Scenario principale

**Attori Principali:** Sviluppatore applicativi.

**Precondizioni:** Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

**Descrizione:** La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

**Postcondizioni:** Il sistema è pronto per permettere una nuova interazione.

**Tabella 4.1:** Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

**Tabella 4.2:** Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

## 4.2 Tracciamento dei requisiti

Come risultato di un'attenta analisi dei requisiti e i relativi casi d'uso effettuata sul progetto sono stati individuati diversi requisiti. Questi sono stati suddivisi per classificazione e tipologia, per questo motivo si utilizza un codice identificativo per distinguerli che è così strutturato:

$$\mathbf{R[classificazione][tipologia][codice]}$$

La descrizione del codice è la seguente:

- \* **R**: acronimo per Requisito;
- \* **classificazione**: individua la classificazione del requisito e può essere:
  - F = funzionale
  - Q = qualitativo
  - V = di vincolo
- \* **tipologia**: individua la tipologia del requisito e può essere:
  - O = obbligatorio
  - D = desiderabile
  - F = facoltativo

Nelle sezioni seguenti sono riassunti i requisiti ed il loro tracciamento con i casi d'uso delineati in fase di analisi.

**Tabella 4.3:** Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-





## Capitolo 5

# Progettazione e codifica

*In questo capitolo verranno spiegate le tecnologie utilizzate per lo sviluppo e l'architettura del prodotto software*

### 5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati per lo sviluppo delle maschere e per la collaborazione con i colleghi stagisti ed il tutor aziendale.

#### 5.1.1 Tecnologie

Vue.js



**Figura 5.1:** Logo di Vue.js

Vue.js è un framework javascript open source nato nel 2013 che presenta un'architettura adottabile in modo incrementale che si concentra sulla composizione dei componenti, inoltre sono presenti funzionalità avanzate offerte tramite librerie e pacchetti di supporto. I componenti Vue estendono gli elementi HTML di base per incapsulare del codice riutilizzabile quindi a livello generale i componenti sono elementi personalizzati a cui il compilatore Vue associa una particolare funzionalità.

Vue utilizza quindi una sintassi basata su HTML e consente di associare il DOM

renderizzato ai dati dell'istanza di Vue sottostante. In questo modo i modelli Vue possono essere analizzati da browser e parser HTML conformi alle modifiche ed inoltre con il sistema di reattività Vue è in grado di calcolare il numero minimo di componenti per eseguire nuovamente il rendering applicando la quantità minima di manipolazioni DOM quando cambia lo stato dell'app. Vue presenta un sistema reattivo grazie all'utilizzo di oggetti semplici Javascript ed ad un re-rendering ottimizzato: ogni componente durante il render tiene traccia delle sue dipendenze in modo tale che il sistema sappia quando e di quali componenti deve effettuare nuovamente il render.

Un problema che affligge le web application a pagina singola è che quest'ultime forniscono agli utenti la risposta basata solamente sull'URL dal server, di conseguenza l'utilizzo dei segnalibri a determinate schermate e la condivisione dei collegamenti a sezioni specifiche risulta molto difficile se non impossibile. Vue.js per riuscire a risolvere questo problema fornisce un'interfaccia, detta router, che dà la possibilità di modificare ciò che viene visualizzato sulla pagina in base all'URL indipendentemente da come esso viene modificato. Infatti Vue.js viene fornito con il pacchetto open source "vue-router" che fornisce un'API per aggiornare l'URL dell'applicazione, supportare la cronologia di navigazione e le reimpostazioni di email e password. Tramite questa tipologia di router i componenti devono essere mappati alla route a cui appartengono per indicare dove deve essere eseguito il loro render.

Questo framework implementa il pattern MVVM, acronimo per Model-View-View-Model, una declinazione del più famoso MVC, ovvero Model-View-Controller. I componenti del MVVM sono:

- \* Model (o Modello): l'implementazione del dominio dati come per il classico Modello del pattern MVC;
- \* View (o Vista): il componente grafico renderizzato dall'utente formato da HTML e CSS;
- \* ViewModel (o Vista per il Modello): il collante tra gli altri due componenti, esso fornisce alla View i dati in formato consono alla rappresentazione ed il comportamento di alcuni elementi dinamici.






La grossa differenza tra il pattern implementato da Vue.js e il Model-View-Controller sta nella differenza tra Controller e ViewModel. Il primo, infatti, è una porzione di codice che gestisce la logica di business grazie al Model e ritorna una View da mostrare all'utente; il secondo, invece, rappresenta una versione parallela al Model che risulta essere legato alla View e descrive il comportamento di quest'ultima con funzioni associate. Quindi mentre il Controller esegue logiche di business prima del rendering della View, il ViewModel definisce il comportamento dell'applicazione a runtime.

### Vuetify

Vuetify è un framework UI completo costruito su Vue.js nel 2014 ed il suo obiettivo è fornire agli sviluppatori gli strumenti per poter creare esperienze utente ricche e coinvolgenti. A differenza di altri framework Vuetify è progettato da zero in modo tale da renderlo facile da imparare ed essere gratificante da padroneggiare con centinaia di componenti realizzate dalle specifiche di Material Design.

Un pregio di questo framework è che adotta un approccio al mobile, questo significa che la web application sviluppata tramite esso sarà pienamente utilizzabile immediatamente su un tablet, un telefono ed un computer. Inoltre è un framework in sviluppo attivo che viene aggiornato settimanalmente rispondendo ai problemi e relativi report della

community. Un altro pregio è, come si può vedere nell'immagine sottostante, il gran numero di funzionalità che possiede Vuetify in confronto agli altri framework di Vue.

Vue Framework Comparison 2021					
Features	 Vuetify	 BootstrapVue	 Buefy	 Element UI	 Quasar
Accessibility and section 508 support	●	●	●		
Business and enterprise support	●				
Long-term Support	●				
Release cadence**	Weekly	Bi-Weekly	Bi-Monthly	Bi-Weekly	Bi-Weekly
RTL support	●	●		●	●
Premium themes	●	●			
Treeshaking	Automatic	Manual	Manual	Manual	Automatic
**Based on average of all Major/Minor/Patch releases over the last 12 months.					

**Figura 5.2:** Funzionalità di Vuetify

## 5.1.2 Strumenti

### Github

Github è un servizio web e cloud-based fondato nel 2008 che aiuta gli sviluppatori ad archiviare, gestire il codice, tracciare e controllare le modifiche. I due argomenti principali legati a Github sono: controllo versioni e Git.

Il controllo versioni aiuta a tracciare e gestire le modifiche del codice di un progetto software: più un progetto risulta essere di grandi dimensioni più il controllo delle versioni diventa fondamentale. Infatti questo aiuta gli sviluppatori a lavorare con sicurezza attraverso due azioni:

- \* branching: uno sviluppatore duplica parte del codice sorgente, detto repository, in modo da apportare modifiche in modo sicuro senza influenzare l'intero progetto;
- \* merging: una volta che lo sviluppatore è certo di aver prodotto del codice funzionante può fondere quel codice nel quello sorgente e renderlo ufficiale.

In questo modo tutte le modifiche possono essere monitorate e, se necessario, ripristinate. Inoltre grazie a questi concetti è possibile lavorare sullo stesso progetto in diversi sviluppatori senza problemi di ripetizioni di codice o conflitti durante lo sviluppo.

Git è un sistema di controllo versioni distribuito realizzato nel 2005, ovvero l'intero codice base e la cronologia sono disponibili sul computer di ogni sviluppatore. In questo modo è possibile creare facilmente ramificazioni e fusioni.

Per organizzare il lavoro abbiamo creato un'*organization*, prodotto offerto da Github che rende più semplice la collaborazione su più progetti contemporaneamente. Infatti ogni stagista ha creato la propria repository all'interno di questo spazio comune in

modo tale da lavorare con i colleghi contemporaneamente e dare la possibilità ai tutor aziendali di visionare il lavoro svolto.

### **Figma**

Figma è uno strumento nato nel 2016 rivolto ai web designer che hanno bisogno di un tool per la progettazione di interfacce. I vantaggi offerti da questo strumento sono i seguenti:

- \* accessibilità multiplatforma
- \* sistema di collaborazione in real-time
- \* utilizzo degli strumenti responsive oriented per una progettazione ottimale
- \* lavora in vettoriale

Grazie a questo strumento io e gli altri stagisti legati all'ambito front end abbiamo potuto creare dei prototipi delle maschere che avremmo dovuto sviluppare.

### **Stoplight**

Stoplight è una piattaforma di progettazione [Application Program Interface \(API\)](#) collaborativa che si integra perfettamente nei flussi di lavoro per consentire a chiunque lavori con le [API](#) di essere più produttive. Questa piattaforma si basa su tre principi guida fondamentali, in modo da essere responsabili, collaborativi e con intenti positivi. I principi sono i seguenti:

- \* quando si vede un'opportunità per avere un impatto, bisogna coglierla;
- \* bisogna confidare l'uno nell'altro per aggiungere valore e risolvere problemi attraverso il lavoro di squadra;
- \* bisogna cercare di capire gli altri ascoltando, indagando e rispondendo con attenzione.

Grazie a questa piattaforma si può aiutare gli utenti interni ed esterni ad integrarsi rapidamente all'[API](#) di un altro utente pubblicando documentazione interattiva, tutorial ed esempi di codice sempre aggiornati.

### **Visual Studio Code**

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft nel 2015. Esso può essere utilizzato con diversi linguaggi di programmazione, infatti incorpora in esso diverse funzioni che variano dal linguaggio con cui si sta programmando. Un punto di forza di questo editor è la sua grande malleabilità, infatti l'utente può installare o disinstallare diversi plugin in funzione del linguaggio che sta utilizzando.

Un altro punto di forza è la sua integrazione con Git facilitando il controllo di versione del lavoro svolto dal programmatore che ha la possibilità di controllare le modifiche fatte, risolvere eventuali conflitti e salvare il lavoro nella repository. Altri punti di forza di questo editor sono i seguenti:

- \* IntelliSense, estensione che fornisce suggerimenti di completamento per variabili, metodi e moduli. In particolare per i metodi fornisce una breve spiegazione dei parametri che accettano e del loro funzionamento;

- \* Debug semplificato grazie ai plugin messi a disposizione e all'interfaccia grafica intuitiva;
- \* Modifica veloce delle righe di codice grazie ad avvisi su codice sospetto, possibilità di selezionare più righe contemporaneamente e suggerimenti sui parametri;
- \* Navigazione e refactoring del codice semplice.

## 5.2 Ciclo di vita del software

## 5.3 Progettazione

### Namespace 1

Descrizione namespace 1.

**Classe 1:** Descrizione classe 1

**Classe 2:** Descrizione classe 2

## 5.4 Design Pattern utilizzati

## 5.5 Codifica



## Capitolo 6

# Verifica e validazione





## Capitolo 7

# Conclusioni

7.1 Consuntivo finale

7.2 Raggiungimento degli obiettivi

7.3 Conoscenze acquisite

7.4 Valutazione personale



Appendice A

Appendice A

Citazione

---

Autore della citazione



# Glossario

**API** in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. [35](#)

**CRUD** in informatica CRUD, acronimo di create read update delete, sono le quattro operazioni di un database persistente. Nel contesto dello sviluppo di applicazioni web corrispondono alle quattro operazioni HTTP: put, get, post e delete.. [12](#), [35](#)

**framework** architettura logica di supporto sulla quale un software può essere progettato e sviluppato, facilitandone lo sviluppo da parte del programmatore. [1](#)

**front end** parte dell'applicazione con cui l'utente può interagire. [1](#)

**NFT** è un tipo speciale di token crittografico che rappresenta qualcosa di unico, i token non fungibili non sono interscambiabili. Questo concetto va in contrasto con le criptovalute che sono di natura fungibili.. [1](#), [4](#), [35](#)

**Scrum** è un framework agile utilizzato per la gestione del ciclo di vita del software, iterativo ed incrementale, con lo scopo di gestire progetti, prodotti software o applicazioni di sviluppo. [5](#)

**stakeholder** In ingegneria del software uno stakeholder è una persona a vario titolo coinvolta nel ciclo di vita di un software, che ha influenza sul prodotto o sul processo. [5](#)

**UML** in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. [35](#)



# Acronimi

**API** [Application Program Interface](#). [24](#), [33](#)

**CRUD** [Create Read Update Delete](#). [33](#)

**NFT** [Non-Fungible Token](#). [5](#), [13](#), [33](#)

**UML** [Unified Modeling Language](#). [15](#), [33](#)





# Bibliografia