POLITECNICO DI BARI

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE
CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Chain4Good

Progettazione e sviluppo di una piattaforma di crowdfunding con tecnologia blockchain

Candidati:
Angelica DE FEUDIS
Jonathan CAPUTO
Luca GENTILE

Docente:
Prof.ssa Marina MONGIELLO

Academic Year: 2025/2026

Chain4Good

Indice

Elenco delle figure

Elenco delle tabelle

Acronimi

CF Crowdfunding.

DCF Donation-based Crowdfunding.

ECF Equity-based Crowdfunding.

IDE Ambiente di Sviluppo Integrato.

LCF Lending-based Crowdfunding.

ODM Object Data Modeling.

P2P Peer-to-Peer.

RBC Reward-based Crowdfunding.

SPOF Single Point of Failure.

VCS Version Control System.

1 Introduzione

Il Crowdfunding (CF) rappresenta uno strumento di finanziamento collettivo, capace di aggregare un elevato numero di contributi economici di modesta entità a supporto di progetti e iniziative di diversa natura. In particolare, nel contesto del Terzo Settore, il CF rappresenta un mezzo efficace per sostenere attività a fini sociali, culturali e umanitari, consentendo agli enti beneficiari di raggiungere una platea ampia e diversificata di donatori attraverso piattaforme digitali.

Nonostante la sua crescente diffusione, i sistemi tradizionali presentano criticità strutturali riconducibili principalmente alla loro architettura centralizzata. In particolare, la gestione fiduciaria dei fondi raccolti e la limitata trasparenza del loro utilizzo rappresentano fattori capaci di incidere negativamente sul livello di fiducia degli utenti, tale da compromettere la loro partecipazione ai processi di donazione.

In questo scenario, la blockchain si configura come una soluzione promettente per il superamento di questi limiti. Le sue proprietà di decentralizzazione, immutabilità e trasparenza permettono infatti di ridurre la dipendenza da intermediari fiduciari, ridefinendo il paradigma delle piattaforme di CF tradizionali.

Alla luce di queste considerazioni, il presente lavoro propone *Chain4Good*, una piattaforma di CF decentralizzata, nata specificamente per supportare iniziative promosse dagli Enti del Terzo settore. Il suo obiettivo principale è restituire al donatore un ruolo attivo lungo l'intero ciclo di vita delle donazioni. Per farlo, implementa un meccanismo di erogazione incrementale dei fondi, basato su processo di approvazione decentralizzato, attraverso il quale i donatori sono chiamati a votare sulle singole richieste di spesa presentate dai beneficiari. In questo senso, la piattaforma proposta ha il potenziale di ridefinire il concetto di donazione trasformandolo da un mero gesto di fiducia in un processo intrinsecamente sicuro, trasparente e verificabile in ogni sua fase.

2 Stato dell'arte

2.1 Crowdfunding

Il *CF* è un modello di finanziamento collettivo in cui una pluralità di individui decide di destinare il proprio denaro, prevalentemente tramite piattaforme digitali, a supporto di progetti e iniziative di varia natura [1]. In ragione della sua etimologia, dall'inglese *crowd* "folla" e *funding*, finanziamento, il CF è stato definito come una pratica di microfinanziamento "dal basso" [2], la cui peculiarità risiede nella capacità di aggregare numerosi contributi finanziari di modesta entità a partire da un'ampia platea di sostenitori.

La letteratura attribuisce al Web 2.0 il principale catalizzatore del successo del CF [3]. Lo sviluppo di Internet e la capillare diffusione di canali digitali di comunicazione, come i *social-media*, infatti, ha permesso non solo di ampliare la platea di donatori, ma anche di abbattere i limiti geografici, trasformando la "folla" in una comunità attiva e globale. Inoltre, la nascita di infrastrutture digitali dedicate, come *GoFundme*, è stato determinate per garantire la scalabilità del fenomeno.

In questo scenario, il modello contemporaneo di CF si articola in un'architettura tripartita, che vede l'interazione sinergica di tre attori chiave: il promotore dell'iniziativa, i sostenitori e la piattaforma digitale [4]. Quest'ultima non funge da mera vetrina, ma rappresenta l'infrastruttura tecnologica che media le interazioni tra le parti, facilitando il processo di raccolta fondi, la diffusione delle informazioni e il coordinamento delle attività connesse alla realizzazione del progetto. Sebbene la struttura relazionale del CF rimanga invariata, la natura del contributo richiesto e le aspettative di ritorno dei sostenitori, rappresentano gli elementi chiave che ne definiscono la tassonomia. E' sulla base di questi criteri, infatti, che gli studi convergono nel classificare le seguenti tipologie di CF:

- Donation-based Crowdfunding (DCF): i contributi economici sono erogati senza alcuna aspettativa di ritorno materiale o finanziario. La donazione è motivata esclusivamente dal desiderio di sostenere una causa di interesse collettivo o di pubblica utilità; per questa ragione, la DCF è stata definita come la forma più "pura" di *crowdfunding* [5];
- Reward-based Crowdfunding (RBC): i sostenitori finanziano un progetto in cambio di una ricompensa, generalmente di natura non finanziaria (come riconoscimenti simbolici oppure ricompense tangi-

bili, configurandosi spesso come un vero e proprio "pre-ordine" del prodotto) [6];

- Equity-based Crowdfunding (ECF): il finanziatore, sia esso un individuo o un ente, riceve quote societarie o titoli partecipativi dell'azienda, in cambio del capitale investito [7]
- Lending-based Crowdfunding (LCF): noto anche come *debt-based crowdfunding*, prevede che il capitale versato dai sostenitori venga rimborsato dal promotore entro una scadenza prestabilita, comprensivo di un tasso di interesse pattuito [8];

Nonostante la natura prettamente finanziaria degli ultimi due modelli, l'elemento che li riconduce univocamente al paradigma del CF è la modalità di raccolta: il capitale non è più appannaggio di un singolo grande istituto di credito, ma deriva dalla somma di innumerevoli micro-investimenti operati da una moltitudine di individui. E' proprio questo a sancire la natura "dal basso" di tali strumenti, trasformando ogni cittadino in un potenziale nodo di una rete di finanziamento globale.

2.1.1 Limiti delle piattaforme tradizionali di CF

La letteratura converge nel considerare le piattaforme di CF caratterizzate da una serie di criticità strutturali, riconducibili principalmente al loro modello architetturale centralizzato. Si distinguono:

- limitata trasparenza nell'utilizzo dei fondi: i donatori non dispongono di strumenti efficaci per verificare l'intero ciclo di vita delle donazioni [4]. La tracciabilità delle quote donate è, infatti, demandata al fruitore della donazione, il quale ha il compito di fornire aggiornamenti sullo stato di avanzamento dell'iniziativa finanziata;
- scarsa fiducia e rischio di frodi: l'assenza di protocolli di verifica automatizzati rende le piattaforme di CF tradizionali vulnerabili a comportamenti fraudolenti, come la creazione di campagne ingannevoli o la mancata realizzazione dei progetti finanziati [9]. Questo clima di incertezza scoraggia la partecipazione degli utenti, alimentando una diffusa sfiducia nei confronti delle piattaforme.
- centralizzazione: l'architettura centralizzata utilizzata dalla maggior parte delle piattaforme introduce Single Point of Failure (SPOF) e attribuisce la gestione dei fondi raccolti interamente alla piattaforma [10];

- mancanza di sicurezza: la gestione centralizzata dei dati espone le piattaforme ad attacchi malevoli, con conseguenze rilevanti in termini di perdita di fondi e fiducia degli utenti [10].
- alti costi operativi: le piattaforme tradizionali applicano commissioni elevate per la gestione delle campagne, riducendo il totale dei fondi raccolti per progetto ([4]);

Nel complesso, queste criticità evidenziano come le piattaforme tradizionali di CF si fondino su un modello fortemente fiduciario, nel quale il corretto funzionamento del sistema dipende dal comportamento onesto degli intermediari e dei promotori delle iniziative. Sebbene questo paradigma abbia dunque democratizzato l'accesso al capitale, l'architettura adottata introduce inefficienze strutturali che limitano il potenziale del modello di CF.

2.2 Crowdfunding e blockchain

La letteratura converge nel considerare la *blockchain* come una soluzione promettente per il superamento delle criticità strutturali delle piattaforme di CF esistenti [11]. L'efficacia di tale tecnologia risiede nelle proprietà native di decentralizzazione, trasparenza, immutabilità e sicurezza (Tabella 1) che implementa.
La *blockchain*, infatti, si configura come un registro distribuito, condiviso e immutabile, mantenuto da una rete di nodi interconnessi secondo un'architettura Peer-to-Peer (P2P). Essa consiste in una catena sequenziale di blocchi legati tra loro da meccanismi crittografici. Ciascun blocco contiene un insieme di transazioni (operazioni atomiche come il trasferimento di *asset* digitali), la cui integrità è preservata da protocolli di consenso eseguiti in modo distribuito dalla rete [12].
L'integrazione di tale architettura nel CF è motivata dai seguenti vantaggi:

- la natura distribuita del registro (proprietà di decentralizzazione) permette di eliminare i SPOF, tipici delle architetture centralizzate, e di attribuire ad un insieme di nodi piuttosto che al singolo ente, la gestione dei fondi raccolti;
- la proprietà di immutabilità assicura l'integrità delle informazioni memorizzate *on-chain* [5], come le transazioni di donazione o i dettagli sulle campagne di raccolta fondi;

- la trasparenza intrinseca del *ledger*, permette ai donatori di monitorare l'intero ciclo di vita delle donazioni, rafforzando in questo modo la fiducia nelle piattaforme.

2.2.1 Soluzioni esistenti

Le potenzialità della tecnologia *blockchain* analizzate nel paragrafo precedente hanno dato origine a diverse implementazioni sperimentali in letteratura, volte a prototipare sistemi di CF decentralizzati. Tali soluzioni mirano principalmente a superare i limiti dei modelli tradizionali attraverso l'eliminazione degli intermediari fiduciari e l'introduzione di meccanismi di controllo trasparenti e verificabili.
Le principali applicazioni si basano sull'utilizzo degli *Smart Contract*, ossia programmi immutabili, registrati direttamente sulla *blockchain* ed eseguiti automaticamente al verificarsi di condizioni prestabilite [1]. Questi contratti vengono principalmente utilizzati per gestire il ciclo di vita delle donazioni e per implementare meccanismi di approvazione delle richieste di spesa fondati sul consenso dei donatori.
Nel modello proposto da Patil e colleghi [1], ad esempio, gli *Smart Contract* hanno la funzione di impedire l'erogazione immediata e integrale dei fondi donati al beneficiario. Il capitale rimane vincolato nel contratto e viene rilasciato in modo incrementale solo al raggiungimento di obiettivi intermedi (*milestone*) stabiliti in fase di creazione del progetto. Tuttavia, lo sblocco delle risorse e l'approvazione dell'inserimento dei progetti in

Proprietà	**Descrizione**
Decentralizzazione	La replicazione del registro su più nodi consente al sistema di operare correttamente anche in presenza di guasti, eliminando i SPOF [13].
Immutabilità	Ogni transazione è irreversibile; una volta registrata nella blockchain, non può essere cancellata o modificata, poiché qualsiasi alterazione comprometterebbe l'intera catena dei blocchi [14].
Trasparenza	Tutte le transazioni sono verificabili pubblicamente (nelle *blockchain permissionless*) oppure dai membri autorizzati (nelle *blockchain permissioned*) [15].
Sicurezza	L'uso combinato della crittografia asimmetrica per l'autenticazione e dei protocolli di consenso distribuito per l'integrità del registro, rende il sistema resistente ad attacchi e frodi.

Tabella 1: Caratteristiche della tecnologia *blockchain*

piattaforma sono affidati a un utente con ruolo di *Admin*, il che riduce il grado di decentralizzazione del sistema.
Diversamente, l'architettura proposta da Yadav e Sarasvathi [16] elimina ogni figura di supervisione esterna in favore di una struttura puramente decentralizzata. In questo studio, infatti, il rilascio dei fondi è sempre incrementale ma è delegato a un sistema di voto integrato nello *Smart Contract*. In particolare, affinché il beneficiario possa utilizzare una quota parte dei fondi raccolti deve presentare una richiesta di spesa, specificando la causale dell'esborso, l'importo richiesto e l'indirizzo pubblico del fornitore destinatario del pagamento.
Solo se la proposta viene approvata da almeno il 50% dei sostenitori del progetto, lo *Smart Contract* esegue la transazione direttamente verso il fornitore. Il valore aggiunto di questo approccio si esplica dunque nella capacità di aumentare la partecipazione dei donatori lungo l'intero ciclo di vita della donazione. Tuttavia, il modello non prevede meccanismi che certifichino l'autenticità del fornitore poiché lo *Smart Contract* si limita a eseguire transazioni verso l'indirizzo fornito senza poterne verificare l'identità reale.
Per mitigare tale criticità, il modello proposto in [4] attribuisce ai donatori il compito di verificare l'indirizzo del portafoglio del destinatario attraverso database esterni prima di esprimere il proprio voto. Inoltre, come ulteriore forma di tutela, viene introdotto un meccanismo di segnalazione delle richieste ritenute sospette, che consente ai donatori di indicare potenziali comportamenti fraudolenti e di bloccare l'erogazione dei fondi qualora venga raggiunta una soglia di consenso prestabilita.
È evidente dunque che sebbene l'adozione di un modello completamente decentralizzato abbia il potenziale di incrementare il numero dei donatori grazie all'aumento dei livelli di trasparenza e verificabilità dell'intero ciclo di donazione, al contempo questo comporta un aumento del livello di complessità sia in fase di sviluppo che di utilizzo della piattaforma.
In questo senso, *Chain4Good* si inserisce come un prototipo applicativo che mutua alcuni dei meccanismi consolidati in letteratura e li adatta a uno scenario operativo specifico. Inoltre, piuttosto che puntare a una decentralizzazione completa, il sistema mantiene la figura di un *Admin* per la validazione e la registrazione degli Enti del Terzo Settore, e introduce al contempo vincoli ulteriori, volti a garantire un utilizzo delle risorse realmente progressivo, trasparente e verificabile dai donatori.

3 Metodologia di progetto

3.1 Modello di processo

Per lo sviluppo di questo sistema è stato adottato un modello di processo *Agile* di tipo *Incrementale*. Questa scelta è motivata dalla necessità di coniugare la flessibilità dei metodi agili, con la capacità del modello incrementale di gestire le fasi di sviluppo in maniera concorrente e sovrapposta.
Il coordinamento del *team*, invece, ha seguito la tecnica *Scrum*. In particolare, le riunioni periodiche hanno permesso una gestione dinamica del *product backlog* (elenco delle attività da svolgere) e un monitoraggio costante dello stato di avanzamento del progetto, garantendo un'integrazione continua dei risultati discussi.
L'orientamento Agile si è manifestato sin dalle fasi iniziali. Le sessioni di *brainstorming* effettuate hanno permesso di proporre e analizzare diverse alternative progettuali. La decisione di abbandonare la proposta iniziale in favore di una più rispondente alle indicazioni dei referenti riflette i principi cardine del Manifesto Agile, quali: collaborazione con gli *stakeholder* e risposta al cambiamento.
L'adozione del modello incrementale, invece, ha permesso di ottimizzare i tempi di sviluppo. Il progetto, infatti, non è stato condotto secondo una sequenza rigida di fasi, ma ha previsto lo svolgimento in parallelo di più attività.

3.1.1 Organizzazione del team

Lo sviluppo concorrente ha richiesto la suddivisione delle responsabilità di progetto in macro-aree (*front-end*, *back-end* e documentazione tecnica), favorendo l'avanzamento simultaneo dei diversi incrementi del sistema. Tale ripartizione, tuttavia, non ha comportato una compartimentazione stagna dei compiti. Al contrario, ogni membro del gruppo ha mantenuto una visione olistica del progetto, partecipando attivamente alla risoluzione delle criticità anche al di fuori della propria area di competenza primaria. Tale impostazione ha favorito una dinamica di supporto reciproco e interdisciplinare. Il *team* ha, inoltre, operato seguendo il principio della *Collective Ownership*, estendendo a ciascun membro la responsabilità della qualità globale del prodotto.
Complessivamente, l'approccio adottato ha permesso sia di valorizzare i punti di forza di ogni singolo membro che di trasformare le riunioni in opportunità di apprendimento trasversale e di crescita collettiva.
Il successo della metodologia adottata è risultato fortemente legato ai

fattori umani, quali competenza tecnica, condivisione degli obiettivi e cooperazione proattiva all'interno del *team*.

3.2 Pianificazione delle attività

La pianificazione delle attività di progetto è stata condotta mediante la definizione di un insieme strutturato delle principali attività da svolgere, riportate in Tabella 2.
Successivamente, al fine di gestire la sequenzialità e il parallelismo tra i *task* individuati, è stato elaborato un Diagramma di Gantt.
Tale approccio ha consentito di organizzare il lavoro in modo strutturato e di definire un riferimento temporale complessivo per l'esecuzione del progetto.

ID	**Descrizione delle attività**	**Durata (ore)**
Fase 1 – Ideazione e analisi		
T1	*Brainstorming* e definizione dell'idea progettuale	15
T2	Analisi del problema e del dominio applicativo	10
T3	Analisi dei requisiti e studio della fattibilità	6
T4	Analisi dei rischi e criticità	6
T5	Studio delle tecnologie e della *blockchain*	10
Fase 2 – Progettazione		
T6	Progettazione dell'architettura del sistema	10
T7	*Design* delle interfacce utente	20
Fase 3 – Implementazione del sistema		
T8	Sviluppo del *Back-end*	60
T9	Sviluppo del *Front-end*	40
T10	Redazione della documentazione tecnica	50
Fase 4 – Test e miglioramenti		
T11	Verifica del prodotto e delle funzionalità implementate	7
T12	Correzione dei *bug* e perfezionamenti	6
Fase 5 – Conclusione		
T13	Consegna	10
	Totale ore	**250**

Tabella 2: Pianificazione delle attività di progetto.

3.2.1 Stima dei costi

La stima dei costi di progetto è stata effettuata sulla base della durata stimata delle attività definite in fase di pianificazione, come riportato nella Tabella 2. In particolare, l'impegno richiesto è stato valutato operando una scomposizione delle principali fasi progettuali in *task* elementari, e associando a ciascuno di essi una stima temporale. Operando in questo modo, il costo complessivo stimato risulta pari a **250 ore**. Tale valore rappresenta l'*effort* espresso in termini di ore/persona, ottenuto come somma delle ore dedicate da ciascun membro del gruppo alle attività previste.
La valutazione ha tenuto conto della natura prototipale del progetto, della dimensione del sistema, delle tecnologie adottate e del livello di esperienza dei membri del gruppo di lavoro. Tuttavia, trattandosi di un progetto a scopo didattico, la stima dei costi non è stata tradotta in un valore economico monetario, ma è stata utilizzata come strumento di supporto alla pianificazione e alla verifica della fattibilità del progetto entro i vincoli temporali prestabiliti. È infine opportuno precisare che la valutazione effettuata è riferita esclusivamente alla realizzazione del prototipo funzionale: un'eventuale evoluzione verso una piattaforma completa e pronta per l'utilizzo in un contesto reale richiederebbe una nuova analisi dei costi e un impegno temporale significativamente superiore.

3.3 Analisi dei rischi

L'analisi dei rischi è stata condotta al fine di individuare le principali criticità che avrebbero potuto incidere sullo sviluppo e sul corretto funzionamento della piattaforma.
In primo luogo, sono stati presi in considerazione i rischi di progetto, quali la possibile sottostima dei tempi di sviluppo e la variabilità dei requisiti individuati in fase di pianificazione. Per mitigare il rischio legato all'organizzazione temporale delle attività, le principali fasi di progetto (Tabella 2) sono state rappresentate mediante un Diagramma di Gantt. Tale strumento ha consentito di monitorare l'avanzamento delle attività, individuare eventuali scostamenti tra la durata pianificata e quella effettiva e supportare una tempestiva riorganizzazione; Al fine di ridurre le criticità connesse all'evoluzione dei requisiti, invece, è stato adottato un approccio di sviluppo incrementale. Questo modello ha permesso di suddividere il sistema in incrementi funzionali successivi, consentendo di adattare le soluzioni progettuali sulla base delle evidenze emerse durante le fasi di implementazione.

4 Progettazione

4.1 Rischi di dominio

In questa sezione sono riportate le potenziali criticità legate al dominio applicativo del CF e le strategie progettuali impiegate al fine di neutralizzarle.

Un primo ambito di analisi riguarda il costo richiesto per la memorizzazione dei dati *on-chain*. Come evidenziato in letteratura, l'esecuzione di ogni transazione comporta il pagamento di una tariffa, nota come *gas fee*. In considerazione di tale criticità, l'architettura proposta è stata progettata con l'obiettivo di ridurre al minimo il consumo di gas per transazione, intervenendo sia sulla complessità computazionale delle istruzioni eseguite dagli *Smart Contract*, sia sulla quantità di dati scritti all'interno del *ledger*.

Un altro rischio analizzato concerne la trasparenza nell'allocazione delle risorse. E' stata difatti valutata l'eventualità che l'Ente promotore non utilizzasse le risorse raccolte per le finalità dichiarate. Per mitigare tale rischio si è deciso di implementare un meccanismo di custodia decentralizzata dei fondi, secondo il quale le risorse non sono trasferite immediatamente all'Ente al termine della raccolta. Il loro rilascio è graduale ed è subordinato a un meccanismo di valutazione che ne consente lo sblocco solo previo raggiungimento di una maggioranza di voti favorevoli dei donatori sulle singole richieste di spesa presentate dall'Ente.

L'introduzione di un meccanismo di votazione ha richiesto necessariamente di valutare il rischio legato alla scarsa partecipazione dei donatori. In particolare, per evitare condizioni di stallo del sistema, si è deciso di adottare una logica di *silenzio-assenso*, progettando la piattaforma in modo tale da approvare automaticamente la richiesta di spesa anche in assenza di voti entro i termini prefissati.

Infine, un ulteriore rischio è rappresentato dall'esposizione dei fondi raccolti alla volatilità intrinseca delle criptovalute. Per ovviare a questo problema, la piattaforma Chain4Good opera esclusivamente tramite *stablecoin* (nello specifico USDC e EURC), ossia criptovalute progettate per mantenere un valore stabile rispetto a una valuta fiat di riferimento. Tale scelta è motivata dal fatto che, a differenza delle criptovalute tradizionali, come Bitcoin o ETH, le *stablecoin* permettono di garantire che il potere d'acquisto delle risorse donate rimanga inalterato dal momento della donazione fino all'effettivo utilizzo da parte del beneficiario.

4.2 Analisi dei requisiti

In questa sezione sono riportati i requisiti richiesti per il corretto funzionamento della piattaforma.
In particolare, nella Tabella 3 sono riportati i requisiti funzionali, ossia le funzionalità che la piattaforma deve implementare.
Nella Tabella 4, invece, sono descritti i requisiti non funzionali ovvero tutte le caratteristiche che, pur non essendo funzionalità dirette, il sistema deve garantire.

4.3 Architettura del Sistema

L'architettura proposta adotta un modello che integra lo *stack* MERN (MongoDB, Express, React, Node.js) con le tecnologie Web3.
In particolare, il *front-end* è sviluppato in React integrato da Tailwind CSS per la definizione di uno stile grafico resposivo, e da React Router per la gestione della navigazione *client-side.*
React Query, invece, è stato utilizzato per l'implementazione di un meccanismo di *caching* volto ad ottimizzare le prestazioni e l'esperienza utente.
L'interazione con la blockchain è invece realizzata mediante le librerie Wagmi e Viem, attraverso le quali è stato possibile gestire l'interconnessione con il *wallet* dell'utente (Metamask) e interagire direttamente con gli *smart contract* per permettere l'esecuzione di transazioni e la lettura dei dati *on-chain*.
Il *back-end* è basato su Express.js ed è sviluppato in TypeScript. L'architettura segue il *pattern* Model-View-Controller (MVC), opportunamente adattato a un sistema basato su API REST. In particolare, il livello di *Model* comprende i modelli Mongoose e l'accesso al database MongoDB, utilizzato per la persistenza dei dati *off-chain*. Il livello di *Controller* è costituito dalle funzioni associate ai singoli *endpoint* REST, responsabili dell'implementazione della logica applicativa.
Infine, il livello di *View* è rappresentato dal sistema di *routing* delle API, che espone le risorse verso il *front-end*.
Tra il livello di *routing* delle API e i controller sono inseriti specifici *middleware*, incaricati di effettuare controlli di autenticazione e autorizzazione prima dell'esecuzione della logica applicativa (ad esempio, il *middleware* `isEnte` controlla che l'utente autenticato possieda i requisiti necessari per operare come ente promotore).
Infine, per evidenti ragioni di efficienza e costi imposti dalla tecnologia blockchain è stata operata una chiara separazione tra i dati da memorizzare *on-chain* e *off-chain*. In particolare, sulla blockchain sono stati memorizzati i dati essenziali e critici, come l'insieme delle donazioni effettuate o le ri-

ID	Requisito	Descrizione
Ente		
RF1	Creazione progetto	L'Ente deve poter avviare una nuova iniziativa di raccolta fondi definendone nome, *budget target* e data di scadenza.
RF2	Richiesta di spesa	L'Ente deve poter richiedere il rilascio di una parte dei fondi avanzando una richiesta di spesa e allegando il relativo preventivo.
RF3	Prova di acquisto	L'Ente deve poter caricare la fattura che attesti l'impiego dei fondi precedentemente sbloccati.
RF4	Vincolo di sequenzialità	L'Ente non deve poter sottomettere una nuova richiesta di spesa se non ha preventivamente caricato la prova di acquisto relativa alla richiesta precedentemente approvata.
Donatore		
RF5	Visualizzazione progetti	Il donatore deve poter visualizzare l'elenco delle iniziative di CF attive e i relativi dettagli.
RF6	Donazione	Il donatore deve poter selezionare un progetto e scegliere arbitrariamente l'importo da donare.
RF7	Votazione richieste	Il donatore deve poter visualizzare il preventivo di spesa allegato dall'Ente e votare se approvare o meno la richiesta, motivando la risposta in caso di negazione.
RF8	Saldo portafoglio	Il donatore deve poter visualizzare il saldo disponibile.
Sistema		
RF9	Registrazione donazioni	Il sistema deve registrare *on-chain* ogni donazione effettuata.
RF10	Blocco dei fondi	Il sistema deve impedire il trasferimento dei fondi, previo consenso dei donatori.
RF11	Gestione votazione	Il sistema deve avviare, gestire e concludere il processo di votazione per ogni richiesta di spesa.
RF12	Erogazione automatica	Il sistema deve trasferire automaticamente i fondi al *wallet* dell'Ente, qualora la richiesta di spesa venga approvata dai donatori.
RF13	Registro operazioni	Il sistema deve registrare *on-chain* le richieste di spesa, gli esiti delle votazioni e il trasferimento dei fondi sbloccati.

Tabella 3: Tabella dei requisiti funzionali.

ID	Requisito	Descrizione
RNF1	Immutabilità	Ogni transazione relativa a donazioni, votazioni e rilascio di fondi deve essere registrata su un registro distribuito in modo permanente e non modificabile.
RNF2	Integrità dei dati	I file pesanti, come preventivi e fatture, devono essere memorizzati *off-chain*. Il sistema deve garantire che tali documenti siano riconducibili in modo univoco alle relative operazioni registrate *on-chain*, impedendone la manipolazione.
RNF3	Usabilità	La *Webapp* deve consentire agli utenti di consultare i dati *on-chain*, come lo storico delle donazioni effettuate, attraverso interfacce intuitive.
RNF4	Sicurezza	L'accesso alle funzionalità della piattaforma e alla consultazione dettagliata dei dati deve essere limitato ai soli utenti autenticati.
RNF5	Portabilità	La *Webapp* deve essere fruibile sia da dispositivi *desktop* che *mobile*.

Tabella 4: Tabella dei requisiti non funzionali.

chieste di spesa avanzate dall'ente. Il database MongoDB invece è stato utilizzato per memorizzare rispettivamente: i metadati dei progetti e delle spese (come la descrizione dettagliata delle richieste di spesa), la logica non critica, i dati dell'utente e i documenti relativi alle prove di acquisto.

4.4 Stack tecnologico

In questa sezione viene fornita una panoramica delle tecnologie impiegate per lo sviluppo della piattaforma.
In particolare, le tabelle 5 e 6 riportano gli strumenti utilizzati rispettivamente per l'implementazione della componente *front-end* e *back-end* del *software*, con una breve descrizione del loro impiego durante le fasi di sviluppo. Le tecnologie relative alla componente *blockchain* sono invece sintetizzate in Tabella 7.
Infine, nella Tabella 8 sono riportati gli strumenti utilizzati per supportare l'organizzazione del lavoro tra i membri del gruppo e per garantire la portabilità del sistema *software*.

Tecnologia	**Descrizione**
TypeScript	Linguaggio utilizzato per lo sviluppo del *front-end*.
React.js	Libreria utilizzata per la realizzazione dell'interfaccia utente secondo un'architettura a componenti.
React Router	Libreria di *routing* configurata in modalità *framework* per la gestione della navigazione *client-side*.
@tanstack/react-query	Libreria utilizzata per la gestione delle chiamate asincrone al *back-end*, con supporto a *caching*, sincronizzazione dei dati e gestione automatica degli stati di caricamento ed errore.
Tailwind CSS	*Framework* CSS utilizzato per la definizione dello stile grafico e la realizzazione di *layout* responsivi e coerenti, integrando una configurazione personalizzata.
Vite	Strumento di *build* utilizzato per la compilazione del codice TypeScript e l'esecuzione dell'applicazione durante la fase di sviluppo.
Wagmi	Libreria utilizzata per l'interazione con i *wallet* e la *blockchain* (Web3) attraverso *hooks* in React.
viem	*Client* RPC a basso livello, utilizzato internamente da Wagmi per il recupero dei dati *on-chain*.
SIWE	Libreria utilizzata per l'implementazione del protocollo *Sign-In with Ethereum*, adottato per verificare l'identità dell'utente tramite firma crittografica del *wallet*.

Tabella 5: Tecnologie utilizzate per lo sviluppo del *front-end*.

Tecnologia	**Descrizione**
TypeScript	Linguaggio utilizzato per lo sviluppo del *back-end*, al fine di garantire tipizzazione statica e maggiore robustezza del codice.
Express.js	*Framework* per Node.js utilizzato per la realizzazione delle API REST e per la gestione delle richieste provenienti dal *front-end*.
MongoDB	Database NoSQL utilizzato per la memorizzazione dei dati *off-chain*, quali metadati dei progetti, informazioni sugli utenti e dati di sessione.
Mongoose	*Object Data Modeling (ODM)* utilizzato per la definizione dei modelli di dati e l'interazione con il database MongoDB.
express-session	*Middleware* utilizzato per la gestione delle sessioni lato *server*, impiegato nel processo di autenticazione.
SIWE	Libreria utilizzata per l'implementazione del protocollo *Sign-In with Ethereum*, basato sulla verifica di messaggi firmati tramite *wallet*.
ethers.js	Libreria JavaScript utilizzata per l'interazione con la *blockchain* Ethereum, in particolare per il recupero di informazioni *on-chain*.
Multer	*Middelware* per Node.js utilizzato per la gestione di flussi *multipart/form-data*, facilitando l'*upload* e il salvataggio dei contenuti multimediali.
node-cron	*Scheduler* JavaScript impiegato per la realizzazione di un worker dedicato al monitoraggio automatico dello stato di progetti e spese.

Tabella 6: Tecnologie utilizzate per lo sviluppo del *back-end*.

Tecnologia	Descrizione
Solidity	Linguaggio utilizzato per lo sviluppo degli *Smart Contract*.
Hardhat	Ambiente di sviluppo utilizzato per la compilazione, il *testing* e il *deployment* degli *Smart Contract*, nonché per la simulazione di una *blockchain* locale.
@openzeppelin/contracts	Libreria di *Smart Contract* riutilizzabili, utilizzata per integrare componenti standard e meccanismi di sicurezza.
Mocha	*Framework* di *testing* utilizzato come motore per l'esecuzione dei *test* automatizzati degli *Smart Contract*.
Chai	Libreria di asserzione utilizzata per verificare che l'*output* degli *Smart Contract* corrisponda ai requisiti attesi.

Tabella 7: Tecnologie utilizzate per la componente *blockchain*.

Strumento	Descrizione
Visual Studio Code	Ambiente di Sviluppo Integrato (IDE) principale, configurato con estensioni specifiche per lo sviluppo dello *stack* tecnologico.
Git	*Version Control System (VCS)* impiegato per la gestione del codice sorgente secondo una strategia di *branching* collaborativa per permettere lo sviluppo parallelo.
GitHub	Piattaforma di *hosting* del *repository* remoto, utilizzata per supportare la collaborazione tra i membri del gruppo.
Docker	Piattaforma di containerizzazione utilizzata per la standardizzazione e l'isolamento dell'ambiente di esecuzione.

Tabella 8: Strumenti utilizzati per lo sviluppo.

5 Implementazione e prototipo

5.1 Login e autenticazione

L'autenticazione all'interno della piattaforma adotta il protocollo Sign-In with Ethereum (SIWE) sostituendo il modello tradizionale basato sull'utilizzo di credenziali tradizionali quali *username* e *password*, coerentemente con il paradigma Web3. L'identità dell'utente è infatti associata in modo univoco al possesso di un *wallet*.
Il flusso di *login* ha inizio con la connessione del *wallet* dell'utente, a partire dal quale il *front-end* ottiene l'indirizzo pubblico associato. Tale indirizzo viene inviato al *back-end*, che a sua volta, genera un *nonce*, ovvero un valore casuale e univoco utilizzato per prevenire attacchi di tipo *replay*. Il *nonce* viene associato temporaneamente all'utente, tramite sessione o persistenza nel database, e restituito al *front-end*.
Successivamente, il *front-end* costruisce un messaggio conforme allo standard SIWE, includendo il *nonce* ricevuto, e richiede all'utente di firmarlo tramite il proprio *wallet*. Il messaggio firmato e la relativa firma vengono quindi inviati al *back-end*, che procede alla verifica dell'autenticazione. In particolare, il *server* controlla sia che l'indirizzo estratto crittograficamente dalla firma coincida con quello dichiarato sia che il *nonce* contenuto nel messaggio corrisponda a quello attualmente valido, impedendo il riutilizzo di firme precedenti.
Solo se entrambi i controlli vanno a buon fine, l'autenticazione viene considerata valida e viene avviata una sessione applicativa. In fase di primo accesso, l'utente viene registrato automaticamente nel sistema, mentre per accessi successivi vengono aggiornate le informazioni di sessione. Al termine del processo, il *nonce* viene rigenerato e quello precedente invalidato, garantendo la sicurezza delle autenticazioni future.
Infine, il *front-end* utilizza il *cookie* di sessione per recuperare le informazioni dell'utente autenticato e completare il processo di accesso all'applicazione.

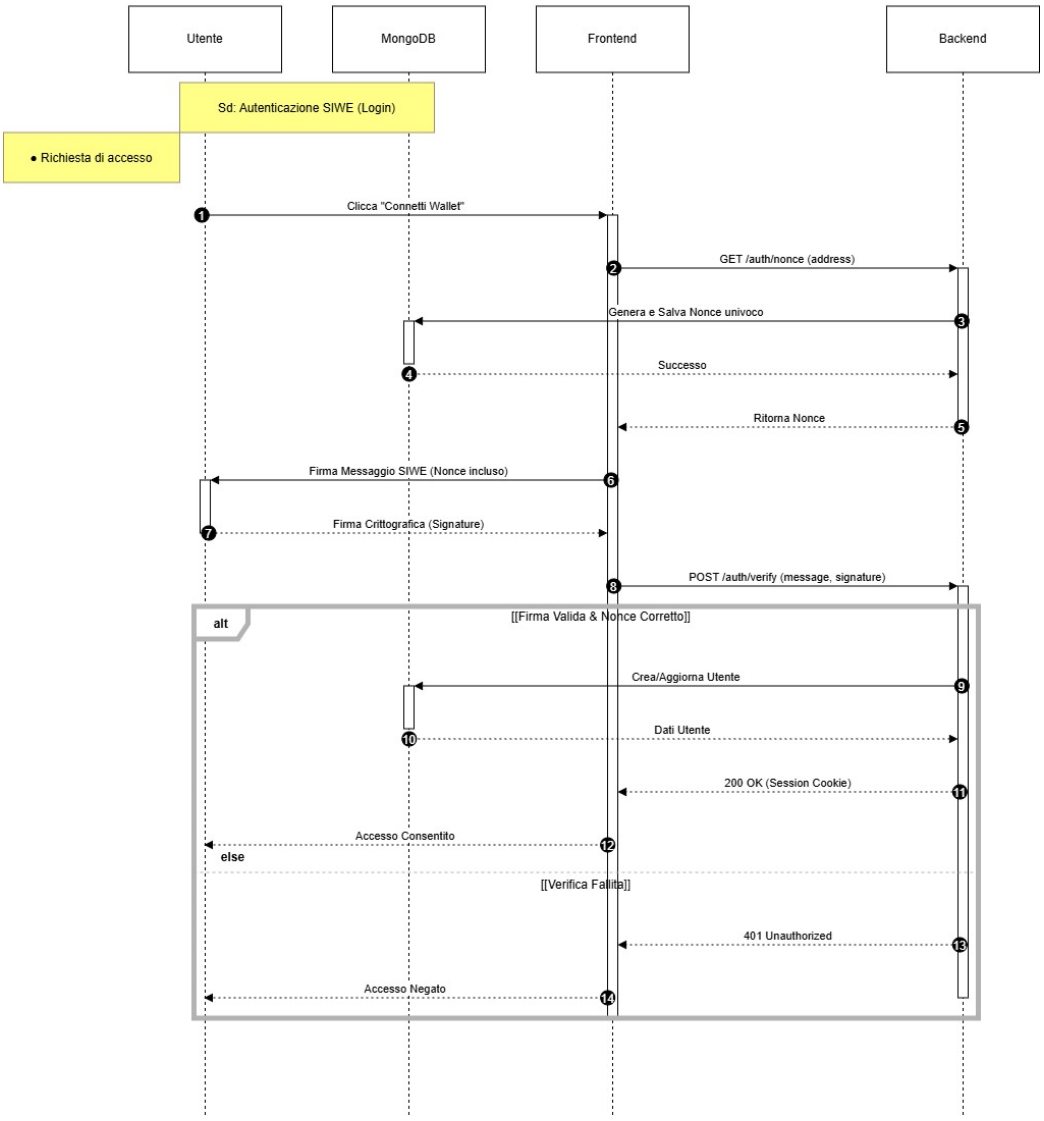


Figura 1: Diagramma di sequenza del processo di *login*.

5.1.1 Ente

L'Ente beneficiario rappresenta l'attore abilitato alla creazione delle iniziative di raccolta fondi sulla piattaforma.
Il suo riconoscimento avviene durante la fase di autenticazione, contestualmente alla verifica della firma crittografica dell'utente.
In particolare, durante la procedura di *login*, il *back-end* accerta l'identità dell'utente e interroga la blockchain per verificare se l'indirizzo del *wallet* risulti associato a un NFT (protocollo ERC721) non trasferibile (*Soulbound Token*) denominato "Ente autorizzato Chain4Good (ETS). Tale *token* funge dunque da attestazione crittografica dello stato di ente autorizzato all'interno del sistema e viene rilasciato esclusivamente a seguito di un processo di registrazione e verifica *off-chain*.
Una volta autenticato, il sistema abilita le funzionalità riservate a questa tipologia di utente, il quale può avviare nuove raccolte o contribuire ai progetti di altri enti, mentre gli è preclusa la possibilità di effettuare donazioni verso le proprie iniziative.

5.2 Dashboard donatore

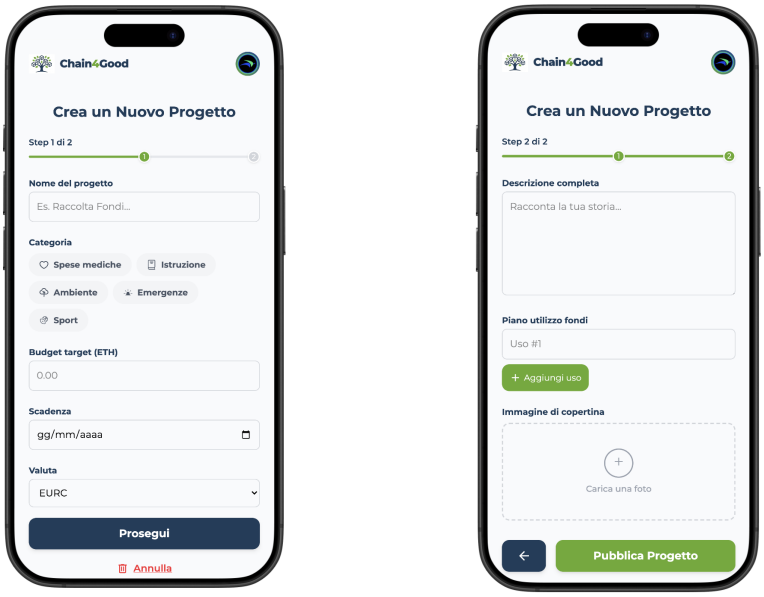
La *dashboard* del donatore è l'interfaccia attraverso la quale gli utenti possono interagire direttamente con la piattaforma a seguito della procedura di autenticazione. Essa è progettata per fornire una visione sintetica e intuitiva delle iniziative di CF attive, permettendo all'utente di monitorarne lo stato di avanzamento attraverso indicatori chiave quali il *budget* raccolto, il numero di donatori coinvolti e la data di scadenza.
Una volta selezionata l'iniziativa di interesse, l'utente ha la possibilità di procedere con l'operazione di donazione. In particolare, il sistema adotta il meccanismo di raccolta *keep-it-all*, in base al quale l'Ente beneficiario conserva i fondi raccolti anche qualora l'obiettivo economico prefissato non venga raggiunto entro i termini prestabiliti. Tale scelta progettuale risulta coerente con la categoria di Enti autorizzati alla creazione delle iniziative sulla piattaforma, individuati esclusivamente negli Enti del Terzo Settore, per i quali anche contributi parziali possono risultare funzionali al perseguimento delle finalità sociali.

Figura 2: Dashboard del donatore

5.3 Creazione progetto

La creazione di un progetto costituisce l'atto attraverso il quale l'Ente formalizza la propria proposta sulla piattaforma. Tale procedura si articola in due *step* sequenziali (3):

- (a): durante il primo *step*, l'Ente è tenuto a specificare le informazioni fondamentali del progetto, quali il nome, la categoria di appartenenza, la valuta, l'obiettivo economico e il termine temporale della raccolta fondi. Gli ultimi due parametri permettono di automatizzare la gestione delle risorse in modalità *trustless*, in quanto vengono utilizzati dallo *Smart Contract* per determinare l'esito della raccolta fondi. Inoltre, la piattaforma è stata progettata per operare specificamente con le *Stablecoin* EURC.
- (b): lo *step* 2 prevede l'inserimento di un piano dettagliato delle spese, oltre ad una descrizione approfondita del progetto e un'immagine di copertina. Tale prospetto non assolve solo finalità informative, ma contribuisce ad incrementare la credibilità dell'iniziativa e a consolidare il rapporto di fiducia con i donatori.

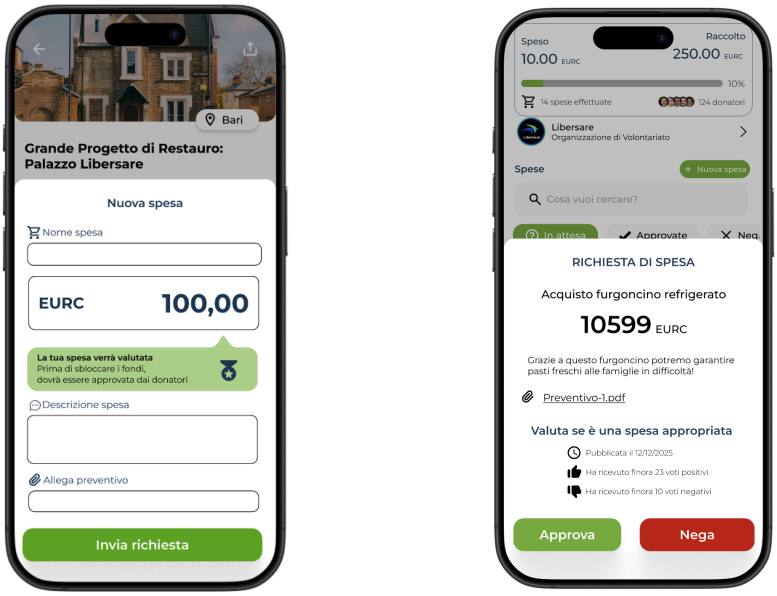


(a) (b)

Figura 3: Inserimento di un nuovo progetto.

5.4 Inserimento e valutazione spesa

A differenza dei sistemi centralizzati in cui l'Ente ha piena e immediata disponibilità del *budget* donato, l'architettura proposta prevede che i fondi raccolti rimangano vincolati all'interno di uno *Smart Contract*. Per poter accedere a tali risorse, il beneficiario deve presentare una "Richiesta di Spesa" (4) specificandone il nome, l'importo richiesto, la finalità dell'esborso e il relativo preventivo. La sottomissione della richiesta è consentita esclusivamente quando il progetto si trova in uno stato attivo, ossia a seguito del raggiungimento del *target* di raccolta oppure al termine della scadenza temporale prevista per la campagna.
A seguito della sottomissione, la richiesta viene poi sottoposta a un meccanismo di valutazione decentralizzato, basato sul voto dei donatori che hanno contribuito al finanziamento del progetto. Coerentemente con la natura della piattaforma, il sistema di voto è stato implementato in modo da attribuire lo stesso peso a ciascun utente, indipendentemente dall'importo della donazione. Questa scelta progettuale è finalizzata a evitare che il potere decisionale risulti influenzato dalla capacità economica dei singoli utenti. Al fine di assicurare un utilizzo progressivo e verificabile delle risorse raccolte, inoltre, la presentazione di una nuova richiesta di spesa è subordinata al caricamento della prova di acquisto relativa all'ulti-



(a) (b)

Figura 4: Inserimento di una nuova richiesta di spesa.

ma richiesta precedentemente approvata. Tale approccio mira a garantire un utilizzo progressivo e verificabile dei fondi raccolti, e ad assicurare la conformità delle spese per le finalità dichiarate.

5.4.1 Meccanismo di votazione

Per evitare condizioni di stallo decisionale, il sistema è stato progettato in modo da limitare la finestra di votazione a un periodo di tre giorni dalla sottomissione della richiesta di spesa.
La gestione dell'intero processo è invece affidata allo *Smart Contract* il quale opera secondo la logica descritta nell'algoritmo 1.
Al verificarsi di una delle condizioni di approvazione, il contratto esegue in modo autonomo e irreversibile il trasferimento della somma richiesta verso il *wallet* del beneficiario, registrando l'operazione *on-chain*.

Algoritmo 1 Logica di approvazione o rifiuto delle richieste di spesa.

Require: vFor, vAgainst, nTot, tNow, tEnd, reqEndTime
Ensure: reqApproved

```
1: simpleMaj ← vFor ⩾ vAgainst
2: mathMaj ← vFor ⩾ nTot - (vFor + vAgainst) and vFor ⩾ 0;
3: tEnd ← tNow ⩾ reqEndTime

4: if tEnd and simpleMaj then
5:     reqApproved ← true
6: end if

7: if not tEnd and mathMaj then
8:     reqApproved ← true
9: end if
```

6 Validazione e discussione

Gli obiettivi di progetto sono stati raggiunti con successo. L'applicazione web decentralizzata soddisfa i principali requisiti definiti in fase di analisi, in quanto fornisce un sistema in grado di garantire trasparenza, tracciabilità delle donazioni e coinvolgimento attivo degli utenti.
Complessivamente, lo sviluppo della piattaforma è stato condotto ponendo particolare attenzione agli aspetti di sicurezza, affidabilità e qualità del *software*. In particolare, sono state adottate misure preventive volte a mitigare le principali vulnerabilità applicative, quali attacchi di tipo Cross-Site Scripting (XSS), SQL injection e *replay attack*, con un focus specifico sui meccanismi di autenticazione.

Particolare attenzione è stata dedicata alle prestazioni e all'esperienza utente. La gestione della *cache* è stata progettata in modo da garantire un accesso efficiente ai dati, prevedendo meccanismi di invalidazione automatica al superamento di determinate soglie temporali, contribuendo così a una maggiore fluidità dell'applicazione.
Analogamente, l'adozione corretta di *framework* orientati al *server-side rendering* ha consentito di migliorare la reattività dell'interfaccia. L'interazione con l'utente è stata progettata, inoltre, per gestire esplicitamente tutti gli stati possibili dell'applicazione, inclusi caricamenti, errori e assenza di risultati, fornendo sempre messaggi informativi chiari.

L'attività di *testing* è stata effettuata progressivamente durante lo sviluppo, piuttosto che sul funzionamento dell'intera piattaforma. In particolare, ogni funzionalità è stata testata immediatamente dopo la sua implementazione attraverso test manuali e mediante l'utilizzo di strumenti di sviluppo dedicati, come *Postman* e richieste HTTP da riga di comando, per validare il corretto comportamento delle API *backend*. Inoltre, sono stati eseguiti test automatizzati sugli *Smart Contract* prima della fase di *deploy*, così da individuare eventuali errori logici o comportamenti inattesi in un ambiente controllato.

Infine, il codice è stato organizzato secondo una struttura modulare e ordinata, con una gestione coerente di file, cartelle, al fine di favorire la manutenibilità e l'evoluzione futura del progetto. Sono state inoltre utilizzate le versioni più recenti e stabili delle librerie adottate. L'intero sistema è stato concepito per essere facilmente containerizzato mediante *Docker* e *docker-compose*, anche se alcune funzionalità risultano volutamente incomplete o ulteriormente ottimizzabili, in linea con la natura prototipale del progetto.

7 Conclusioni e sviluppi futuri

Il presente lavoro ha affrontato la progettazione e lo sviluppo di Chain4Good, una piattaforma di CF basata su tecnologia blockchain, concepita per rispondere alle criticità strutturali dei sistemi di raccolta fondi tradizionali, con particolare riferimento al contesto filantropico e al Terzo Settore. L'implementazione realizzata ha consentito di verificare la fattibilità del modello proposto e di dimostrare come l'impiego di questa tecnologia possa costituire un valido strumento per aumentare la trasparenza e la sicurezza dell'intero processo di donazione. Tuttavia, in ragione della la sua natura sperimentale è possibile evidenziare diverse possibilità di sviluppo volte a migliorare l'efficacia complessiva della piattaforma.

Un primo ambito sviluppo riguarda la gestione completa del ciclo di vita dei progetti. In particolare, il sistema potrebbe essere esteso per gestire scenari attualmente non implementati, quali la cancellazione di un progetto, la disabilitazione di un ente promotore, la modifica del *budget* target o della scadenza temporale, nonché i casi in cui una campagna non raggiunga l'obiettivo economico prefissato. L'introduzione di tali funzionalità consentirebbe di rendere la piattaforma più robusta e aderente a scenari reali.

Dal punto di vista della trasparenza e del controllo delle spese, un possibile miglioramento potrebbe prevedere l'integrazione di tecniche di intelligenza artificiale per la verifica automatica della documentazione caricata *off-chain*, al fine di individuare potenziali anomalie o incongruenze tra il contenuto di fatture e preventivi caricati dall'Ente e gli obiettivi dichiarati all'avvio dell'iniziativa (nella sezione "Piano utilizzo fondi").

Ulteriori sviluppi riguardano il miglioramento dell'interazione con i donatori. L'introduzione di un sistema di notifiche permetterebbe di informare gli utenti in tempo reale sui cambiamenti di stato dei progetti, sull'apertura di nuove richieste di spesa o sull'esito delle votazioni. Allo stesso modo, la possibilità per gli Enti di pubblicare aggiornamenti periodici, corredati da testi e immagini, consentirebbe di rafforzare il coinvolgimento della comunità e di rendere il processo di donazione più partecipativo e trasparente.

Infine, i risultati ottenuti evidenziano come l'architettura proposta non sia strettamente limitata al dominio applicativo di riferimento. Infatti, sebbene Chain4Good sia stata progettata specificamente per il Terzo Settore, il modello adottato potrebbe essere generalizzato e applicato anche ad altri scenari di raccolta fondi, includendo iniziative promosse da soggetti privati.

In questo senso, l'approccio proposto rappresenta una possibile base per lo sviluppo di piattaforme di CF più aperte e trasparenti, in grado di estendere i benefici della tecnologia blockchain a una platea di utilizzatori sempre più vasta.

Riferimenti bibliografici

[1] V. Patil, V. Gupta e R. Sarode, «Blockchain-based crowdfunding application,» in *2021 Fifth international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)*, IEEE, 2021, pp. 1546–1553.

[2] Y. Thalassinos, «Crowdfunding: A Different Approach to Investment,» *European Research Studies Journal*, vol. 26, n. 2, pp. 318–333, 2023.

[3] F. Brunetti, «Web 2.0 as Platform for the Development of Crowdfunding,» in *Crowdfunding for SMEs: A European Perspective*, Springer, 2016, pp. 45–60.

[4] O. K. Alia, D. M. Suleiman e H. A. Noman, «IHSAN: A Secure and Transparent Crowdfunding Platform Leveraging Comprehensive Decentralized Technologies,» *IEEE Access*, 2024.

[5] N. Salido-Andres, M. Rey-Garcia, L. I. Alvarez-Gonzalez e R. Vazquez-Casielles, «Mapping the field of donation-based crowdfunding for charitable causes: systematic review and conceptual framework,» *VOLUNTAS: International Journal of Voluntary and Nonprofit Organizations*, vol. 32, n. 2, pp. 288–302, 2021.

[6] S. Hohen, C. Hüning e L. Schweizer, «Reward-based crowdfunding—a systematic literature,» 2025.

[7] M. Kuti, Z. Bedő e D. Geiszl, «Equity-based crowdfunding,» *Financial and Economic Review*, vol. 16, n. 4, pp. 187–200, 2017.

[8] M. Hossain e G. O. Oparaocha, «Crowdfunding: Motives, definitions, typology and ethical challenges,» *Entrepreneurship Research Journal*, vol. 7, n. 2, p. 20 150 045, 2017.

[9] A. Rejeb, K. Rejeb, A. Appolloni, S. Zailani e M. Iranmanesh, «Mapping the research landscape of blockchain and crowdfunding,» *Financial Innovation*, vol. 11, n. 1, p. 22, 2025.

[10] K. Mukherjee, A. Rana e S. Rani, «Crowdfunding Platform using Blockchain,» in *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, IEEE, 2024, pp. 1–6.

[11] P. Shelke, S. Zanjal, R. Patil, D. Desai, H. Chavan e V. Kulkarni, «Blockchain technology based crowdfunding using smart contracts,» in *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, IEEE, 2022, pp. 939–943.

[12] S. Nakamoto, B. Bit et al., «Bitcoin: A peer-to-peer electronic cash system,» *2008*, 2007.

[13] R. Rodrigues e P. Druschel, «Peer-to-peer systems,» *Communications of the ACM*, vol. 53, n. 10, pp. 72–82, 2010.

[14] A. A. Monrat, O. Schelén e K. Andersson, «A survey of blockchain from the perspectives of applications, challenges, and opportunities,» *Ieee Access*, vol. 7, pp. 117 134–117 151, 2019.

[15] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman et al., «Blockchain technology: Beyond bitcoin,» *Applied innovation*, vol. 2, n. 6-10, p. 71, 2016.

[16] N. Yadav e V. Sarasvathi, «Venturing crowdfunding using smart contracts in blockchain,» in *2020 third international conference on smart systems and inventive technology (ICSSIT)*, IEEE, 2020, pp. 192–197.