



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

Acronimi	vi
1 Introduzione	1
2 Stato dell'arte	2
2.1 Crowdfunding	2
2.1.1 Limiti delle piattaforme tradizionali di CF	3
2.2 Crowdfunding e blockchain	4
2.2.1 Soluzioni esistenti	5
3 Metodologia di progetto	7
3.1 Modello di processo	7
3.1.1 Organizzazione del team	7
3.2 Pianificazione delle attività	8
3.2.1 Stima dei costi	8
3.3 Analisi dei rischi	10
3.3.1 Rischi di progetto	10
3.3.2 Rischi di dominio	10
4 Progettazione e implementazione	12
4.1 L'obiettivo di Chain4Good	12
4.2 Analisi dei requisiti	12
4.2.1 Requisiti funzionali	12
4.2.2 Requisiti non funzionali	14
4.3 Architettura del Sistema	15
4.4 Stack tecnologico	16
4.4.1 Front-end	16
4.4.2 Back-end	17
4.4.3 Blockchain	18
4.4.4 Strumenti di sviluppo	19
5 Implementazione e prototipo	20
5.1 Login e autenticazione	20
5.2 Dashboard donatore	22
5.3 Creazione progetto	24
5.4 Inserimento e valutazione spesa	24
5.4.1 Meccanismo di validazione	25
6 Validazione e discussione	27

7 Conclusioni e sviluppi futuri	29
Riferimenti bibliografici	31

Elenco delle figure

1	Diagramma di sequenza del processo di <i>login</i>	21
2	Dashboard del donatore	23
3	Inserimento di un nuovo progetto	25
4	Valutazione di una richiesta di spesa	26

Elenco delle tabelle

1	Caratteristiche della tecnologia <i>blockchain</i>	5
2	Pianificazione delle attività di progetto.	9
3	Tecnologie e librerie utilizzate per lo sviluppo del front-end.	16
4	Tecnologie e librerie utilizzate per lo sviluppo del <i>back-end</i> . .	17
5	Tecnologie utilizzate per la componente <i>blockchain</i>	18
6	Strumenti utilizzati per lo sviluppo del progetto	19

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

Negli ultimi anni, il **Crowdfunding (CF)** si è affermato come 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 crescente diffusione, le piattaforme di **CF** tradizionali presentano criticità strutturali riconducibili principalmente alla loro architettura centralizzata. In tali sistemi, la gestione dei fondi e delle informazioni è demandata a intermediari fiduciari, limitando la trasparenza sull'utilizzo delle risorse raccolte e riducendo il controllo effettivo dei donatori sull'intero ciclo di vita della donazione. Questo aspetto risulta particolarmente critico nel **CF** filantropico, dove la fiducia e la tracciabilità rappresentano elementi essenziali per garantire la credibilità delle iniziative e incentivare la partecipazione degli utenti.

In questo scenario, la tecnologia blockchain si configura come una soluzione promettente per il superamento di tali limiti. Grazie alle sue proprietà intrinseche di decentralizzazione, immutabilità, trasparenza e sicurezza, la blockchain consente di realizzare sistemi in cui le operazioni risultano verificabili e resistenti a manomissioni, riducendo la necessità di intermediari centrali. L'impiego degli *Smart Contract*, inoltre, permette di automatizzare la gestione delle donazioni e dei flussi finanziari sulla base di regole predefinite e pubblicamente verificabili.

Alla luce di queste considerazioni, il presente lavoro propone *Chain4Good*, una piattaforma di **CF** basata su tecnologia blockchain, progettata specificamente per il Terzo Settore. L'obiettivo principale è restituire al donatore un ruolo attivo e continuo nel processo di donazione, garantendo al contempo trasparenza e tracciabilità nell'utilizzo dei fondi. A differenza dei modelli tradizionali, *Chain4Good* adotta un approccio ibrido che combina meccanismi di verifica *off-chain* sugli enti promotori con una *governance* decentralizzata *on-chain*, basata su *Smart Contract* e processi di approvazione partecipativa delle richieste di spesa.

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 determinante 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 architettonicale 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;

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 fondate 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 <i>blockchain permissionless</i>) oppure dai membri autorizzati (nelle <i>blockchain permissioned</i>) [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.

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.

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, associan-
do 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 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.

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 <i>blockchain</i>	10
Fase 2 – Progettazione		
T6	Progettazione dell'architettura del sistema	10
T7	<i>Design</i> delle interfacce utente	20
Fase 3 – Implementazione del sistema		
T8	Sviluppo del <i>Back-end</i>	60
T9	Sviluppo del <i>Front-end</i>	40
T10	Redazione della documentazione tecnica	50
Fase 4 – Test e miglioramenti		
T11	Verifica del prodotto e delle funzionalità implementate	7
T12	Correzione dei <i>bug</i> e perfezionamenti	6
Fase 5 – Conclusione		
T13	Consegna	10
Totale ore		250

Tabella 2: Pianificazione delle attività di progetto.

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.

3.3.1 Rischi di progetto

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.

3.3.2 Rischi di dominio

Oltre ai rischi di progetto, sono stati analizzati quelli legati al dominio applicativo del CF. In particolare:

- **Asimmetria informativa:** sulla base di quanto emerso dall'analisi in letteratura, è stata considerata la possibilità che l'Ente promotore non utilizzasse i fondi raccolti per le finalità dichiarate. Per mitigare tale rischio si è deciso di implementare un meccanismo di custodia decentralizzata dei fondi, affinché le risorse raccolte fossero sbloccate esclusivamente a seguito dell'approvazione delle richieste di spesa da parte dei donatori. Al fine di rafforzare ulteriormente la trasparenza del processo di donazione, si è deciso di vincolare la presentazione di nuove richieste di spesa al caricamento del preventivo relativo alla richiesta precedente;

- **Inerzia dei votanti:** poiché l'espressione del voto comporta il sostentamento di un costo di transazione, è stata considerata la possibilità che gli utenti non partecipassero attivamente al processo decisionale. 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;
- **Volatilità finanziaria:** l'impiego di criptovalute per effettuare donazioni espone i fondi raccolti alle elevate fluttuazioni di mercato. Per neutralizzare tale rischio, la piattaforma è stata progettata per operare esclusivamente tramite *stablecoin* (es. USDC), criptovalute progettate per mantenere un valore stabile rispetto a una valuta reale di riferimento (ad esempio, 1 USDC = 1 Dollaro USA);

4 Progettazione e implementazione

4.1 L'obiettivo di Chain4Good

Chain4Good è una piattaforma decentralizzata di **CF** nata per superare le criticità intrinseche dei sistemi di raccolta fondi tradizionali. Il suo obiettivo principale è restituire al donatore un ruolo attivo lungo l'intero ciclo di vita della donazione, mitigando il problema della limitata tracciabilità nell'utilizzo dei fondi tipico dei sistemi centralizzati.

A differenza dei modelli tradizionali, nei quali le risorse vengono trasferite integralmente all'Ente beneficiario al termine della raccolta, in Chain4Good l'erogazione dei fondi avviene in maniera incrementale ed è subordinata a un processo di approvazione decentralizzato. In tale contesto, lo sblocco delle risorse è vincolato all'espressione del consenso dei donatori sulle singole richieste di spesa.

E' importante sottolineare che tale meccanismo non è esente da potenziali comportamenti fraudolenti. La tecnologia *blockchain*, difatti, non è in grado di garantire la veridicità dei dati forniti *off-chain*, quali i preventivi allegati alle richieste di spesa. Tuttavia, essa consente di rendere l'intero processo di richiesta, approvazione ed erogazione delle risorse immutabile, trasparente e pubblicamente verificabile, grazie alla registrazione *on-chain* di ogni operazione e di ogni trasferimento di fondi. In questo modo, al donatore è permesso di certificare la congruità tra gli obiettivi dichiarati e quelli effettivamente perseguiti.

Chain4Good, dunque, si propone come una piattaforma capace di ridefinire il concetto stesso di donazione, il quale non si configura più come un mero atto di fiducia, bensì come un processo intrinsecamente sicuro e verificabile in ogni sua fase.

4.2 Analisi dei requisiti

In questa sezione sono riportati per punti i requisiti richiesti per il corretto funzionamento della piattaforma.

4.2.1 Requisiti funzionali

I requisiti funzionali definiscono le funzionalità che la piattaforma deve implementare. Essi vengono di seguito categorizzati in base agli attori che interagiscono con il sistema.

1. l'Ente Beneficiario:

- **Creazione progetto:** l'Ente deve poter avviare una nuova iniziativa di raccolta fondi definendone nome, *budget target* e data di scadenza;
- **Inserimento di una richiesta di spesa:** l'Ente deve poter richiedere il rilascio di una parte dei fondi raccolti avanzando una richiesta di spesa e allegando il relativo preventivo;
- **Caricamento della prova di acquisto:** l'Ente deve poter caricare la fattura che attesti l'effettivo impiego dell'importo richiesto per lo scopo dichiarato;
- **Vincolo di sequenzialità sulle richieste di spesa:** 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;

2. per i Donatori:

- **Visualizzazione dei progetti:** il donatore deve poter visualizzare l'elenco delle iniziative di **CF** attive e i relativi dettagli;
- **Donazione ad un progetto:** il donatore deve poter selezionare un progetto e scegliere arbitrariamente l'importo da donare;
- **Votazione delle richieste di spesa:** il donatore deve poter visualizzare il preventivo di spesa allegato dall'Ente ed esprimere una preferenza (se favorevole o contrario);
- **Visualizzazione del portafoglio:** il donatore deve poter visualizzare il saldo disponibile;

3. per il Sistema (logica implementata tramite *Smart Contract*):

- **Registrazione delle donazioni:** il sistema deve registrare *on-chain* ogni donazione effettuata;
- **Blocco dei fondi:** il sistema deve impedire il trasferimento dei fondi, previo consenso dei donatori;
- **Gestione del processo di votazione:** il sistema deve avviare, gestire e concludere il processo di votazione per ogni richiesta di spesa;

- **Erogazione automatica dei fondi:** il sistema deve trasferire automaticamente i fondi al *wallet* dell'Ente, qualora la richiesta di spesa venga approvata dai donatori;
- **Registrazione delle operazioni:** il sistema deve registrare *on-chain* le richieste di spesa, gli esiti delle votazioni e il trasferimento dei fondi sbloccati;

4.2.2 Requisiti non funzionali

Di seguito si riporta l'elenco dei requisiti non funzionali, ossia tutte le caratteristiche che pur non essendo funzionalità, il sistema deve garantire.

1. **Immutabilità:** ogni transazione relativa a donazioni, votazioni e rilascio di fondi deve essere registrata su un registro distribuito in modo permanente e non modificabile;
2. **Integrità dei dati:** i file pesanti, come preventivi e prove d'acquisto, 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;
3. **Usabilità:** la *Webapp* deve consentire agli utenti di consultare i dati *on-chain*, come lo storico delle donazioni effettuate, attraverso interfacce intuitive;
4. **Sicurezza:** l'accesso alle funzionalità della piattaforma e alla consultazione dettagliata dei dati deve essere limitato ai soli utenti autenticati;
5. **Portabilità:** la *Webapp* deve essere fruibile sia da dispositivi *desktop* che *mobile*;

4.3 Architettura del Sistema

4.4 Stack tecnologico

Prima di poter procedere alla progettazione dell'architettura del sistema da realizzare si è resa necessaria l'individuazione delle tecnologie da utilizzare in fase di sviluppo per poter comprendere come queste potessero interagire tra loro e soddisfare tutti i requisiti funzionali e non funzionali emersi dalla precedente fase di analisi.

4.4.1 Front-end

La Tabella 3 riassume le principali tecnologie adottate per lo sviluppo del front-end della piattaforma.

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

Tabella 3: Tecnologie e librerie utilizzate per lo sviluppo del front-end.

4.4.2 Back-end

La Tabella 4 riassume le principali tecnologie e librerie adottate per lo sviluppo del *back-end* della piattaforma.

Tecnologia	Descrizione
TypeScript	Linguaggio utilizzato per lo sviluppo del <i>back-end</i> , 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 <i>front-end</i> .
MongoDB	Database NoSQL utilizzato per la memorizzazione dei dati <i>off-chain</i> , quali metadati dei progetti, informazioni sugli utenti e dati di sessione.
Mongoose	<i>Object Data Modeling (ODM)</i> 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 <i>server</i> , impiegato nel processo di autenticazione.
SIWE	Libreria utilizzata per l'implementazione del protocollo <i>Sign-In with Ethereum</i> , basato sulla verifica di messaggi firmati tramite <i>wallet</i> .
ethers.js	Libreria JavaScript utilizzata per l'interazione con la <i>blockchain Ethereum</i> , in particolare per il recupero di informazioni <i>on-chain</i> .
Multer	Middleware per Node.js utilizzato per la gestione di flussi <i>multipart/form-data</i> , facilitando l' <i>upload</i> e il salvataggio dei contenuti multimediali.

Tabella 4: Tecnologie e librerie utilizzate per lo sviluppo del *back-end*.

4.4.3 Blockchain

La Tabella 5 riassume le tecnologie e le librerie adottate per l'implementazione della componente *blockchain* del sistema.

Tecnologia	Descrizione
Solidity	Linguaggio utilizzato per lo sviluppo degli <i>Smart Contract</i> .
Hardhat	Ambiente di sviluppo utilizzato per la compilazione, il <i>testing</i> e il <i>deployment</i> degli <i>Smart Contract</i> , nonché per la simulazione di una <i>blockchain</i> locale.
@openzeppelin/contracts	Libreria di <i>Smart Contract</i> riutilizzabili, utilizzata per integrare componenti standard e meccanismi di sicurezza.
Mocha e Chai	Framework di <i>testing</i> (Mocha) e libreria di asserzione (Chai) utilizzati per la validazione automatizzata della logica degli <i>Smart Contract</i> .

Tabella 5: Tecnologie utilizzate per la componente *blockchain*.

4.4.4 Strumenti di sviluppo

Visual Studio Code è stato utilizzato come [Ambiente di Sviluppo Integrato \(IDE\)](#) principale.

Per favorire lo sviluppo parallelo e la portabilità del sistema *software*, invece, sono stati utilizzati strumenti riportati in Tabella 6.

Strumento	Descrizione
Git	<i>Version Control System (VCS)</i> impiegato per la gestione del codice sorgente secondo una strategia di <i>branching</i> collaborativa per permettere lo sviluppo parallelo.
GitHub	Piattaforma di <i>hosting</i> del <i>repository</i> remoto, utilizzata per supportare la collaborazione tra i membri del gruppo.
Docker	Utilizzato per la standardizzazione e l'isolamento dell'ambiente di esecuzione.

Tabella 6: Strumenti utilizzati per lo sviluppo del progetto

5 Implementazione e prototipo

5.1 Login e autenticazione

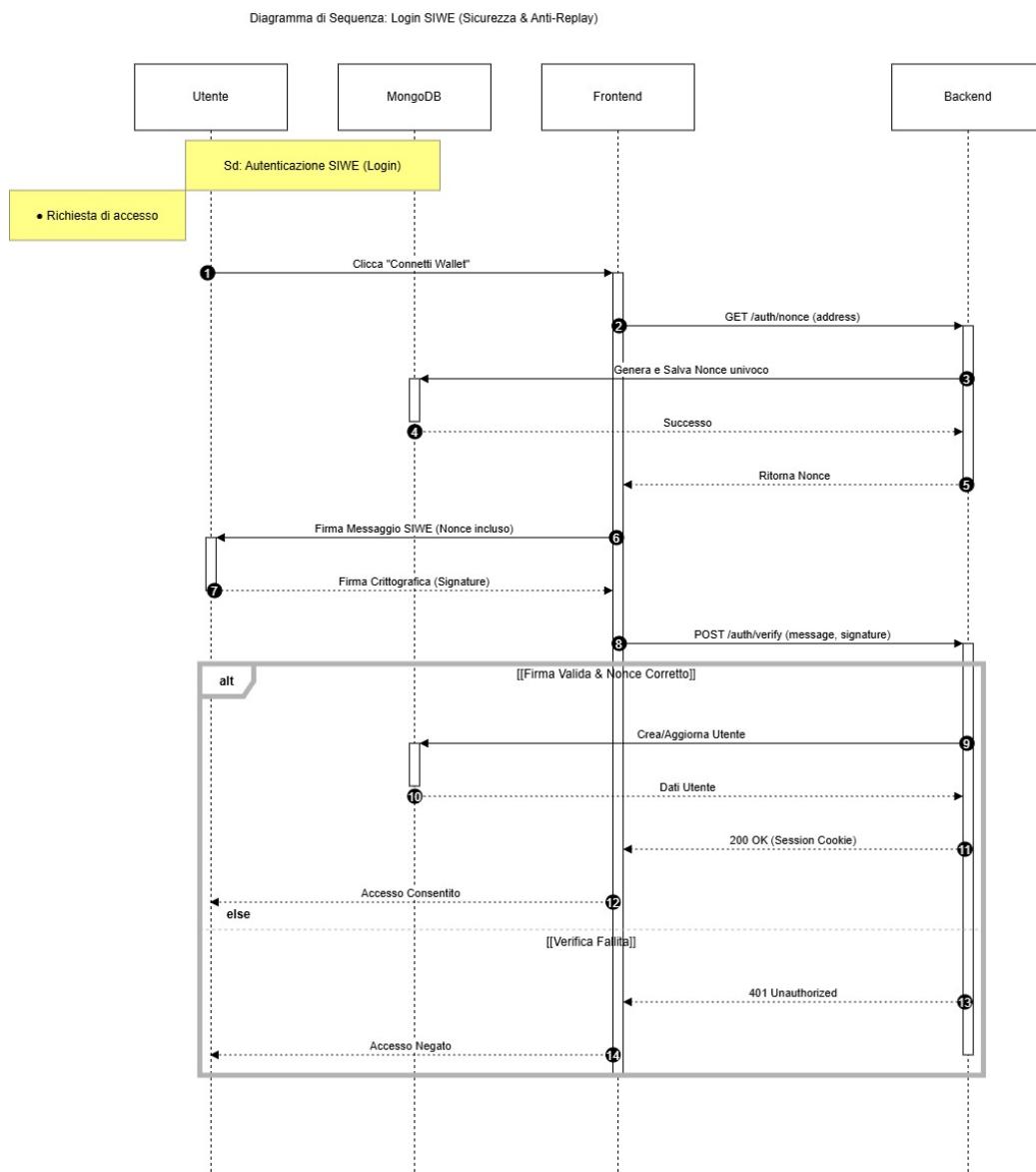
L'autenticazione all'interno della piattaforma adotta il protocollo [Sign-In with Ethereum \(SIWE\)](#) sostituendo il modello tradizionale basato sull'utilizzo di credenziali tradizionali quali *username* e *password*, coerentemente con il paradigma Web3. L'identità dell'utente è infatti associata in modo univoco al possesso di un *wallet* blockchain.

Il flusso di *login* ha inizio con la connessione del *wallet* da parte dell'utente, dalla 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*.

5.2 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.

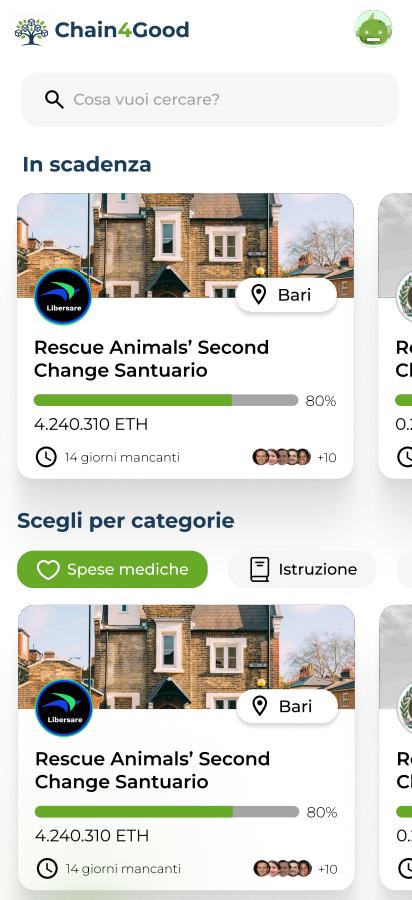


Figura 2: Dashboard del donatore

5.3 Creazione progetto

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

- Step 1: l'Ente è tenuto a specificare le informazioni fondamentali del progetto, quali il nome, la categoria di appartenenza, 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;
- 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.

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) specificando il nome della spesa, 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 effettuata. 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'ultima richiesta precedentemente approvata. Tale approccio mira a garantire un utilizzo progressivo

The figure consists of two side-by-side screenshots of a mobile application. Both screenshots have a header with the 'Chain4Good' logo and a user profile picture.

Left Screenshot (Step 1 di 2):

- Nome del progetto:** Input field.
- Categoria:** Buttons for Spese mediche (selected), Istruzione, Emergenze, Ambiente, and Sport.
- Budget target:** Input field.
- Scadenza:** Input field.
- Buttons at the bottom:** 'Proseguì' (in blue), 'Annulla' (in red), and 'Carica progetto' (in green).

Right Screenshot (Step 2 di 2):

- Descrizione:** Input field.
- Come useremo i fondi:** Input field.
- Immagine di copertina:** Input field.
- Buttons at the bottom:** 'Proseguì' (in blue) and 'Carica progetto' (in green).

Figura 3: Inserimento di un nuovo progetto

e verificabile dei fondi raccolti, e ad assicurare la conformità delle spese per le finalità dichiarate.

5.4.1 Meccanismo di validazione

Per evitare lo stallo decisionale, lo *Smart Contract* è stato programmato per gestire le richieste di spesa attraverso un periodo di votazione di durata prefissata pari a tre giorni, al termine del quale l'esito viene determinato secondo le seguenti regole:

- **Approvazione per maggioranza alla scadenza:** la richiesta è approvata se, al termine del periodo di votazione, i voti favorevoli sono maggiori;
- **Approvazione anticipata per maggioranza matematica:** il sistema prevede la chiusura anticipata della votazione qualora il numero di pareri favorevoli raggiunga una soglia tale da rendere l'esito finale

non più invertibile, anche nell'ipotesi in cui tutti i restanti aventi diritto esprimessero un voto contrario;

- **Parità dei voti:** qualora, allo scadere del periodo di votazione, si verifichi una situazione di parità tra voti favorevoli e contrari, la richiesta viene considerata approvata;
- **Mancata partecipazione:** nel caso in cui non venga espresso alcun voto entro la scadenza, il sistema approva automaticamente la richiesta al fine di non ostacolare l'avanzamento del progetto.

Al soddisfacimento di una delle condizioni di approvazione sopra elencate, lo *Smart Contract* esegue in modo autonomo e irreversibile il trasferimento della somma richiesta verso il *wallet* del beneficiario.

Nuova spesa

Nome spesa

EUR **100,00**

La tua spesa verrà valutata
Prima di sbloccare i fondi,
dovrà essere approvata dai donatori

Descrizione spesa

Allega preventivo

Invia richiesta

RICHIESTA DI SPESA

Acquisto furgoncino fantastico (usato) **1570 USDC**

Grazie al furgone potremmo andare a ricercare i
trovatelli in giro per la città di Bari.

Preventivo-1.pdf

Valuta se è una spesa appropriata

Publicata il 12/12/2025

Ha ricevuto finora 23 voti positivi
Ha ricevuto finora 10 voti negativi

Approva **Nega**

Figura 4: Valutazione di una richiesta di spesa

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 evitare fenomeni di "flash" nel *rendering* delle pagine, migliorando la reattività dell'interfaccia.

L'interazione con l'utente è stata progettata per gestire esplicitamente tutti gli stati possibili dell'applicazione, inclusi caricamenti, errori e assenza di risultati, fornendo sempre messaggi informativi chiari.

Tuttavia, data la natura prototipale della piattaforma, 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 sui contratti intelligenti 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, *export* e *import*, 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.

Nonostante la natura prototipale, la sua implementazione 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. Attraverso l'integrazione della tecnologia blockchain e l'uso di *Smart Contract*, inoltre, è stato possibile ridefinire il concetto stesso donazione, trasformandolo da un mero atto di fiducia in un processo attivo e verificabile.

Tuttavia, nonostante la validità del prototipo realizzato, la sua natura sperimentale apre la strada a 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, correddati 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 proposto potrebbe essere esteso anche ad altri contesti. Questa evoluzione ha il potenziale di democratizzare ulteriormente l'accesso al CF, estendendo i vantaggi della 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. 20150045, 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 (ICAIS)*, 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.