



## 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

|   |           |
|---|-----------|
| <b>Acronimi</b>   | <b>vi</b> |
| <b>1 Introduzione</b>   | <b>1</b>  |
| <b>2 Stato dell'arte</b>  | <b>2</b>  |
| 2.1 Crowdfunding . . . . .  | 2         |
| 2.1.1 Limiti delle piattaforme tradizionali di CF . . . . .       | 3         |
| 2.2 Crowdfunding e blockchain . . . . .                           | 4         |
| 2.2.1 Il ruolo della blockchain nel <i>crowdfunding</i> . . . . . | 4         |
| 2.2.2 Soluzioni esistenti . . . . .                               | 5         |
| <b>3 Metodologia di progetto</b>                                  | <b>7</b>  |
| 3.1 Modello di processo . . . . .                                 | 7         |
| 3.1.1 Organizzazione del team . . . . .                           | 7         |
| 3.2 Pianificazione delle attività . . . . .                       | 8         |
| 3.3 Stima dei costi . . . . .                                     | 8         |
| 3.4 Analisi dei rischi . . . . .                                  | 8         |
| <b>4 Progettazione e implementazione</b>                          | <b>10</b> |
| 4.1 L'obiettivo di Chain4Good . . . . .                           | 10        |
| 4.2 Analisi dei requisiti . . . . .                               | 10        |
| 4.2.1 Requisiti funzionali . . . . .                              | 10        |
| 4.2.2 Requisiti non funzionali . . . . .                          | 12        |
| 4.3 Architettura del Sistema . . . . .                            | 12        |
| 4.4 Stack tecnologico . . . . .                                   | 13        |
| 4.4.1 Front-end . . . . .   | 13        |
| 4.4.2 Back-end . . . . .  | 14        |
| 4.4.3 Blockchain . . . . .  | 15        |
| 4.4.4 Strumenti di sviluppo . . . . .                             | 15        |
| <b>5 Implementazione e prototipo</b>                              | <b>16</b> |
| 5.1 Login e autenticazione . . . . .                              | 16        |
| 5.2 Dashboard donatore . . . . .                                  | 16        |
| 5.3 Creazione progetto . . . . .                                  | 16        |
| 5.4 Inserimento e valutazione spesa . . . . .                     | 17        |
| 5.4.1 Meccanismo di validazione . . . . .                         | 18        |
| <b>6 Validazione e discussione</b>                                | <b>20</b> |
| 6.1 Valutazione dell'applicazione . . . . .                       | 20        |
| 6.2 Realizzazione dei requisiti . . . . .                         | 20        |

|  |           |
|--|-----------|
| <b>7 Conclusioni e sviluppi futuri</b> | <b>21</b> |
| <b>Riferimenti bibliografici</b>       | <b>23</b> |

## **Elenco delle figure**

|   |   |    |
|---|---|----|
| 1 | Dashboard del donatore . . . . .                | 17 |
| 2 | Inserimento di un nuovo progetto . . . . .      | 18 |
| 3 | Valutazione di una richiesta di spesa . . . . . | 19 |

## **Elenco delle tabelle**

|   |  |    |
|---|--|----|
| 1 | Caratteristiche della tecnologia <i>blockchain</i> . . . . .             | 4  |
| 2 | Pianificazione delle attività di progetto. . . . .                       | 9  |
| 3 | Tecnologie e librerie utilizzate per lo sviluppo del front-end.          | 13 |
| 4 | Tecnologie e librerie utilizzate per lo sviluppo del <i>back-end</i> . . | 14 |
| 5 | Tecnologie utilizzate per la componente <i>blockchain</i> . . . . .      | 15 |
| 6 | Strumenti utilizzati per lo sviluppo del progetto . . . . .              | 15 |

## **Acronimi**

**CF** Crowdfunding.

**DCF** Donation-based Crowdfunding.

**ECF** Equity-based Crowdfunding.

**LCF** Lending-based Crowdfunding.

**ONP** Organizzazioni no-profit.

**RBC** Reward-based Crowdfunding.

## 1 Introduzione

## 2 Stato dell'arte

### 2.1 Crowdfunding

Il *Crowdfunding (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 *Kickstarter* e *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];

### 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].

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

### 2.2.1 Il ruolo della blockchain nel *crowdfunding*

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].

| 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*

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.

### 2.2.2 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.

La maggior parte degli studi esaminati mostra un ricorso diffuso a *Smart Contract*<sup>1</sup> per la gestione delle donazioni e a meccanismi di approvazione basati sul consenso dei donatori, per l'approvazione delle richieste di spesa.

Un esempio significativo è il modello proposto in [1], in cui le clausole automatizzate impediscono l'erogazione immediata e integrale dei fondi al beneficiario. Il capitale rimane infatti vincolato nel contratto e viene rilasciato in modo incrementale solo al raggiungimento di obiettivi intermedi stabiliti in fase di creazione del progetto. La peculiarità di questo modello, tuttavia, si esplica nella presenza di un utente *Admin* a cui è delegata la responsabilità di approvare l'inserimento dei progetti in piattaforma e di autorizzare lo sblocco delle quote raccolte.

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.

---

<sup>1</sup>programmi immutabili, registrati direttamente sulla *blockchain* ed eseguiti automaticamente al verificarsi di condizioni prestabilite [1]

Lo *Smart Contract* esegue la transazione direttamente verso il fornitore solo qualora venga raggiunta una maggioranza dei voti superiore al 50% tra i soli sostenitori del progetto.

## 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. La stima della durata, invece, è stata effettuata scomponendo ogni attività in *task* elementari: lo sforzo complessivo stimato è risultato pari a **221 ore**.

Successivamente, al fine di gestire la sequenzialità e il possibile parallelismo tra i *task* individuati, è stato elaborato un Diagramma di Gantt, utilizzato come strumento di supporto alla pianificazione temporale.

Tale approccio ha consentito di organizzare il lavoro in modo strutturato e di definire un riferimento temporale complessivo per l'esecuzione del progetto.

### 3.3 Stima dei costi

La stima dei costi del progetto è stata effettuata sulla base della durata delle attività definite in fase di pianificazione. La valutazione ha tenuto conto della natura prototipale del progetto, della dimensione del sistema, delle tecnologie adottate e del livello di esperienza di ogni membro.

Per fornire una base quantitativa alla stima, si è fatto riferimento al modello algoritmico **Constructive Cost Model (CoCoMo)**, il quale permette di calcolare una stima in termini di tempo/persona.

### 3.4 Analisi dei rischi

| ID  | Descrizione delle attività                              | Durata (ore)      |
|---|---|-------------------|
| <b>Fase 1 – Ideazione e analisi</b>         |   |                   |
| T1  | Brainstorming e definizione dell'idea progettuale       | 6                 |
| T2  | Analisi del problema e del dominio applicativo          | 10                |
| T3  | Analisi dei requisiti e studio della fattibilità        | 12                |
| T4  | Analisi dei rischi e criticità                          | 6                 |
| T5  | Studio delle tecnologie e della <i>blockchain</i>       | 16                |
| <b>Fase 2 – Progettazione</b>               |   |                   |
| T6  | Progettazione dell'architettura del sistema             | 14                |
| T7  | <i>Design</i> delle interfacce utente                   | 20                |
| <b>Fase 3 – Implementazione del sistema</b> |   |                   |
| T8  | Sviluppo del <i>Backend</i>                             | 35                |
| T9  | Sviluppo del <i>Frontend</i>                            | 35                |
| T10   | Redazione della documentazione tecnica                  | 35                |
| <b>Fase 4 – Test e miglioramenti</b>        |   |                   |
| T11   | Integrazione dei componenti                             | 10                |
| T12   | Verifica del prodotto e delle funzionalità implementate | 8                 |
| T13   | Correzione dei bug e perfezionamenti                    | 12                |
| <b>Fase 5 – Conclusione</b>                 |   |                   |
| T14   | Consegna  | 2                 |
|   |   | <b>Totale ore</b> |
|   |   | <b>221</b>        |

Tabella 2: Pianificazione delle attività di progetto.

## 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

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 Stack tecnologico

### 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 impiegata per la gestione della navigazione <i>client-side</i> .  |
| @tanstack/react-query | Libreria utilizzata per la gestione delle chiamate asincrone al <i>backend</i> , con supporto a <i>caching</i> , 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 <i>layout</i> responsivi e coerenti.   |
| Vite                  | Strumento 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'integrazione Web3 e l'interazione con <i>wallet</i> Ethereum e <i>Smart Contract</i> tramite <i>hook React</i> .   |
| viem                  | <i>Client RPC</i> a basso livello, utilizzato internamente per la comunicazione con i nodi <i>blockchain</i> e il recupero dei dati <i>on-chain</i> .  |
| SIWE                  | Protocollo di autenticazione 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 della componente *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        | <a href="#">Object Data Modeling (ODM)</a> 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 degli utenti.  |
| SIWE            | Libreria utilizzata per l'implementazione del protocollo <i>Sign-In with Ethereum</i> , basato sulla verifica di messaggi firmati tramite <i>wallet</i> Ethereum. |
| ethers.js       | Libreria JavaScript utilizzata per l'interazione con la <i>blockchain</i> Ethereum, in particolare per il recupero di informazioni <i>on-chain</i> .              |

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 testing e il deployment 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, come il controllo degli accessi.               |

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

#### 4.4.4 Strumenti di sviluppo

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

| Strumento          | Descrizione  |
|--------------------|--|
| Visual Studio Code | Utilizzato come <a href="#">Ambiente di Sviluppo Integrato (IDE)</a> principale.   |
| Git                | <a href="#">Version Control System (VCS)</a> 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

### 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 il *budget* raggiunto e la 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.

### 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 (2):

- 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 campagna di **CF**;
- 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.

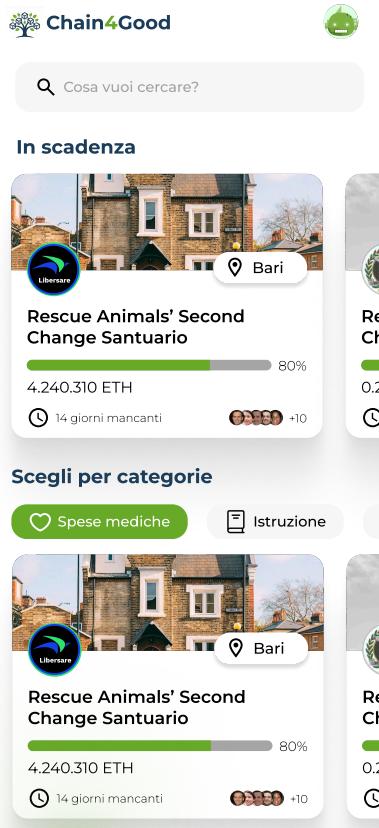


Figura 1: Dashboard del donatore

## 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 formalizzare una "Richiesta di Spesa" (3) attraverso l'inserimento dei seguenti parametri: nome della spesa, importo richiesto, finalità dell'esborso e preventivo.

La sottomissione di una nuova richiesta di spesa, tuttavia, è vincolata al caricamento della prova di acquisto relativa all'ultima richiesta approvata dalla comunità di votanti (insieme degli individui che finanzia un'iniziativa di CF). Tale approccio serve per garantire un utilizzo progressivo e verificabile dei fondi raccolti, e ad assicurare la conformità delle spese per le finalità dichiarate.

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, Istruzione, Emergenze, Ambiente, and Sport.
- Budget target:** Input field.
- Scadenza:** Input field.
- Buttons:** 'Proseguì' (dark blue button), 'Annulla' (red button), and 'Carica progetto' (green button).

**Right Screenshot (Step 2 di 2):**

- Descrizione:** Input field.
- Come useremo i fondi:** Input field.
- Immagine di copertina:** Placeholder input field.

Figura 2: Inserimento di un nuovo progetto

#### 5.4.1 Meccanismo di validazione

Per evitare lo stallo decisionale, lo *Smart Contract* è stato programmato per agire secondo le seguenti regole:

- La richiesta è approvata se la maggioranza dei votanti esprime un parere favorevole.  
In presenza di una partecipazione parziale, la soglia di maggioranza viene ricalcolata in funzione dei soli voti espressi.
- Qualora non venga registrata alcuna attività di voto, invece, il sistema approva automaticamente la richiesta.

Al soddisfacimento dei requisiti di approvazione, lo *Smart Contract* esegue in modo autonomo e irreversibile il trasferimento della somma raccolta verso il *wallet* del beneficiario.

**Nuova spesa**

Nome spesa

**100,00**

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

Descrizione spesa

Allega preventivo

**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**

Approva    Nega

**Invia richiesta**

Figura 3: Valutazione di una richiesta di spesa

## 6 Validazione e discussione

### 6.1 Valutazione dell'applicazione

Gli obiettivi di progetto sono stati raggiunti con successo, l'applicazione web decentralizzata realizzata infatti soddisfare tutti i requisiti definiti in fase di analisi. La sua progettazione e realizzazione hanno richiesto lo studio di diverse tecnologie innovative e un'analisi attenta per capire come integrarle tra loro per riuscire a concretizzare il risultato atteso.

### 6.2 Realizzazione dei requisiti

## 7 Conclusioni e sviluppi futuri



**Angelica De Feudis**



**Jonathan Caputo**



**Luca Gentile**

## 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.