



A.D. 1308  
**unipg**  
DIPARTIMENTO  
DI INGEGNERIA

Tesina Finale di  
**Data Security and Blockchain**  
Corso di Laurea Magistrale in Ingegneria Informatica e Robotica – A.A. 2024-2025  
DIPARTIMENTO DI INGEGNERIA

docente  
Prof. Luca GRILLI

# EthFunder

Web DApp frontend e backend eseguita sulla Testnet Sepolia Ethereum  
tecnologie: HTML/CSS/JavaScript/Web3.js/Node.js/Express.js



studente

363433 **Gian Marco Ferri** gianmarco.ferri@studenti.unipg.it

# 0. Indice

<b>1</b>	<b>Descrizione del Problema</b>	<b>2</b>
1.1	Introduzione alla DApp EthFunder . . . . .	2
<b>2</b>	<b>Specifica dei Requisiti</b>	<b>4</b>
<b>3</b>	<b>Progetto</b>	<b>6</b>
3.1	Architettura del Sistema Software . . . . .	6
3.2	Smart Contract . . . . .	7
3.2.1	Struttura dello smart contract . . . . .	8
<b>4</b>	<b>Utilizzo del sistema</b>	<b>13</b>
<b>5</b>	<b>Conclusioni e sviluppi futuri</b>	<b>20</b>
<b>6</b>	<b>Bibliografia</b>	<b>21</b>

# 1. Descrizione del Problema

L'obiettivo di questo progetto è quello di sviluppare un'applicazione web decentralizzata (*DApp*) chiamata **EthFunder**, progettata per gestire efficacemente le campagne di raccolta fondi utilizzando la tecnologia blockchain. La piattaforma sfrutta le potenzialità offerte dagli smart contract di Ethereum per fornire una soluzione sicura e trasparente per gli utenti per creare, gestire e contribuire alle campagne di raccolta fondi.

EthFunder è una Web DApp che opera direttamente nel browser degli utenti, supportata da un server Node.js come backend. L'interfaccia grafica dell'applicazione è sviluppata utilizzando le tecnologie web standard (HTML, CSS e JavaScript), mentre l'interazione con la blockchain è gestita attraverso la libreria Web3.js [6]. Lo smart contract della DApp, scritto in Solidity [4], sarà eseguito nella testnet pubblica Sepolia di Ethereum [5].

## 1.1 Introduzione alla DApp EthFunder

EthFunder semplifica l'accesso alla tecnologia blockchain attraverso un'interfaccia intuitiva. La piattaforma permette di interagire con la blockchain attraverso semplici moduli HTML, rendendo accessibili operazioni come la creazione di campagne e il trasferimento di fondi anche a chi non possiede competenze tecniche specifiche.

La sicurezza è garantita dalle caratteristiche intrinseche della tecnologia blockchain: l'immutabilità e la decentralizzazione assicurano che ogni transazione sia protetta e verificabile. Questo permette agli utenti di contribuire alle campagne con la certezza che i loro fondi saranno gestiti in modo trasparente e responsabile.

La struttura portante di EthFunder è uno smart contract implementato sulla blockchain di Ethereum, che automatizza e regola ogni aspetto delle campagne di

raccolta fondi. Questo contratto definisce regole chiare per l'invio e la ricezione dei fondi, eliminando la necessità di supervisione manuale e riducendo drasticamente il rischio di frodi. Le transazioni vengono elaborate automaticamente quando vengono soddisfatte le condizioni prestabilite, dimostrando come la blockchain possa modernizzare e rendere più efficienti i processi tradizionali.

EthFunder è un esempio concreto della potenzialità della tecnologia blockchain. Semplici applicazioni come questa aprono la strada a soluzioni più sofisticate in diversi settori, dimostrando le svariate possibilità di applicazione della tecnologia blockchain.

## 2. Specifica dei Requisiti

Nel seguente capitolo, vengono elencati i requisiti che l'applicazione EthFunder dovrà soddisfare.

1. Autenticazione utente:
  - Gli utenti devono connettere il loro portafoglio MetaMask [2] per interagire con la DApp
  - Gli utenti possono connettere e disconnettere il proprio portafoglio
2. Gli utenti possono creare una nuova raccolta fondi fornendo:
  - Titolo del progetto
  - Descrizione del progetto
  - Obiettivo di finanziamento (in wei [1])
  - Durata (in giorni)
3. Gli utenti possono contribuire ad un fondo esistente specificando:
  - Identificativo del progetto
  - Importo del contributo (in wei)
4. Gli utenti possono visualizzare i progetti attivi con dettagli quali:
  - Identificativo del progetto
  - Titolo del progetto
  - Descrizione del progetto
  - Obiettivo di finanziamento (in wei)
  - Scadenza

- Stato (attivo, completato)

5. Prelievo fondi:

- I proprietari del progetto possono prelevare fondi dal loro progetto se l'obiettivo di finanziamento è stato raggiunto
- Solo il proprietario del progetto può prelevare i fondi

6. Rimborsi:

- I contributori possono richiedere un rimborso della loro quota se un progetto non ha raggiunto il suo obiettivo di finanziamento entro la data di scadenza
- I rimborsi sono disponibili solo per i progetti che non sono stati completati entro la data di scadenza specificata da chi ha creato il fondo

7. Avvisi per varie azioni come:

- Connessione riuscita a MetaMask
- Creazione di un progetto riuscita
- Contribuzione ad un progetto riuscita
- Prelievo riuscito
- Rimborso riuscito
- Messaggi informativi per ogni azione riuscita
- Messaggi di errore per ogni operazione non riuscita

8. Interfaccia ed esperienza utente (frontend):

- Intuitiva e minimale, al fine di agevolare l'utilizzo del sistema
- Gli utenti ricevono feedback in tempo reale sulle loro azioni ed eventuali errori tramite avvisi
- Compatibile con i browser web più recenti

9. Server (backend):

- La DApp utilizza un server implementato tramite Node.js
- Il server è in ascolto sulla porta 3000.

## 3. Progetto

La DApp EthFunder è un'applicazione decentralizzata progettata per facilitare il *crowdfunding* sulla blockchain di Ethereum. Questa piattaforma consente agli utenti di creare e supportare vari progetti tramite meccanismi di finanziamento sicuri. Lo smart contract personalizzato alla base della DApp assicura che tutte le transazioni e le varie interazioni siano immutabili e affidabili, fornendo un elevato livello di sicurezza e trasparenza per tutti gli utenti.

Nei prossimi paragrafi verrà descritta l'architettura e i principali componenti di EthFunder.

### 3.1 Architettura del Sistema Software

EthFunder è una Web DApp che opera sulla Testnet Sepolia di Ethereum, la cui architettura integra un frontend per gestire le interazioni degli utenti e un backend che processa le richieste.

L'interfaccia utente è stata realizzata utilizzando le principali tecnologie web: la struttura è costruita in HTML, lo stile e il design sono definiti tramite CSS, mentre JavaScript gestisce le interazioni e il comportamento dinamico dell'applicazione.

Il backend è sviluppato con Node.js ed Express, un framework leggero che gestisce le richieste HTTP e serve i file statici dell'applicazione. Il server si occupa di instradare le richieste in arrivo e fornire le risorse necessarie al client, operando sulla porta 3000.

L'applicazione interagisce con la blockchain di Ethereum sulla rete di test Sepolia attraverso la libreria Web3.js, che fornisce un'interfaccia per comunicare con

lo smart contract. Quest'ultimo, scritto in Solidity, gestisce in modo sicuro e trasparente tutte le operazioni fatte sulla blockchain.

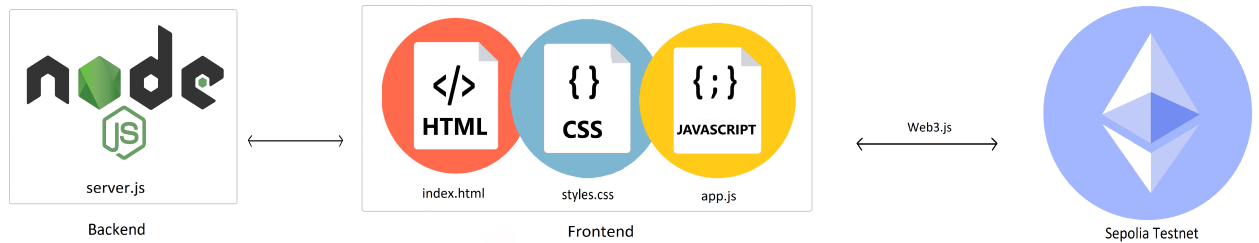


Figura 3.1: Architettura della DApp EthFunder

La Figura 3.1 illustra l'architettura di EthFunder, evidenziando i componenti chiave dell'applicazione:

- `index.html`: punto d'accesso all'applicazione che contiene la struttura di base della pagina web;
- `styles.css`: definisce gli stili e il layout per i componenti della pagina web;
- `app.js`: responsabile della logica lato client. Include funzioni per la connessione a MetaMask, la creazione di progetti, il contributo a progetti, il prelievo di fondi e la richiesta di rimborsi. Gestisce inoltre la visualizzazione di avvisi, la gestione di sessioni utente (login/logout) e l'aggiornamento dinamico dell'interfaccia in base alle interazioni utente e ai dati recuperati dalla blockchain;
- `server.js`: sviluppato con Node.js ed Express.

La discussione dettagliata dello smart contract e delle sue funzionalità verrà affrontata nella prossima sezione.

## 3.2 Smart Contract

Lo smart contract `Crowdfunding.sol` è il componente principale della DApp EthFunder. È scritto in *Solidity* e distribuito sulla testnet Sepolia della blockchain di Ethereum. Per ottenere il bytecode e l'ABI dello smart contract è stato utilizzato l'IDE online Remix [3].

`Crowdfunding.sol` gestisce tutta la logica relativa alla creazione di progetti, al contributo di fondi, alla gestione dei prelievi e all'emissione di rimborsi.



Di seguito è riportata una discussione dettagliata dello smart contract e delle sue funzioni chiave.

### 3.2.1 Struttura dello smart contract

Lo smart contract `Crowdfunding.sol` è strutturato come segue:

#### Variabili di stato

- `address public platformOwner`: Memorizza l'indirizzo del proprietario della piattaforma che fa il *deploy* del contratto.
- `uint public projectCount`: Tiene traccia del numero totale di progetti creati sulla piattaforma.
- `struct Project`: Definisce la struttura di un progetto, incluso l'indirizzo del proprietario, il titolo, la descrizione, l'obiettivo di finanziamento, la scadenza, i fondi raccolti e lo stato di completamento.
- `mapping(uint => Project) public projects`: Associa gli ID dei progetti alle strutture dei progetti corrispondenti.
- `mapping(uint => mapping(address => uint)) public contributions`: Tiene traccia dei contributi forniti da ciascun indirizzo a ciascun progetto.

#### Eventi

- `event ProjectCreated(uint projectId, string title, uint goal, uint deadline)`: Emesso quando viene creato un nuovo progetto.
- `event ContributionReceived(uint projectId, address contributor, uint amount)`: Emesso quando si riceve un contributo per un progetto.
- `event ProjectCompleted(uint projectId)`: Emesso quando un progetto raggiunge il suo obiettivo di finanziamento.
- `event RefundIssued(uint projectId, address contributor, uint amount)`: Emesso quando viene effettuato un rimborso a un collaboratore.

#### Funzioni chiave

**constructor** Inizializza lo smart contract impostando il proprietario della piattaforma sull'indirizzo che ha distribuito il contratto.

```

constructor() {
    platformOwner = msg.sender;
}

```

**createProject** Consente agli utenti di creare un nuovo progetto di crowdfunding fornendo un titolo, una descrizione, un obiettivo di finanziamento e una durata. Incrementa il `projectCount` e memorizza il nuovo progetto nella lista dei progetti.

```

function createProject(
    string memory _title,
    string memory _description,
    uint _goal,
    uint _durationInDays
) public returns (uint) {
    require(_goal > 0, "Goal must be greater than zero");
    require(_durationInDays > 0, "Duration must be
        greater than zero");

    projectCount++;
    projects[projectCount] = Project(
        msg.sender,
        _title,
        _description,
        _goal,
        block.timestamp + (_durationInDays * 1 days),
        0,
        false
    );

    emit ProjectCreated(projectCount, _title, _goal,
        projects[projectCount].deadline);
    return projectCount;
}

```

**contribute** Consente agli utenti di contribuire con fondi a un progetto attivo. La funzione verifica che il progetto sia ancora attivo e che l'importo del contributo sia maggiore di zero. Aggiorna i fondi raccolti dal progetto e registra il contributo nella mappatura dei contributi.

```

function contribute(uint _projectId) public payable
    activeProject(_projectId) {

```

```

    require(msg.value > 0, "Contribution must be greater
        than zero");

    Project storage project = projects[_projectId];
    require(project.fundsRaised < project.goal, "Funding
        goal already met");

    project.fundsRaised += msg.value;
    contributions[_projectId][msg.sender] += msg.value;

    emit ContributionReceived(_projectId, msg.sender,
        msg.value);

    if (project.fundsRaised >= project.goal) {
        project.isCompleted = true;
        emit ProjectCompleted(_projectId);
    }
}

```

**withdrawFunds** Consente al proprietario del progetto di prelevare fondi se il progetto ha raggiunto il suo obiettivo di finanziamento. La funzione trasferisce i fondi raccolti all'indirizzo del proprietario del progetto e imposta `fundsRaised` a zero.

```

function withdrawFunds(uint _projectId) public {
    Project storage project = projects[_projectId];
    require(msg.sender == project.owner, "Only project
        owner can withdraw");
    require(project.isCompleted, "Project funding goal
        not met yet");

    uint amount = project.fundsRaised;
    project.fundsRaised = 0;

    // Transfer funds to the project owner
    (bool success, ) = project.owner.call{value:
        amount}("");
    require(success, "Transfer failed");
}

```

**refund** Consente ai contributori di richiedere un rimborso se il progetto non riesce a raggiungere il suo obiettivo di finanziamento entro la scadenza. La funzione

trasferisce l'importo versato al contribuente e imposta il suo contributo su zero.

```
function refund(uint _projectId) public {
    Project storage project = projects[_projectId];
    require(block.timestamp > project.deadline, "Project
    is still active");
    require(!project.isCompleted, "Project funding goal
    met");

    uint amount = contributions[_projectId][msg.sender];
    require(amount > 0, "No contributions to refund");

    contributions[_projectId][msg.sender] = 0;

    // Transfer funds back to the contributor
    (bool success, ) = msg.sender.call{value:
        amount}("");
    require(success, "Refund failed");

    // By setting contributions[_projectId][msg.sender]
    to 0 before transferring funds, we minimize the
    risk of reentrancy.
    emit RefundIssued(_projectId, msg.sender, amount);
}
```

**getProject** Recupera i dettagli di un progetto specifico tramite il suo ID. Questa funzione restituisce vari attributi del progetto, tra cui proprietario, titolo, descrizione, obiettivo, scadenza, fondi raccolti e stato di completamento.

```
function getProject(uint _projectId) public view returns (
    address, string memory, string memory, uint, uint, uint,
    bool
) {
    Project storage project = projects[_projectId];
    return (
        project.owner,
        project.title,
        project.description,
        project.goal,
        project.deadline,
        project.fundsRaised,
        project.isCompleted
    );
}
```

**getActiveProjects** Restituisce un elenco di ID per tutti i progetti attivi. Questa funzione scorre tutti i progetti e raccoglie gli ID di quelli che sono ancora attivi.

```
function getActiveProjects() public view returns (uint[]
memory) {
    uint[] memory activeProjects = new uint[](projectCount);
    uint counter = 0;

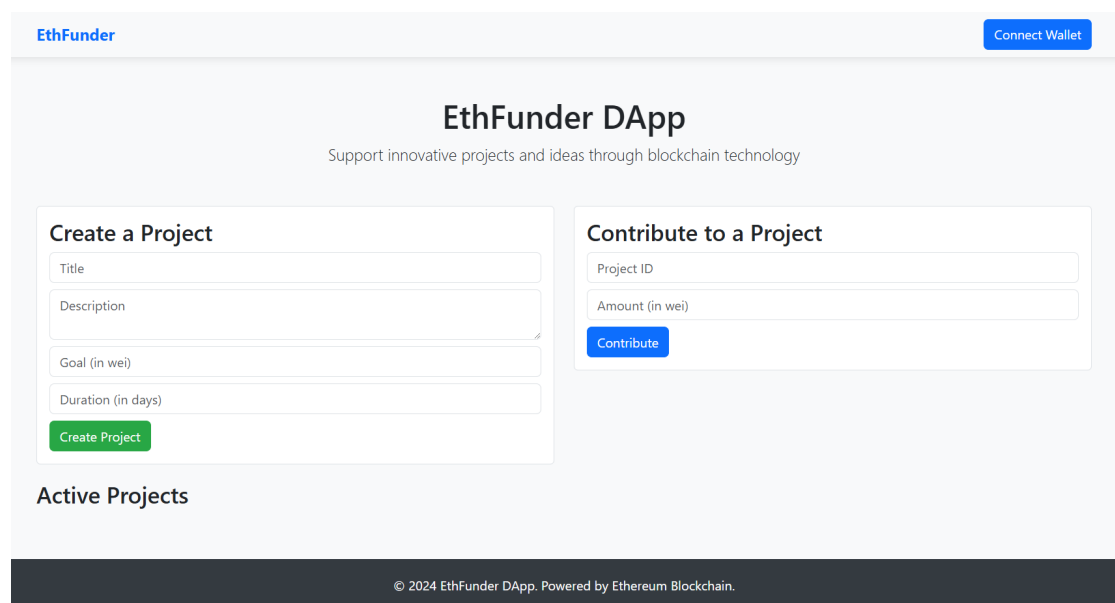
    for (uint i = 1; i <= projectCount; i++) {
        if (!projects[i].isCompleted && block.timestamp <
            projects[i].deadline) {
            activeProjects[counter] = i;
            counter++;
        }
    }

    return activeProjects;
}
```

## 4. Utilizzo del sistema

In questa sezione viene mostrata l'interfaccia di EthFunder e viene illustrato come interagire con essa.

### Home Page



The screenshot displays the home page of the EthFunder DApp. At the top left is the 'EthFunder' logo, and at the top right is a 'Connect Wallet' button. The main heading is 'EthFunder DApp' with the tagline 'Support innovative projects and ideas through blockchain technology'. Below this, there are two primary action boxes. The left box, titled 'Create a Project', contains input fields for 'Title', 'Description', 'Goal (in wei)', and 'Duration (in days)', followed by a green 'Create Project' button. The right box, titled 'Contribute to a Project', contains input fields for 'Project ID' and 'Amount (in wei)', followed by a blue 'Contribute' button. Below these boxes is a section labeled 'Active Projects'. At the bottom of the page, a dark footer bar contains the text '© 2024 EthFunder DApp. Powered by Ethereum Blockchain.'

Figura 4.1: Home Page della DApp EthFunder

In Figura 4.1 è raffigurata la Home Page della piattaforma visualizzata dall'utente all'avvio dell'applicazione prima del login.

## Connessione con Metamask

All'avvio dell'applicazione MetaMask chiederà all'utente di connettere il proprio *wallet*.

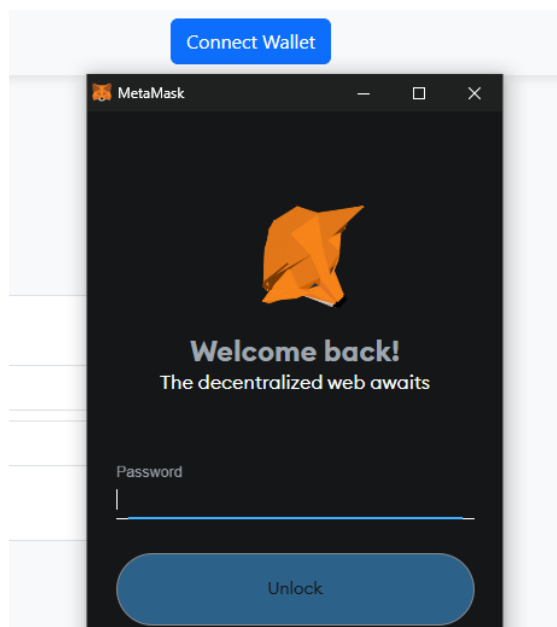


Figura 4.2: Richiesta connessione wallet Metamask

Approvando la richiesta, verrà visualizzato l'indirizzo del portafoglio connesso, preceduto dalla scritta "Connected" su un pulsante in alto a destra nella pagina.

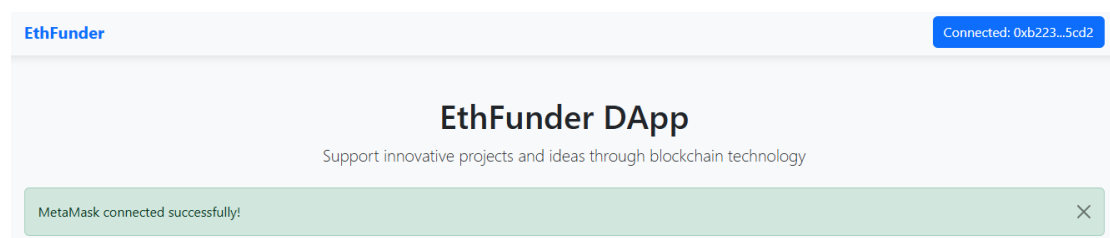


Figura 4.3: Login effettuato

Cliccando su tale pulsante, l'utente potrà disconnettere il proprio portafoglio, e la scritta cambierà in "Connect wallet".

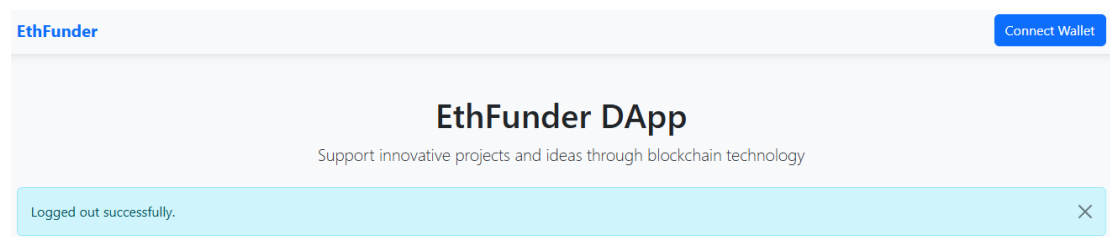


Figura 4.4: Logout effettuato

Come si può vedere nella Figura 4.3 e nella Figura 4.4, entrambe le operazioni sono accompagnate da un messaggio informativo che conferma l'avvenuto login o logout.

### Creazione di un progetto

L'utente può creare una raccolta fondi semplicemente compilando il *form* mostrato in Figura 4.5.

The image shows a 'Create a Project' form. It has a title 'Create a Project' at the top. Below the title are four input fields: 'Title', 'Description', 'Goal (in wei)', and 'Duration (in days)'. The 'Title' field is highlighted with a blue border. At the bottom of the form is a green button labeled 'Create Project'.

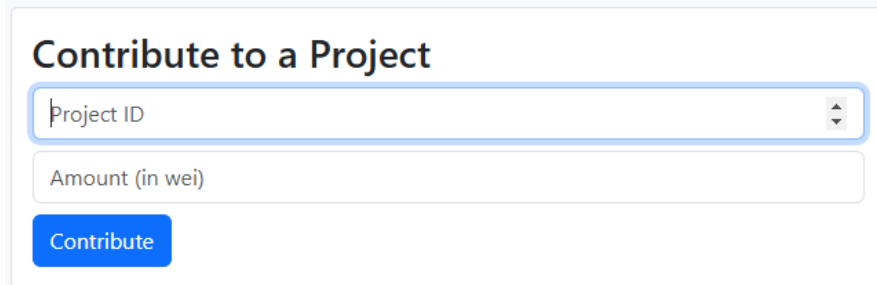
Figura 4.5: Creazione di un fondo

Cliccando sul pulsante "Create Project" verrà creata una raccolta fondi con i parametri specificati dall'utente, previa autorizzazione della transazione da parte di Metamask.



## Contribuire ad un progetto

L'utente può contribuire ad una raccolta fondi con una quota a sua scelta semplicemente compilando il *form* mostrato in Figura 4.6.



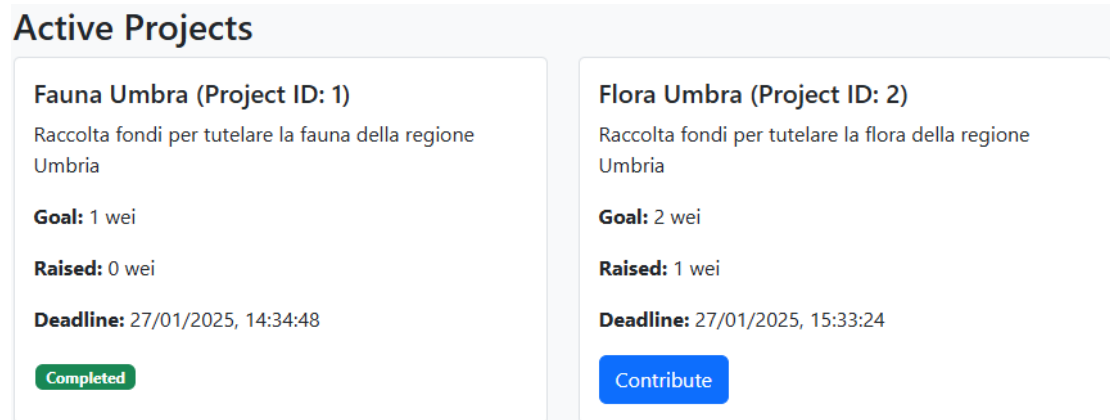
The form is titled "Contribute to a Project". It contains two input fields: "Project ID" and "Amount (in wei)". Below these fields is a blue button labeled "Contribute".

Figura 4.6: Contribuire ad un fondo

Cliccando sul pulsante "Contribute" verrà inviata la quota scelta alla raccolta fondi specificata dall'utente, previa autorizzazione della transazione da parte di Metamask.

## Progetti attivi

Dopo aver fatto il login, l'utente potrà vedere una lista di raccolte fondi create sia da sé stesso che da altri utenti, con le relative informazioni.



The section is titled "Active Projects". It displays two project cards. The first card is for "Fauna Umbra (Project ID: 1)" with a goal of 1 wei, 0 wei raised, and a deadline of 27/01/2025, 14:34:48. It has a green "Completed" button. The second card is for "Flora Umbra (Project ID: 2)" with a goal of 2 wei, 1 wei raised, and a deadline of 27/01/2025, 15:33:24. It has a blue "Contribute" button.

Figura 4.7: Lista dei progetti attivi

Cliccando sul pulsante "Contribute" di un progetto, il suo Project ID verrà automaticamente inserito nel relativo campo del form "Contribute to a project" e verrà evidenziato il campo corrispondente alla quota di finanziamento, come mostrato nella Figura 4.8.

The image shows a web interface for contributing to a project. On the left, there is a vertical list of project cards. On the right, a larger form titled 'Contribute to a Project' is displayed. This form has a text input field containing the number '2', a label 'Amount (in wei)', and a blue 'Contribute' button. Below the form, a project card for 'Flora Umbra (Project ID: 2)' is shown. The card text includes: 'Raccolta fondi per tutelare la flora della regione Umbria', 'Goal: 2 wei', 'Raised: 1 wei', and 'Deadline: 27/01/2025, 15:33:24'. A blue 'Contribute' button is at the bottom of the card.

Figura 4.8: Contribuire ad un progetto dalla lista

I progetti completati verranno etichettati come "Completed", mentre quelli che non hanno raggiunto l'obiettivo di finanziamento entro la data di scadenza verranno etichettati come "Expired".

The image shows a section titled 'Active Projects'. It contains two project cards. The left card is for 'Fauna Umbra (Project ID: 1)' with the description 'Raccolta fondi per tutelare la fauna della regione Umbria'. It shows 'Goal: 1 wei', 'Raised: 1 wei', and 'Deadline: 27/01/2025, 14:34:48'. At the bottom, there is a green button labeled 'Completed'. The right card is for 'Flora Umbra (Project ID: 2)' with the description 'Raccolta fondi per tutelare la flora della regione Umbria'. It shows 'Goal: 2 wei', 'Raised: 1 wei', and 'Deadline: 27/01/2025, 15:33:24'. At the bottom, there is a red button labeled 'Expired'.

Figura 4.9: Label relative allo stato dei progetti

## Withdraw

Se una raccolta fondi dovesse raggiungere il suo obiettivo di finanziamento prima della scadenza, verrà etichettata come completata e il proprietario potrà recuperare i fondi donati cliccando sul pulsante "Withdraw" che apparirà appositamente.

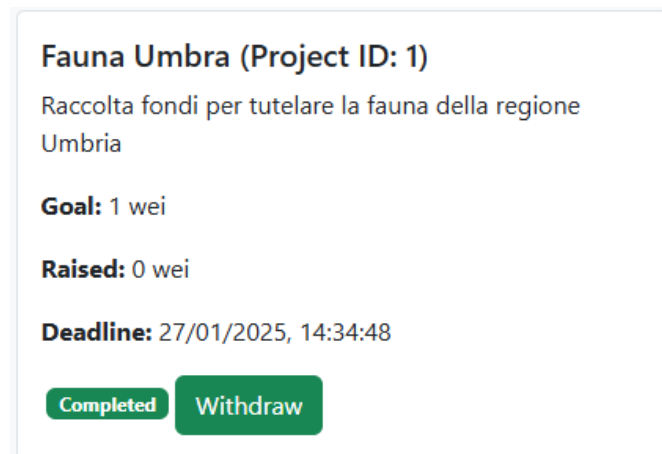


Figura 4.10: Raccolta fondi completata con successo

## Refund

Se una raccolta fondi non dovesse raggiungere il suo obiettivo di finanziamento prima della scadenza, coloro che avevano finanziato il progetto potranno recuperare la propria quota cliccando sul pulsante "Refund" che apparirà appositamente.

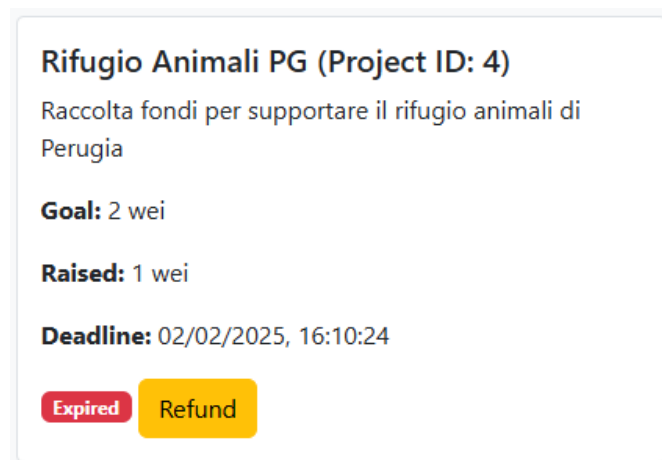


Figura 4.11: Raccolta fondi non completata con successo

## Avvisi per errori, conferme e altri messaggi

Come già mostrato nella Figura 4.3 e nella Figura 4.4, ogni operazione avvenuta con successo è accompagnata da messaggi di conferma e, in caso di errore, da

messaggi esplicativi che guidano l'utente nella risoluzione del problema, come si può vedere in Figura 4.12.



Figura 4.12: Esempi di messaggi di errore

Ogni interazione con la blockchain richiede alcuni secondi di elaborazione. Durante questo periodo, verrà visualizzata la scritta "Loading..." che scomparirà automaticamente al completamento dell'operazione, quando la lista dei progetti sarà aggiornata.

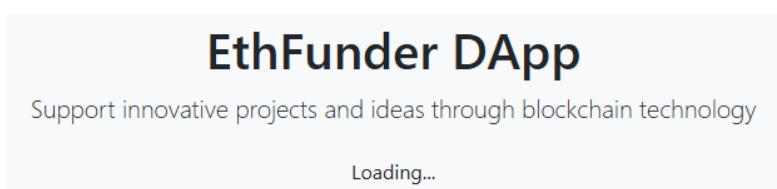


Figura 4.13: Indicatore di caricamento

## 5. Conclusioni e sviluppi futuri

La DApp EthFunder è un'applicazione decentralizzata progettata per facilitare il crowdfunding sulla blockchain di Ethereum. Consente agli utenti di creare e supportare vari progetti tramite un meccanismo di finanziamento sicuro e trasparente. La piattaforma utilizza uno smart contract che si appoggia sulla blockchain di Ethereum per garantire che tutte le transazioni siano immutabili e affidabili, fornendo un elevato livello di sicurezza e trasparenza per tutti gli utenti coinvolti.

Per quanto EthFunder soddisfi appieno i requisiti attuali, ci sono sicuramente ancora possibilità di miglioramento ed estensione per renderla ancora più versatile e adattabile alle esigenze degli utenti. A tal fine si possono prendere in considerazione diversi sviluppi futuri, come ad esempio l'introduzione di funzionalità di progetto avanzate. Si potrebbero infatti introdurre obiettivi intermedi con rilasci di finanziamento gradualmente, cosa che consentirebbe ai creatori del progetto di ricevere fondi in più fasi, subordinatamente al raggiungimento di traguardi specifici.

Un altro possibile sviluppo potrebbe riguardare l'integrazione della DApp con altre blockchain, cosa che aumenterebbe l'accessibilità della piattaforma e la base di utenti.

## 6. Bibliografia

- [1] Binance Academy. Wei — Glossary, 2025. [Online <https://academy.binance.com/en/glossary/wei>; controllata il 30-gennaio-2025].
- [2] Consensys. MetaMask: Crypto Wallet & Gateway to Blockchain Apps, 2025. [Online <https://metamask.io/it/>; controllata il 30-gennaio-2025].
- [3] Ethereum Foundation. Remix - Ethereum IDE & Community, 2025. [Online <https://remix-project.org/>; controllata il 30-gennaio-2025].
- [4] Ethereum Foundation. Solidity Programming Language, 2025. [Online <https://soliditylang.org/>; controllata il 30-gennaio-2025].
- [5] sepolia.etherscan.io. Testnet Sepolia (ETH) Blockchain Explorer, 2025. [Online <https://sepolia.etherscan.io/>; controllata il 30-gennaio-2025].
- [6] ChainSafe Systems. Web3.js — Javascript Ethereum API, 2025. [Online <https://web3js.org/>; controllata il 30-gennaio-2025].