Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Sviluppo e integrazione di funzionalità per una piattaforma per comunità energetiche basata su blockchain

Mario Squillace

SoftStrategy Drive the Value

RdS/PTR(2021)/045

Sviluppo e integrazione di funzionalità per una piattaforma per comunità energetiche basata su blockchain

Mario Squillace, Emilio Alfieri

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero della Transizione Ecologica- ENEA
Piano Triennale di Realizzazione 2019-2021 - III annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: LA1.50 Test e sperimentazione dell'infrastruttura e dei servizi per le local energy communities
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno del Contratto "Sviluppo di ulteriori funzionalità per servizi di comunità energetiche"
Responsabile Unico del Procedimento ENEA: Gianluca D'Agosta
Responsabile del Contratto per il Contraente Soft Strategy SpA: Roberto Provenzano

Indice

Indice Figure

Sommario

Il documento descrive le attività svolte per la realizzazione di ulteriori funzionalità della piattaforma per comunità energetiche basata su blockchain.

Le funzionalità aggiuntive sono a disposizione di tutta la comunità energetica che è aperta alle tipologie di partecipanti privati, condomini, aziende o enti pubblici che siano interessati a migliorare il proprio profilo energetico sia dal punto di vista economico, con la riduzione dei consumi o con forme differenti di generazione, sia ambientale.

La soluzione realizzata permette agli utenti delle comunità energetiche di gestire i token presenti nel proprio "wallet" in modo semplice, intuitivo e trasparente, permettendone quindi lo scambio all'interno della comunità per usufruire di determinati servizi messi a disposizione e di acquistare attraverso il marketplace tutti i servizi messi a disposizione dalla comunità energetica.

1 Introduzione

L'obiettivo di questa attività è quello di realizzare, sulla base di una piattaforma già esistente, un sistema e relativi sotto-moduli in grado di:

- gestire uno o più token di Comunità (separando le differenti Comunità che afferiranno alla medesima piattaforma), seguendo delle specifiche politiche di comunità che sono state definite.
- Implementare in maniera trasparente e sicura dei modelli e delle logiche di assegnazione, perdita e scambio di token fra gli utenti e la piattaforma, legate sia ai servizi elettrici (consumo, produzione ma anche messa a disposizione di flessibilità energetica) che sociali (fruizione o fornitura dei servizi);
- di controllare e di modificare, attraverso dei parametri, le stesse logiche all'interno dei singoli smart contract da parte di un operatore (non per forza esperto programmatore), in base alle necessità di modifica del modello della comunità;
- essere connesso ad una o più Dapp per la gestione/visualizzazione delle transazioni e mostrare le informazioni relative attraverso delle interfacce grafiche e dashboard;
- essere connesso a sorgenti dati interne, come i database relazionali e di altro genere, che agiscono come applicazioni "oracoli" ad uso della blockchain stessa e degli smart contract in essa sviluppati.

Gli smart contract implementano le logiche e i modelli di scambio dei token, consentendo, al gestore stesso della comunità, di settare alcuni parametri interni agli smart contract, in modo da modificare il risultato della transazione stessa in funzione delle proprie necessità.

Tutti gli utenti che si registrano alla comunità dispongono di un portafogli ("wallet") che mantiene il bilancio dei token in possesso. E stata realizzata una Dapp che permette agli utenti di interagire con la piattaforma, avendo accesso al proprio saldo in token e permettendo il "pagamento" o la "riscossione" di token.

Rispetto alle applicazioni già sviluppate, l'obiettivo è quello di rendere "semplice" l'utilizzo dei token, ad esempio attraverso la lettura di un QR-Code installato presso un esercizio commerciale che partecipi all'iniziativa e che, in cambio di token, dia accesso ad uno sconto o a dei specifici servizi.

2 Descrizione delle attività svolte e risultati

La soluzione proposta consente di ampliare l'attuale piattaforma software con ulteriori funzionalità e perfezionamenti volte ad un migliore funzionamento della comunità energetica, soprattutto in termini di quantità di servizi e loro gestione. Di seguito viene mostrato il nuovo schema logico architetturale con evidenziate in arancione le parti che sono state modificate/aggiunte rispetto all'architettura precedente.

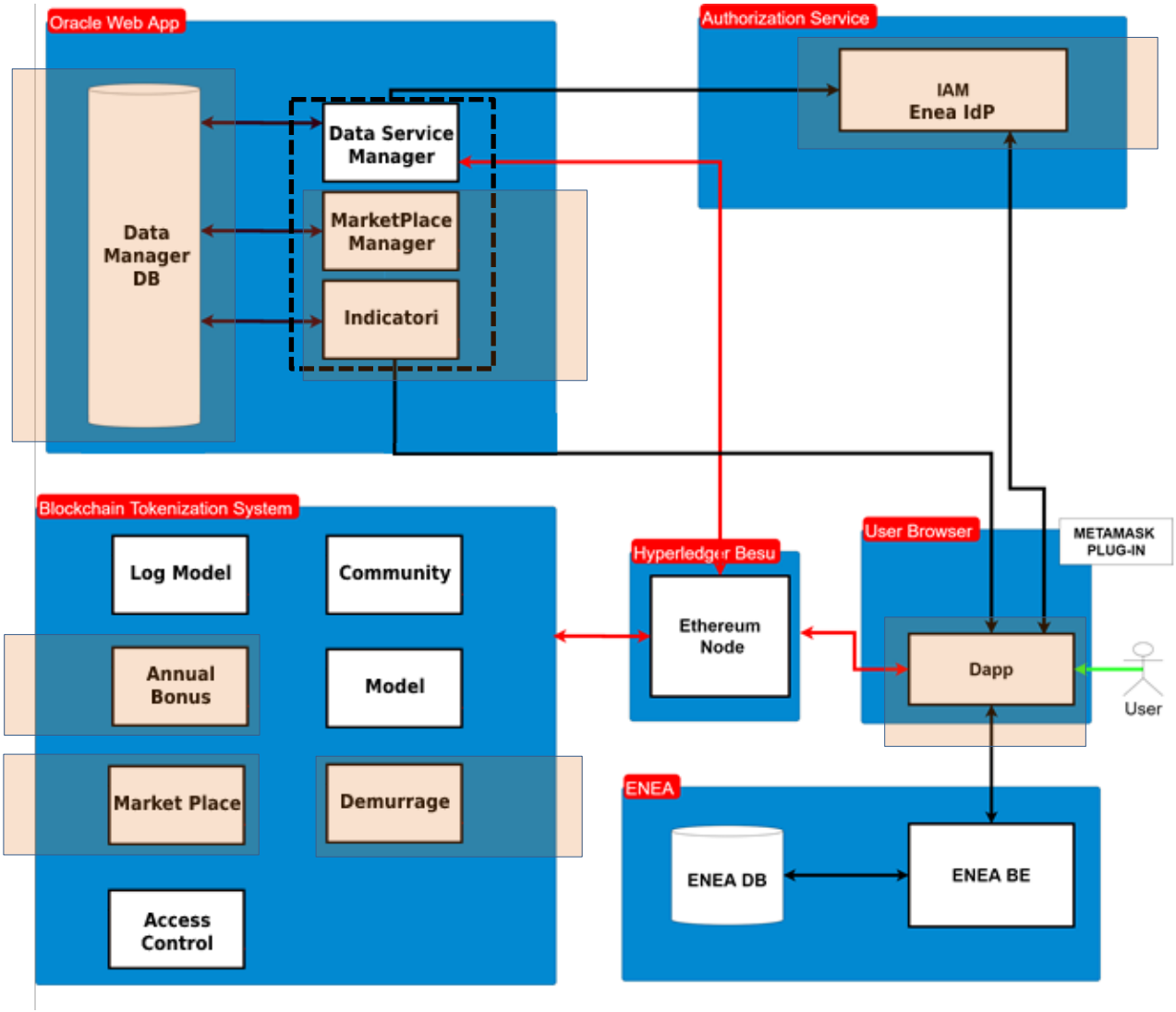

Figura 1 - Nuovo schema logico architetturale

2.1 Integrazione con il nuovo Identity Provider di Enea

Tutte le componenti di autenticazione del Back End e Front End sono state integrate e configurate con il nuovo Identity Provider fornito da ENEA che sostituisce l'IdP precedente basato sul software open source KeyCloak.

Lo schema di autenticazione di ENEA sfrutta un gestore dell'identità dell'utente, un Identity Provider (IdP), al quale il Font End richiede la verifica dell'identità.

Di seguito viene mostrato uno schema logico dell'autenticazione verso il nuovo IdP.

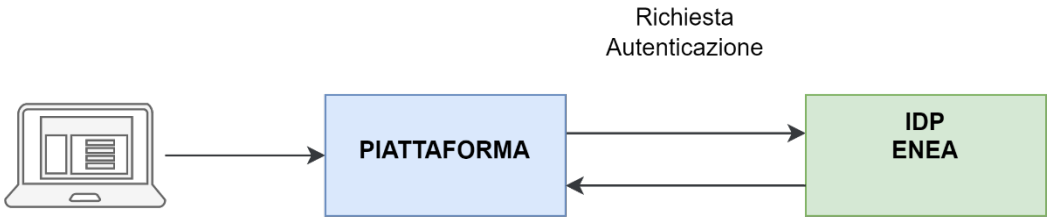

Figura 2 - Schema Logico IdP

Lista della API che vengono invocate dal DSM verso l'Identity Provider (IdP) di ENEA

Login

L'API Login consente di effettuare l'autenticazione dell'utente nell'applicativo. L'API restituirà il token che verrà salvato nel localStorage dell'applicativo.

URI: /rdsIDP/user/login?username={username}&password={password}
HTTP Method: POST
Formats: application/json
Parameters:

- Username (String)
- Password (String)

Registrazione

L'API Registrazione consente di effettuare la registrazione di un nuovo utente sull'applicativo.

URI: /rdsIDP/user/register
HTTP Method: POST
Formats: application/json
Parameters:

- email (String)
- name(String)
- organization(String)
- password(String)
- surname(String)
- usercommunities (Array)
    - communityid(String)
    - wallet(String)

Esempio di JSON nel request body:

```
email: parse.email,
name: parse.name,
organization: parse.organization,
password: parse.password,
surname: parse.surname,
usercommunities : [{
        communityid: parse.community,
        wallet: parse.wallet
}]
```

Logout
L'API Logout consente di effettuare la disconnessione dell'utente dall'applicativo. In caso di risposta positiva, l'applicativo provvederà all'eliminazione del token di autenticazione immagazzinato all'interno del localStorage e l'utente verrà re-indirizzato allo stato di login dell'applicazione.

URI: /rdsIDP/user/logout?token={token}
HTTP Method: POST
Formats: application/json
Parameters:

- token (String)

2.2 Nuovi servizi e interfacce per la gestione della comunità

2.2.1 Gestione incentivi ai fornitori

È stato introdotto un nuovo parametro di configurazione "incentivo", percentuale o fisso, per i fornitori dei servizi (similmente a quello per i fruitori dei servizi). L'incentivo introdotto è un valore in token versato al fornitore di un determinato servizio all'interno della comunità al momento della ricezione di token dal fruitore.
Sono state realizzate delle interfacce di amministrazione per perfezionare la gestione della configurazione della comunità.
È stato re-implementato e corretto il meccanismo di calcolo e pagamento dei servizi e integrati fra loro i meccanismi di tassazione, di incentivi al fruitore e al fornitore, modificando gli smart contract per la gestione del flusso dei token.

2.2.2 Nuova interfaccia per scambio di token fra utenti

È stata migliorata l'interfaccia per lo scambio di token fra utenti, per fornire all'utente un riassunto dettagliato dell'operazione che si sta effettuando (costo, tasse e incentivi applicati, entità del versamento finale). In particolare, quando l'utente inizia la fase di pagamento di un bene/servizio, la piattaforma presenterà un popup riassuntivo dei costi relativi a quel bene/servizio.

2.2.3 Gestione del bonus annuale

Attraverso l'utilizzo di interfacce esistenti e degli smart contract, è stato implementato un servizio di amministrazione per la gestione del **bonus token annuale**, ovvero il bonus che viene corrisposto una volta all'anno agli utenti della piattaforma, versato in una determinata data.

2.2.4 Demurrage

È stata introdotta la gestione del sistema di **demurrage**: questo consente di indicare il valore di soglia oltre il quale non è possibile accumulare token in un singolo wallet utente. Il demurrage deve essere scatenato con una periodicità da fissare (ad esempio ogni settimana, in un certo giorno e ad una certa ora), e i token presenti, se in eccesso, vengono sottratti dai wallet dei singoli utenti. Questi token possono essere inseriti in un secondo wallet di solidarietà della comunità oppure essere conteggiati e finire nel wallet di comunità.

2.2.5 Prezzo orario servizi di comunità e mutualistici

È stata creata un'interfaccia che ospita un form da compilare per impostare il costo orario di un servizio di comunità o mutualistico. Il costo orario sarà fisso e determinerà la tariffa oraria per un servizio.

2.3 Marketplace

È stato implementato sul sistema esistente un marketplace, suddiviso in categorie di offerte e relativi servizi a costo orario o fisso.

Il marketplace:

- consente all'utente registrato di offrire un bene o un servizio (il servizio avrà una tipologia presa da una lista preesistente creata dall'amministratore della piattaforma) su un marketplace digitale tramite una interfaccia a lui accessibile all'interno della propria home utente. L'utente potrà inserire una serie di informazioni sul servizio offerto (es. nome servizio, tipologia - importante per gli incentivi - descrizione, durata, orari disponibili, note) e fornire i suoi contatti (e-mail, telefono).
- successive comunicazioni fra gli utenti avvengono al di fuori della piattaforma di comunità tramite i contatti forniti e non sono gestite;
- permette ad un utente fornitore di un servizio la cancellazione del servizio stesso e l'eliminazione dal catalogo;
- consente ad un utente di visualizzare il catalogo delle offerte e selezionare il servizio che gli interessa. Il servizio selezionato è quindi accettato ed entra in una lista di servizi acquistati dall'utente;
- una volta fruito del servizio, tramite interfaccia utente, consente con un pulsante di confermare la fruizione del servizio-(bene) ed attivare il relativo processo di scambio token. Nel relativo processo di pagamento, il sistema calcola l'importo corretto (in base a costo, tasse, incentivi ecc.) e fornisce una schermata di sintesi (riepilogo dei costi, tasse e incentivi) relativa all'importo finale e, previa conferma dell'utente, effettua il versamento dei token dal wallet dell'utente fruitore del servizio al wallet dell'utente fornitore del servizio.

Il Marketplace è stato implementato in blockchain ma le attività di supporto ad esso sono svolte nel back-end.

2.4 AGID

La piattaforma rivolta agli utenti, MarketPlace, è stata verificata rispettando le norme di accessibilità AGID al fine di erogare servizi e fornire informazioni fruibili, senza discriminazioni, anche da parte di coloro che a causa di disabilità necessitano di tecnologie assistive.

2.5 Indicatori

Gli indicatori sono utilizzati dal community admin per visualizzare l'andamento del marketplace e più nel dettaglio consentono di visualizzare le informazioni in merito alle singole transazioni.
Gli indicatori sono stati sviluppati direttamente all'interno del back-end.

Sono state identificate e realizzate un set di API in grado di fornire degli indicatori all'esterno o alle interfacce di amministrazione della Dapp.

Gli indicatori individuati sono classificabili in:

- **Indicatori sugli utenti:**
  - Numero di utenti totale;
  - Numero di utenti registrati ultimo mese;
  - Utenti più attivi;
  - Numero di utenti attivi in un periodo da specificare (ad esempio se si specifica 1 mese, sono quelli con almeno una transazione nell'ultimo mese);
  - Numero di utenti inattivi in un periodo da specificare (ad esempio se si specifica 1 mese, sono quelli con nessuna transazione nell'ultimo mese);

- **Indicatori sulle transazioni:**
  - Numero di transazioni nell'ultimo mese;
  - Valore medio di una transazione;
  - Transazione più elevata;
  - Transazione più bassa;
  - Lista dei servizi usati dagli utenti in ordine di interesse.

- **Indicatori sui wallet:**
  - Valore medio dei wallet di comunità;
  - Numero token immessi (es. nell'ultimo mese);
  - Numero token spesi (es. ultimo mese);
  - Numero token ritirati (demurrage) (es. ultimo mese).

2.6 Soluzioni tecnologiche e modalità di sviluppo

In questo paragrafo sono descritti i framework, le librerie e gli ambienti utilizzati per lo sviluppo delle singole componenti della piattaforma. Per ogni componente viene anche indicata la struttura del codice con la descrizione dei principali package e delle risorse statiche.
Tutto il codice consegnato come elaborato del progetto è corredato di documentazione dettagliata come da standard attuali.

2.6.1 DSM – Data Service Manager

2.6.1.1 Ambiente, Framework e librerie, altre informazioni

Gli ambienti, framework e librerie non sono state modificate rispetto alla versione precedente. Si riportano per completezza.
Nel seguente paragrafo vengono descritte tutte le specifiche tecniche degli ambienti, framework, librerie e altre informazioni delle funzionalità aggiuntive sviluppate.

Ambiente di sviluppo

Ambiente	Java
Compatibilità Java	Java 11
Dependency Management	Maven

Framework e librerie usati per lo sviluppo

Framework	Spring
Boilerplate	Spring Boot 2.4.4
Layer autorizzativo	Spring Security 5.4.5
Connettore Blockchain	Web3j 5.0.0
Driver Database	org.postgresql 42.2.19
Gestione Token su standard JWT	com.auth0.java-jwt 3.15.0
Gestione testi/Stringhe	org.apache.commons 1.9
Libreria Crittografica	org.bouncycastle 1.68
Unit Test	Junit
Standard crittografici	Sha3, HMAC
Algoritmo di derivazione delle chiavi crittografiche	Custom

Framework per sviluppo delle interfacce web

Framework		Angular
Versione		10.2.3

2.6.1.2 Elenco delle nuove API

Nel seguente paragrafo vengono descritte le API REST che sono state realizzate:

- **Gestione MarketPlace**
  - Servizi
  - Categorie
  - Prenotazione e Pagamento di un servizio
- **Gestione KPI di piattaforma, ovvero Indicatori relativi a:**
  - Transazioni
  - Utenti
  - Servizi
- **Gestione KPI della Community, ovvero Indicatori relativi a specifica Community:**
  - Transazioni
  - Utenti
  - Servizi

Si rimanda al par. 2 per la collocazione della API REST all'interno dell'architettura logica.

Gestione MarketPlace

Route	Method	Description
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service	GET	Lista dei servizi appartenenti ad una categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service	POST	Creazione di un servizio appartenente ad una categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service/{_serviceId}	GET	Restituisce i dettagli del servizio
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service/{_serviceId}	POST	Modifica di un servizio all'interno della categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service/{_serviceId}	DELETE	Cancellazione di un servizio dalla categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId}	GET	Restituisce i dettagli della categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId}	POST	Modifica della categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId}	DELETE	Cancellazione della categoria
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service/{_serviceId}/pay	POST	Pagamento di un bene/servizio
/api/dsm/v1/community/{_communityId} /marketplace/serviceByUser	GET	Restituzione di tutti i servizi prenotati da un utente
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory	GET	Restituzione della lista delle categorie
/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory	POST	Creazione di una categoria
/api/dsm/v1/community/{_communityId} /marketplace/transactions	GET	Restituzione della lista di transazioni riguardanti una determinata comunità

/api/dsm/v1/community/{_communityId} /marketplace/serviceCategory/{_categoryId} /service/{_serviceId}/book	PUT	Prenotazione di un bene/servizio

Gestione KPI di piattaforma

Route	Method	Description
/api/dsm/v1/kpi/totalTransactionValue	GET	Mostra il valore totale delle transazioni
/api/dsm/v1/kpi/lessExpensiveTransaction	GET	Restituisce la transazione meno dispendiosa
/api/dsm/v1/kpi/activeUserWallets	GET	Mostra i wallet degli utenti attivi
/api/dsm/v1/kpi/mostExpensiveTransaction	GET	Restituisce la transazione più costosa
/api/dsm/v1/kpi/countActiveUsers	GET	Ritorna il numero totale degli utenti attivi
/api/dsm/v1/kpi/mintedTokens	GET	Restituisce il totale dei token assegnati
/api/dsm/v1/kpi/mostUsedServices	GET	Ritorna i servizi più usati
/api/dsm/v1/kpi/countTransactions	GET	Restituisce il conteggio totale delle transazioni
/api/dsm/v1/kpi/averageTransactionValue	GET	Mostra il valore della media delle transazioni

Gestione KPI della Community

Route	Method	Description
/api/dsm/v1/community/{_communityId}/kpi /countActiveUsers	GET	Ritorna il numero totale degli utenti attivi per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /countTransactions	GET	Restituisce il conteggio totale delle transazioni per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /averageTransactionValue	GET	Mostra il valore della media delle transazioni per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /mostExpensiveTransaction	GET	Restituisce la transazione più costosa per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /mostUsedServices	GET	Ritorna i servizi più usati per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /lessExpensiveTransaction	GET	Restituisce la transazione meno dispendiosa per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /mintedTokens	GET	Restituisce il totale dei token assegnati per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /totalTransactionValue	GET	Mostra il valore della media delle transazioni per la comunità richiesta
/api/dsm/v1/community/{_communityId}/kpi /activeUserWallets	GET	Mostra i wallet degli utenti attivi per la comunità richiesta

2.6.1.3 Descrizione delle risorse statiche

Nel seguente paragrafo vengono descritte le risorse statiche SQL e Contracts.

Sql: script per la creazione del database DMDB contenente dettagli configurativi dei modelli
Contracts: contiene gli smart contract utilizzati

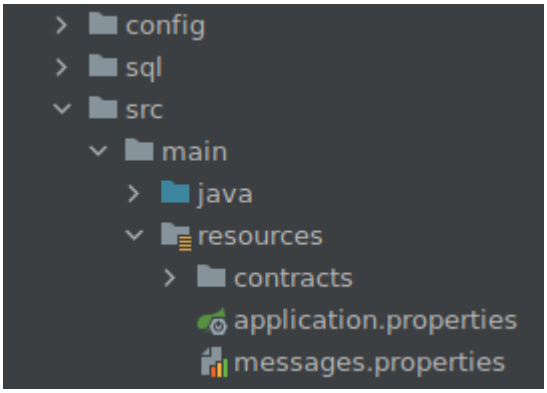

Figura 3 - Struttura delle risorse statiche del progetto

2.6.1.4 DMDB Data Model

La nuova versione del DMDB oltre ad essere utilizzato per la gestione dei modelli e delle loro relazioni con la comunità integra la gestione del marketplace e degli indicatori.

Nello specifico, sono state realizzate le seguenti tabelle:

- **MintedTokens**, contiene le informazioni relative alle assegnazioni dei token e viene utilizzata esclusivamente dal modulo indicatori;
- **Service Category**, contiene le informazioni delle categorie di servizi necessarie al corretto funzionamento del MarketPlace;
- **Service**, contiene le informazioni dei servizi necessari al corretto funzionamento del MarketPlace;
- **Transaction**, contiene le informazioni relative alle prenotazioni e acquisto dei servizi. Viene utilizzata sia dal modulo MarketPlace che dal modulo Indicatori.

Di seguito viene riportata la nuova struttura del DMDB:

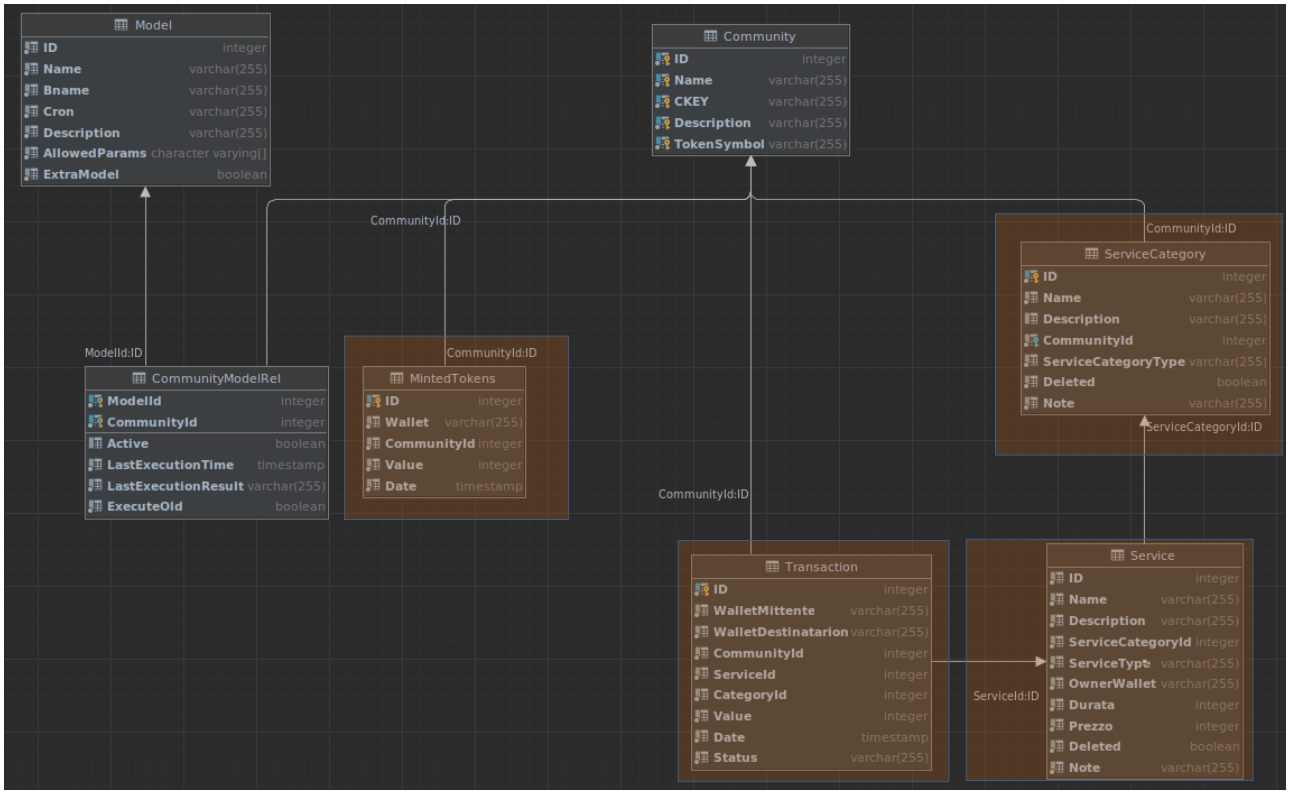

Figura 4 - Modello DBMS della Comunità Energetica

2.6.2 Blockchain BESU

I componenti del par. 2.6.2.1 della Blockchain BESU non sono stati modificati rispetto alla fase di progetto precedente. Si riportano per completezza.

Con il termine Hyperledger Besu si indicano tutte le componenti software che compongono il network blockchain della soluzione.

2.6.2.1 Componenti:

Di seguito vengono descritti tutti i componenti.

- **Besu**: è un client Open Source scritto in Java che offre le funzionalità di interfacciamento verso la blockchain e, di conseguenza, di analisi e verifica sulla correttezza delle stesse. Besu implementa diversi meccanismi di validazione delle transazioni, tra i quali anche quello utilizzato dalla soluzione: IBFT 2.0.
- **IBFT 2.0**: è il meccanismo di validazione utilizzato dalla soluzione attraverso il quale i validatori raggiungono la super maggioranza nel definire quali transazioni sono legittime per essere aggiunte in un blocco e quindi, nella catena dei blocchi della blockchain.
- **Validator**: la soluzione utilizza un totale di quattro validatori scelti randomicamente ad ogni nuova proposta di blocco da aggiungere alla catena dei blocchi. Nello specifico, il validatore selezionato, impacchetta le nuove transazioni in un blocco e propone il blocco agli altri validatori. Quando il numero minimo dei validatori (nella soluzione 3) valida il blocco proposto, il blocco viene aggiungo.
- **EthSigner**: ogni transazione, prima di poter essere inviata al client Besu, va firmata ed inviata ad un altro componente della blockchain l'EthSigner. L'EthSigner è un servizio che fa da proxy firmando e inoltrando le transazioni verso il client. Le chiavi utilizzate per firmarle le transazioni possono essere salvate in svariati modi, la soluzione proposta prevede l'utilizzo di Metamask.

- **Metamask**: è un plug-in per browser necessario per interagire con la DApp, esso ha il compito di memorizzare gli account degli utenti (chiave pubblica e chiave privata) e offrire un interfacciamento via RPC verso l'EthSigner.

2.6.2.2 Struttura del Codice

Di seguito viene descritta la struttura del codice.

- **Config**: folder contenente i file di configurazione per le diverse componenti software
- **Docker-compose.yml**: file yml di start-up di tutte le componenti in ambiente dockerizzato

Nel folder sono inoltre presenti cinque script utili ad interagire con l'applicazione

- **List.sh**: script di visualizzazione di tutti gli endpoint esposti dai diversi componenti
- **Remove.sh**: script di rimozione di tutti i container docker e la catena dei blocchi
- **Resume.sh**: script di riavvio dei contenitori docker
- **Run.sh**: script di avvio dei container docker
- **Stop.sh**: script per terminare i container docker

2.6.3 Smart Contract Tokenization System

Lo smart Contract Tokenization System è l'insieme di tutti gli Smart Contract dell'applicazione e delle loro interfacce.
Il linguaggio utilizzato per la creazione degli smart contract è Solidity compilato con la versione 0.8.2 di solc-js e compatibili con versioni di solidity comprese tra 0.7 e 0.9.

2.6.3.1 Interfacce

Nel seguente paragrafo vengono descritte le interfacce degli Smart Contracts che definiscono la logica di ogni funzione che viene a sua volta implementata all'interno del relativo Smart Contract.

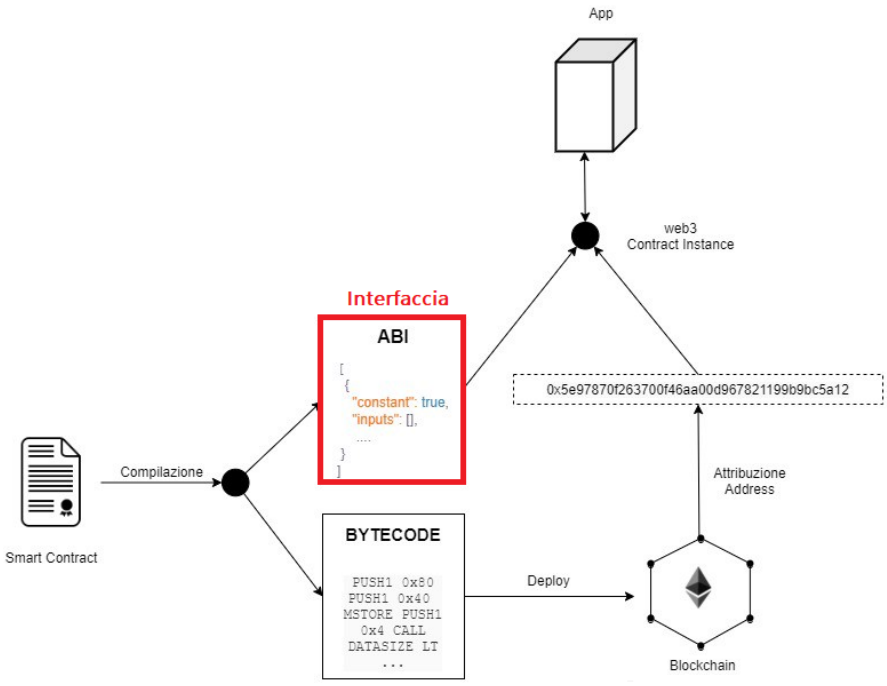

Figura 5 – Schema di deployment e interazione con lo Smart Contract

- **IAnnualBonus:** Utilizzata per la gestione del bonus annuale, essa definisce le seguenti funzioni:
  - Imposta il valore del bonus annuale, la data della sua prossima emissione e la comunità a cui deve essere assegnato
  - Implementa l'evento di assegnazione bonus
- **IDemurrage:** Utilizzata per la gestione del demurrage, essa definisce le seguenti funzioni:
  - Settaggio del limite di demurrage per una comunità
  - Settaggio del wallet sul quale inviare i fondi prelevati dagli utenti della comunità nel momento di avvio del demurrage
  - Implementa gli eventi di:
    - Variazione del limite di Demurrage
    - Variazione del wallet di Demurrage
    - Avvio del Demurrage
- **IMarketPlace:** Utilizzata per la compravendita di beni e servizi, essa definisce le seguenti funzioni:
  - Creazione e Rimozione delle categorie all'interno di una comunità
  - Creazione e Rimozione dei beni/servizi all'interno di una categoria
  - Prenotazione dei beni/servizi
  - Implementa gli eventi:
    - Creazione/Rimozione categoria
    - Creazione/Rimozione bene/servizio
    - Prenotazione del bene/servizio
    - Pagamento del bene/servizio

2.6.3.2 Smart Contract

Nel seguente paragrafo viene mostrata una lista di tutti i nuovi Smart Contract realizzati all'interno dello Smart Tokenization System. Ogni Smart Contract implementa una o più interfacce elencate al par. 2.7.3.1.

- **AnnualBonus**: Implementa l'interfaccia IAnnualBonus
- **Demurrage:** Implementa l'interfaccia IDemurrage
- **MarketPlace:** Implementa le interfacce IMarketPlace, IAccessControl, ICommunity, ICommunityUser

2.6.3.3 Truffle

I componenti del framework Truffle non sono stati modificati rispetto alla fase di progetto precedente. Si riportano per completezza.

Il framework di sviluppo utilizzato è Truffle, il quale offre un'ambiente automatizzato per:

- compilazione codice sorgente
- deploy degli smart contract su blockchain
- test degli smart contract
- trascodifica del bytecode degli smart contract in javascript

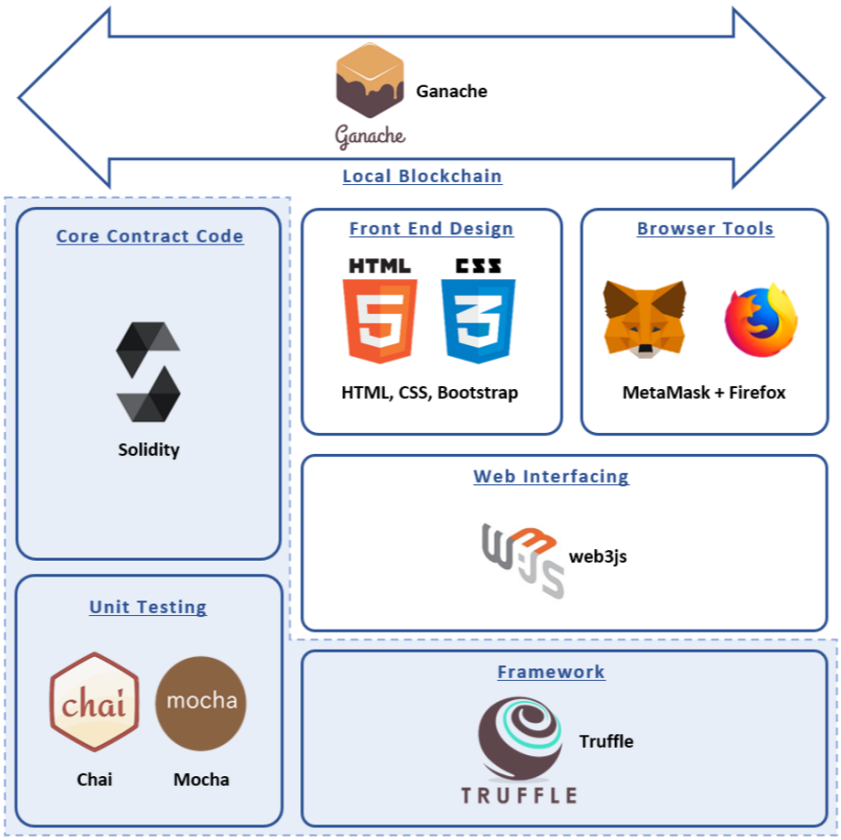

Figura 6 - Framework Truffle per Deplyment e Testing degli Smart Cotnract

2.6.3.4 Struttura del Codice

Nel seguente paragrafo viene descritta la struttura del repository che ospita gli Smart Contract dello Smart Tokenization System.

- contract: cartella contenente tutti gli smart contract e le interfacce
- docgen: cartella contenente la documentazione generata automaticamente per le singole funzionalità degli smart contract
- migration: cartella contenente i file utili per poter deployare gli smart contract su blockchain
- test: cartella contenente i file di test degli smart contract
- docify.js: script di creazione della documentazione contenuta nel folder docgen
- package.json: file contenente la descrizione, le dipendenze del progetto e gli script di utility
- truffle-config.js: file di configurazione di truffle.

2.6.3.5 Fase di Test

È stata condotta una fase di test della piattaforma attraverso l'implementazione di script di controlli e rilascio sugli Smart Contract, Modello dei Log, Market Place e Deploy dell'applicativo.
Nella radice del progetto è presente il file **package.json**, il quale contiene tutti metadati rilevanti della piattaforma e la definizione degli script utili affinché sia possibile effettuare tutte le fasi di test.

Di seguito vengono riportati gli script di test implementati:

Test di Verifica

- **test-accesscontrol**
  Script che consente di verificare lo smart contract identificato come "**AccessControl**"

- **test-community**
  Script che consente di verificare lo smart contract identificato come "**Community**"
- **test-erc1203**
  Script che consente di verificare lo smart contract identificato come "**Erc1203**"
- **test-logModel**
  Script che consente di verificare lo smart contract identificato come "**LogModel**"
- **test-marketplace**
  Script che consente di verificare lo smart contract identificato come "**MarketPlace**"
- **test-model**
  Script che consente di verificare lo smart contract identificato come "**Model**"

Test di Rilascio

- **deploy-to-ganache**
  Script che consente di eseguire rilascio degli Smart Contract sulla Blockchain di **Test.**

2.7 Funzionamento della Web Application

2.7.1 Homepage DAPP ENEA

Accesso con ruolo **community admin**. È possibile effettuare una delle seguenti operazioni:

- Aprire il Marketplace;
- Visualizzare il Dettaglio della Comunità e modificare le impostazioni della stessa;
- Aprire il Wallet di Comunità e vedere tutte le transazioni in ingresso/uscita;
- Visualizzare l'elenco degli utenti appartenenti alla comunità attraverso la voce di menu "Lista Utenti di Comunità".

Figura 7 - HomePage (Accesso come Community Admin)

Accesso con ruolo **community member** (normale utente). È possibile effettuare una delle seguenti operazioni:

- Visitare il Marketplace;
- Aprire il "Riepilogo Token" dove l'utente può:
  - visualizzare il saldo Token;
  - visualizzare i Token ricevuti da altri utenti;
  - aprire la lista di Token spesi;
  - consultare la lista di assegnazione Token.

Figura 8 - HomePage (Accesso come Community Member)

2.7.2 Marketplace

Prima pagina o "pagina di accoglienza" del Marketplace: è suddiviso in **categorie** e **servizi** usufruibili dalla comunità. I servizi sono contenuti dentro le categorie.

Il ruolo di **community admin** può:

- creare categorie di servizi;
- modificare o eliminare la categoria gestita.

Nella parte superiore si possono trovare le azioni che un utente admin può effettuare. Subito dopo, viene mostrata la lista delle categorie di servizi offerti da e per la comunità:

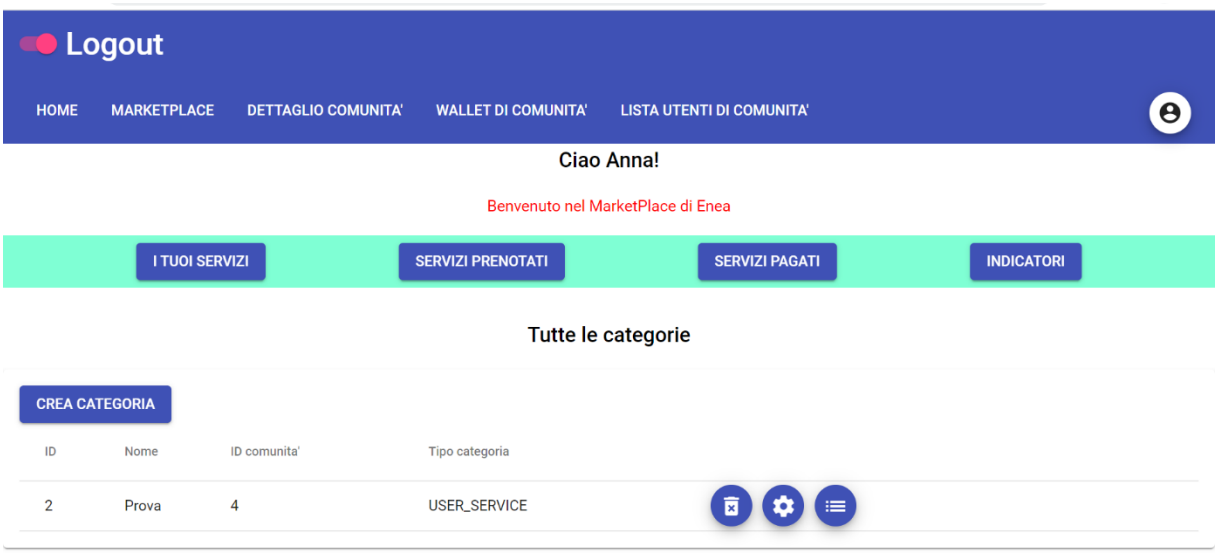

Figura 9 - Sezione MarketPlace (Accesso come Community Admin)

Il **community user** può:

- visualizzare l'elenco dei servizi;
- creare un servizio;
- modificare/cancellare un proprio servizio;
- prenotare/acquistare il servizio di un altro utente;
- consultare la lista dei servizi prenotati/acquistati.

2.7.3 Marketplace > Creazione di una categoria

La categoria deve avere un **nome** e può essere di **un solo tipo** tra i due disponibili: **USER_SERVICE** e **COMMUNITY_SERVICE**. È possibile aggiungere una **descrizione** e delle **note**.

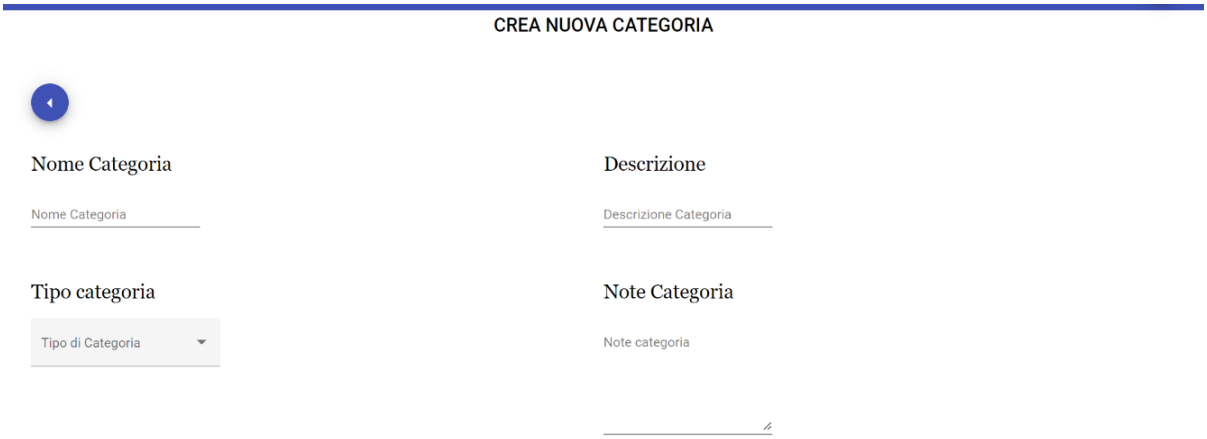

Figura 10 - Sezione Creazione Nuova Categoria

La schermata di creazione della categoria propone anche un form di compilazione per settare il **costo orario** di un servizio appartenente a quella categoria.
Il costo orario sarà fisso e determinerà la tariffa oraria per un servizio.
Ad esempio: se il costo orario è pari a 10 token, il servizio dura 3 ore. Il totale sarà di 30 token.

Il costo orario può essere **modificato** o **cancellato** con l'eliminazione della categoria.

Figura 11 - Sezione Creazione Nuova Categoria (Costo Orario)

Inoltre, si possono creare degli **incentivi** a **tasso fisso o percentuale**:

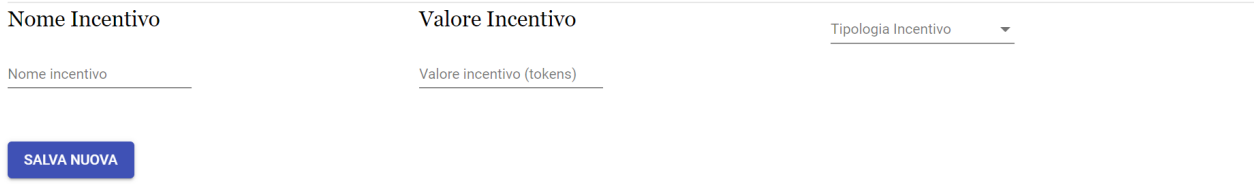

Figura 12 - Sezione Creazione Nuova Categoria (Incentivo)

2.7.4 Marketplace > Dettaglio di una categoria

Per ogni elemento della lista delle Categorie è possibile:

- Eliminare la categoria selezionata (solo **community admin**);
- Visualizzare e Modificare (solo **community admin**) la categoria scelta;
- Consultare la lista di servizi collegati alla categoria selezionata.

Figura 13 - Sezione MarketPlace (Lista Categorie)

2.7.5 Nuove opzioni disponibili per la gestione della comunità

La seguente interfaccia, che era già presente nella versione precedentemente rilasciata, è stata estesa con nuove opzioni.

Le nuove opzioni sono visualizzabili solo da utenti con ruolo "**community admin"** ed è accessibile mediante la voce del menù in alto **"Dettaglio Comunità"**.

Essa consente di:

- aggiungere un bonus di benvenuto per i nuovi utenti della comunità;

Figura 14 - Dettaglio Comunità (Impostazioni del Bonus di Benvenuto)

- inserire un Bonus annuale per la comunità;

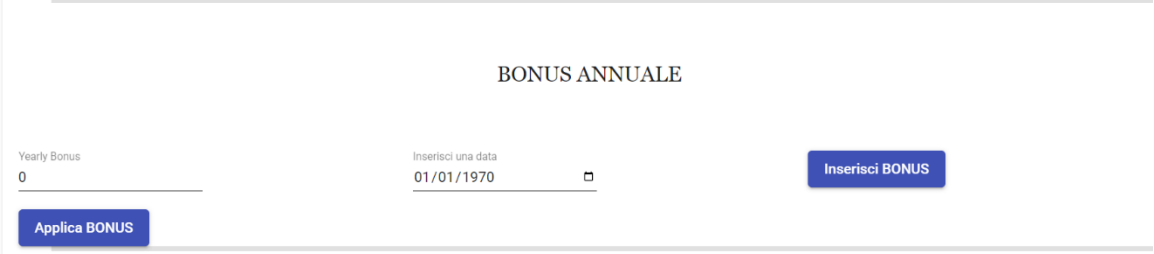

Figura 15 - Dettaglio Comunità (Impostazioni del Bonus Annuale)

- inserire un valore limite di demurrage e inserire il "Wallet" sul quale trasferire i Token decurtati dagli utenti della comunità in questione. Si riporta il dettaglio tecnico al par. 2.7.3.1.;

Figura 16 - Dettaglio Comunità (Impostazioni del Demurrage)

- inserire/modificare un wallet di comunità e impostare un valore per il Saldo Token iniziale:

Figura 17 - Dettaglio Comunità (Wallet di Comunità)

2.7.6 Wallet di comunità

La seguente interfaccia è visualizzabile solo dal **community admin**. Essa consente di:

- visualizzare la lista trasferimenti in ingresso;
- controllare la lista dei trasferimenti in uscita.

Figura 18 - Dettaglio Wallet di Comunità

2.7.7 Lista utenti di comunità

Nella seguente interfaccia è visualizzabile solo dal **community admin**. Essa consente di:

- visionare tutti gli utenti della comunità;
- visualizzare le informazioni di dettaglio dell'utente;
- aggiungere/rimuovere un utente dalla comunità;
- assegnare/rimuovere un ruolo all'utente.

Figura 19 - Lista Utenti di Comunità

2.7.8 Riepilogo Token

Il **community user** può vedere e gestire il proprio wallet e tutte le sue transazioni in ingresso:

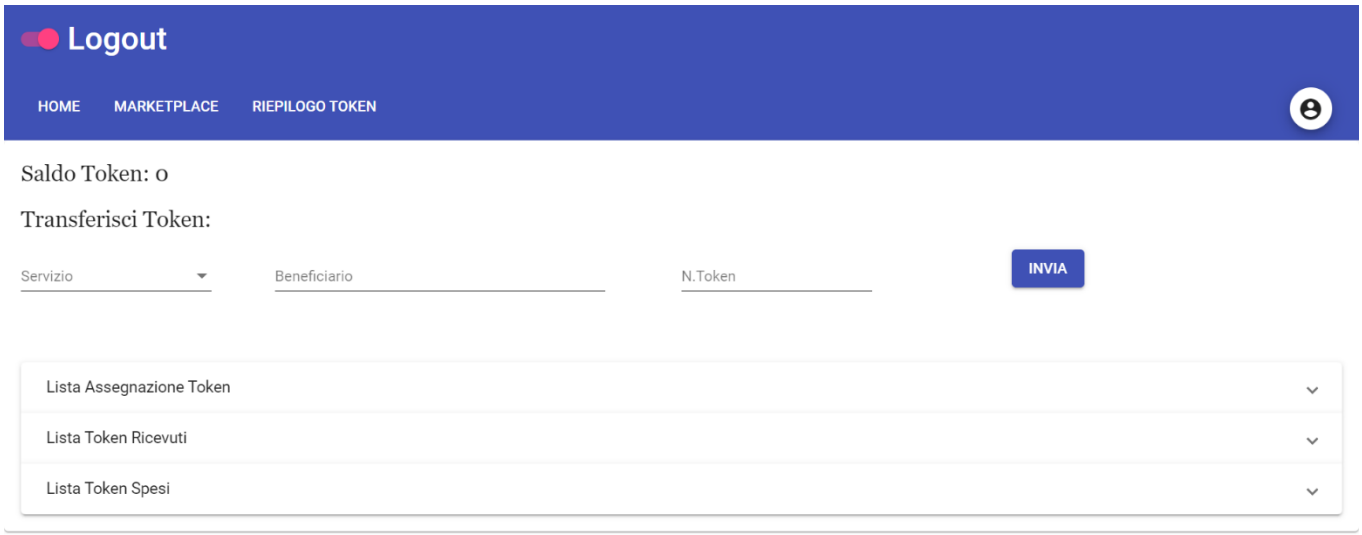

Figura 20 - Interfaccia di Gestione dei Token (Community Member)