



POR-FESR EMILIA ROMAGNA 2014-2020

Asse 1 - Ricerca e innovazione

Azione 1.2.2 - Supporto alla realizzazione di progetti complessi di attività di ricerca e sviluppo su poche aree tematiche di rilievo e all'applicazione di soluzioni tecnologiche funzionali alla realizzazione della strategia di S3

Bando 2018

Progetti di ricerca industriale strategica rivolti agli ambiti prioritari della Strategia di Specializzazione Intelligente



Sistemi interoperabili ed efficienti per la gestione sicura di filiere industriali

Deliverable D3.2: Progetto di dettaglio di SmartChain per filiere

Data di consegna prevista:	31 Gennaio 2022
Autori:	CRIS, CIRI ICT, CROSSTEC, MECHLAV, TTLAB
Versione:	1

Indice

1.	Introduzione.....	3
2.	Dettagli di progetto del Sistema di tracciatura della filiera.....	4
2.1.	Applicazione Web.....	5
2.2.	Gestione identità e chiavi.....	8
2.3.	Confidential Blockchain Manager (CBM)	12
2.4.	Smart contract Ethereum	14
3.	Progettazione di dettaglio sistema Carpigiani.....	16
3.1.	Funzionalità chaincode.....	20
3.2.	Funzionalità applicazione Web.....	21

1. Introduzione

In questo documento si descrive la progettazione di dettaglio delle piattaforme sviluppate nell'ambito del progetto SmartChain, includendo le tecnologie, le soluzioni software e le informazioni di dettaglio che permettono di istanziare il di progetto di alto livello descritto nel Deliverable 3.1 "Progetto di alto livello di SmartChain per filiere". Le soluzioni software impiegate sono quelle incluse nel Deliverable D4.1 "Piattaforma SmartChain (versione 1.0)". Questo documento non fornisce le informazioni necessarie ad installare e verificare il funzionamento del software, che sono descritte nel Deliverable D4.2 "Descrizione del software e manuale utente".

2. Dettagli di progetto del Sistema di tracciatura della filiera

Si descrive l'architettura di dettaglio del sistema di tracciatura della filiera, che ha come riferimento il caso d'uso dell'azienda Bianco Accessori, tramite la Figura 1.

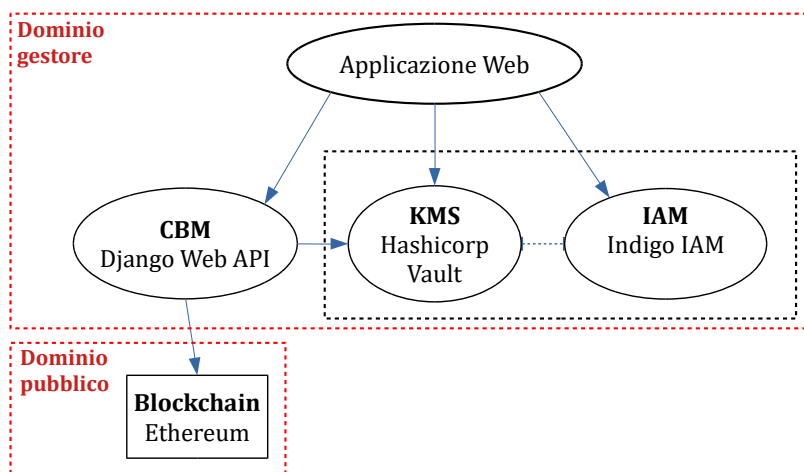


Figura 1 Architettura di tracciatura della filiera istanziata con componenti software

L'architettura ricalca quella di alto livello proposta del documento D3.1. "Architettura di alto livello di SmartChain per filiere", introducendo i componenti software impiegati per realizzarla:

- L' Applicazione Web è stata realizzata utilizzando il framework Javascript Angular 12 ed il framework Java Quarkus
- Il servizio di gestione delle identità (Identity and Access Management, o IAM) è stato realizzato impiegando il software Indigo IAM¹, opportunamente configurato per rispondere alle caratteristiche dell'architettura e dei requisiti del sistema applicativo;
- Il Key Management Service (KMS) è stato istanziato utilizzando il software Hashicorp Vault².
- Il Confidential Blockchain Manager (CBM) è stato realizzato tramite il framework di sviluppo Web open source Django³.
- La parte relativa alle tecnologie blockchain è stata realizzata tramite Smart Contract Ethereum.

Nel resto del documento si presentano le attività di progettazione di dettaglio relative a ogni componente.

¹ <https://github.com/indigo-iam/iam>

² <https://www.vaultproject.io/>

³ <https://www.djangoproject.com/>

2.1. Applicazione Web

L'applicazione web è la componente di front end per il progetto Smartchain: offre un'interfaccia grafica che permette agli utenti di avere accesso alle funzionalità del progetto in maniera conforme alle linee guida AGID. L'applicazione web si compone di quattro servizi, ripartiti uniformemente in due layer, come illustrato in Figura 2.

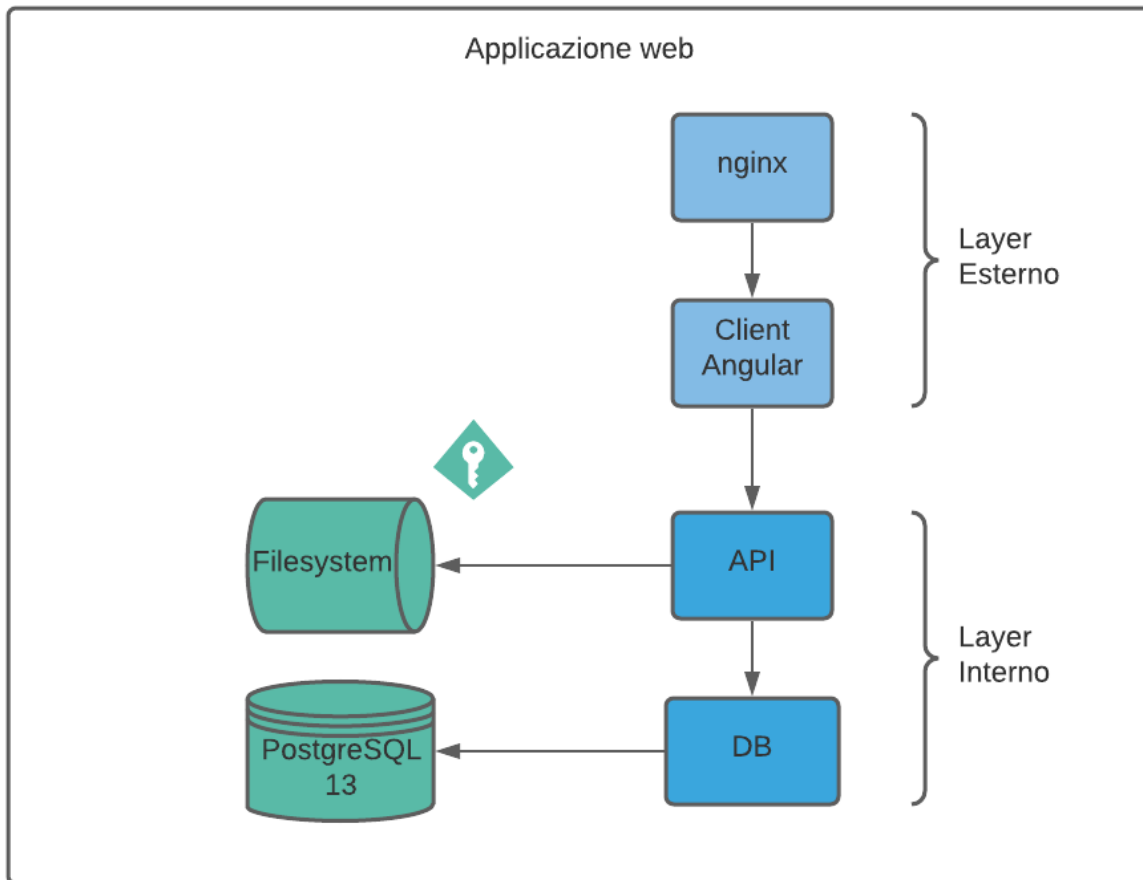


Figura 2 Architettura applicazione Web

Il layer più esterno, adibito all'interfaccia con utenti, è stato sviluppato attraverso il framework Angular 12 di Javascript. Al suo interno sono presenti il servizio `nginx` di proxy che mette a disposizione degli utenti il servizio client dell'applicazione. Il layer più interno, adibito alla gestione dei dati, è stato sviluppato attraverso il framework Java Quarkus. Al suo interno sono presenti il servizio di API, che ha il compito di interrogare le componenti di back end del progetto Smartchain oltre a fornire un microservizio di accesso alla mount del filesystem.

Il servizio di DB è adibito alla gestione dell'infrastruttura dati interna costruita utilizzando PostgreSQL 13.

Servizio	Immagine	Mount
smartchain-proxy	nginx	-
smartchain-client	smartchain-client	-
smartchain-api	smarthchain-api	filesystem-data
smartchain-db	postgresql:13	postgres-data

2.1.1. Interfacce utenti

La home del client mette a disposizione collegamento per autenticarsi a Smartchain; il processo di autenticazione viene gestito da INDIGO IAM con cui l'applicazione web interagisce tramite protocollo OIDC. In caso di login eseguito correttamente, viene eseguito un callback alla pagina dell'applicazione web. A seconda delle credenziali di autenticazione inserite, la web application reindirizza l'utente alla sua pagina principale. Se nessuna autenticazione ha luogo, si può comunque accedere alle informazioni pubbliche di Smartchain come utente "guest". Se l'access token recuperato dal componente IAM riporta un'appartenenza al gruppo "admin-webapp", l'utente ha accesso alle funzionalità di amministratore.

Un utente amministratore può validare le registrazioni pending e decidere se approvare o rifiutare una registrazione; inoltre, un amministratore può visualizzare tutti i contenuti inseriti da parte degli utenti registrati per verificare il corretto comportamento degli utenti.

The screenshot shows a user interface for an administrator. At the top, there is a blue navigation bar with the user's name 'Benvenuto Mario Rossi' and an 'ESCI' button. Below the navigation bar, there are three menu items: 'HOME', 'IMPRESE REGISTRATE', and 'RICHIESTE REGISTRAZIONI'. The main content area features a welcome message 'Benvenuto Mario Rossi' and a description: 'Questa piattaforma consente di visualizzare tutte le informazioni dell'impresa registrate e di gestire le nuove richieste di registrazione'. Below this, there are two white cards with blue borders. The first card is titled 'ATTUALI IMPRESE REGISTRATE' and shows a large blue number '150' next to a building icon. Below the number is a blue button with the text 'ACCEDI ALLA SEZIONE' and a right arrow. The second card is titled 'NUOVE RICHIESTE DI REGISTRAZIONE' and shows a large blue number '45' next to a document icon. Below the number is a blue button with the text 'ACCEDI ALLA SEZIONE' and a right arrow.

Ogni altra utenza è considerata un'utenza impresa. Un utente impresa senza account Smartchain viene reindirizzato alla componente IAM per effettuare una richiesta di registrazione. Una volta approvata la richiesta, alla sua prima autenticazione l'utente impresa trova sulla pagina principale della web app un tasto funzione per registrare la propria impresa dove gli saranno richieste le informazioni e i dati non critici di impresa.

The screenshot shows a web application interface for company registration. The top navigation bar is blue with the text 'HOME', 'IMPRESE REGISTRATE', and 'LA MIA REGISTRAZIONE'. On the right, there is a user profile 'Benvenuto Mario Rossi' and an 'ESCI' button. Below the navigation, there is a breadcrumb 'Home / La mia registrazione'. The main content area has a 'SALVA' button and an 'INVIA REGISTRAZIONE' button. The form is titled 'DATI IMPRESA' and is divided into four sections: 'DATI ANAGRAFICI', 'STRUTTURA AZIENDA', 'CERTIFICATI', and 'FORNITORI'. The 'DATI IMPRESA' section includes several input fields: 'Ragione sociale', 'Partita IVA', 'Indirizzo legale', 'Link al sito dell'impresa', 'Descrizione testuale dell'impresa', 'Smart Chain Account', 'Posta Elettronica Certificata', 'Indirizzo email pubblico', and 'Contatto telefonico pubblico'.

Al termine dell'inserimento delle informazioni, l'utente può richiedere al sistema di registrare la propria azienda premendo un tasto specifico della Web Application. Il sistema registra la richiesta e tramite access token JWT vengono effettuate le chiamate alla componente Vault e alla componente CBM per la registrazione di un'impresa illustrate nel Deliverable 3.1. Completato il processo, un'impresa registrata può accedere alle complete funzionalità di Smartchain, inserire e modificare tutti i dati critici e non critici relativi alla propria impresa, certificazioni ed elenco fornitori, secondo le logiche illustrate nel Deliverable 3.1. La web app non detiene nel suo database le informazioni relative ai fornitori di un'impresa registrata.

In caso di inserimento di certificati, la Web Application calcola l'hash del documento caricato con algoritmo SHA3-256 da inserire come dalle logiche illustrate nel Deliverable 3.1. In fase di inserimento del documento, viene richiesto all'utente di scegliere di far memorizzare il file del certificato nell'applicazione. Se l'utente decide di far memorizzare il certificato, questo viene cifrato ed inserito nel filesystem. In caso contrario, il certificato non viene memorizzato e l'impresa si farà carico di condividere nuovamente i certificati in caso di richieste di verifiche.

2.1.2. Infrastruttura Database

Il database interno della Web Application ha un'infrastruttura relazionale realizzata in PostgreSQL 13. Si riporta in Figura 3 il diagramma dell'infrastruttura.

Gli schemi “certificate” e “document” fanno riferimento ai file caricati nel filesystem; sono privati e vengono utilizzati dalla Web Application internamente per recuperare o eliminare a richiesta dell’utente che ne ha effettuato l’upload in precedenza.

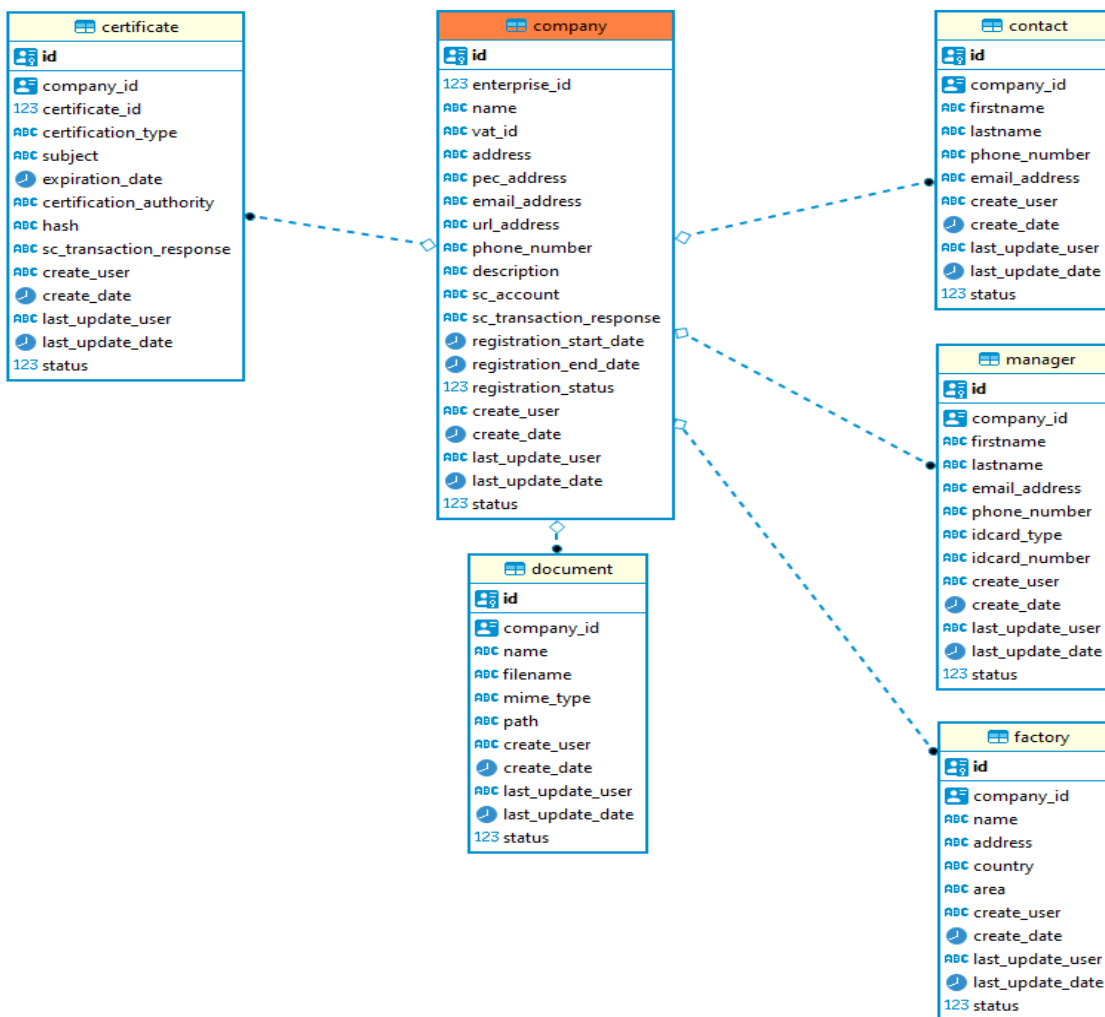


Figura 3 Infrastruttura database interno della Web application

2.2. Gestione identità, ruoli e chiavi

Il sistema progettato prevede la gestione centralizzata delle identità da parte dell’IAM tramite protocolli di Federazione delle Identità e Single Sign On, e la gestione delle autorizzazioni per l’accesso alle funzionalità del KMS.

L’architettura prevede tre tipologie di ruoli per l’accesso alle funzionalità del KMS:

- **Amministrazione del KMS**, associato a tutte le tipiche funzionalità di controllo avanzato delle impostazioni del componente. In Figura 4 si mostra il dettaglio della policy di Vault che definisce le regole di accesso per questo ruolo;
- **Amministrazione delle imprese**, associato a funzionalità di attivazione e disattivazione di imprese, e all'abilitazione e disabilitazione di funzionalità di accesso ai dati specifici. In Figura 5 si mostra il dettaglio della policy di Vault che definisce le regole di accesso per questo ruolo;
- **Gestione dei dati di un'impresa**, associato alle funzionalità necessarie per ogni impresa per poter cifrare e decifrare i propri dati memorizzati sul KMS. Da questo punto di vista, si identificano due tipi di policy:
 - ogni impresa può gestire arbitrariamente chiavi di cifratura afferenti a un proprio keyring, utilizzando una chiave per cifrare i dati associati a ciascun fornitore differente, e derivando delle sotto-chiavi in base al tipo di informazione da cifrare (ad esempio, il ruolo che lega il rapporto di fornitura o le certificazioni dei rapporti di fornitura vengono cifrate con due chiavi differenti derivate dalla stessa chiave). La Figura 6 mostra i dettagli della policy definita per gestire correttamente questo tipo di regole di accesso;
 - ogni impresa può richiedere l'utilizzo della chiave impiegato da un'impresa committente per cifrare dati nell'ambito del rapporto di fornitura fra le due imprese. Questo tipo di permesso consente di modellare la possibilità ad ogni fornitore di verificare i dati inseriti da altre imprese a loro nome. La Figura 7 mostra i dettagli della policy che regola questo tipo di autorizzazione.

```

# Read system health check
path "sys/health" {
  capabilities = ["read", "sudo"]
}
# Create and manage ACL policies broadly across Vault
# List existing policies
path "sys/policies/acl" {
  capabilities = ["list"]
}
# Create and manage ACL policies
path "sys/policies/acl/*" {
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}
# Enable and manage authentication methods broadly across Vault
# Manage auth methods broadly across Vault
path "auth/*" {
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}
# Create, update, and delete auth methods
path "sys/auth/*" {
  capabilities = ["create", "update", "delete", "sudo"]
}
# List auth methods
path "sys/auth" {
  capabilities = ["read"]
}
# Manage secrets engines
path "sys/mounts/*" {
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}
# List existing secrets engines.
path "sys/mounts" {
  capabilities = ["read"]
}
# Create and manage entities and groups
path "identity/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}

```

Figura 4 Policy amministrazione Vault

```

# Manage secrets engines
path "sys/mounts/transit/${IAM_GROUP_COMPANY_PREFIX}*" { capabilities
= ["create", "delete", "update"]}

## Enable user to list only at other paths
path "transit/*" {
  capabilities = ["list"]
  allowed_parameters = { "*" = [] }
}

```

Figura 5 Policy gestione imprese

```

#Enable user for all operations at their path
path "kv/{{identity.entity.aliases.${ACCESSOR}.metadata.auth_company}}/*"
{
    capabilities = ["create","update","list","read","delete"]
    allowed_parameters = { "*" = [] }
}
# Enable user to list only at other paths
path "kv/*" {
    capabilities = ["list"]
    allowed_parameters = { "*" = [] }
}
#Enable user for all operations at their path
path
"transit/{{identity.entity.aliases.${ACCESSOR}.metadata.auth_company}}/*"
{
    capabilities = ["create","update","list","read","delete"]
    allowed_parameters = { "*" = [] }
}
# Enable user to list only at other paths
path "transit/*" {
    capabilities = ["list"]
    allowed_parameters = { "*" = [] }
}
# Enable user to encrypt/decrypt data associated with the company they
# have been associated with
path
"transit/{{identity.entity.aliases.${ACCESSOR}.metadata.auth_company}}/enc
rypt/${IAM_GROUP_COMPANY_PREFIX}*" {
    capabilities = ["create", "update"]
    required_parameters = ["plaintext", "context", "type"]
    allowed_parameters = {
        "plaintext" = []
        "context" = []
        "type" = ["aes256-gcm96"]
    }
}
path
"transit/{{identity.entity.aliases.${ACCESSOR}.metadata.auth_company}}/dec
rypt/${IAM_GROUP_COMPANY_PREFIX}*" {
    capabilities = ["create", "update"]
    required_parameters = ["ciphertext", "context"]
    allowed_parameters = {
        "ciphertext" = []
        "context" = []
    }
}
}

```

Figura 6 Policy gestione keyring da parte di un'impresa

```

# Enable user to decrypt ciphertexts generated by other parties for
himself
path
"transit/+/decrypt/{{identity.entity.aliases.${ACCESSOR}.metadata.auth_com
pany}}" {
  capabilities = ["create", "update"]
  required_parameters = ["ciphertext", "context"]
  allowed_parameters = {
    "ciphertext" = []
    "context" = []
  }
}
}

```

Figura 7 Policy decifratura dati forniture

2.3. Confidential Blockchain Manager (CBM)

Il CBM è il componente software che fa da tramite tra l'applicazione web, il componente di autorizzazione e crittazione (KMS) e lo smartcontract deployato sulla blockchain Ethereum. Una volta che la web application invoca una delle funzioni offerte dal CBM, esso interagisce prima con il componente di autorizzazione per controllare se il chiamante ha sufficienti privilegi per eseguire la funzione chiamata. In particolare, il CBM valida il token fornito dall'applicazione web invocando la funzione di validazione offerta da Hashicorp Vault e, se l'utente possiede i privilegi richiesti, si connette e invoca le funzioni sullo smart contract presente sulla blockchain Ethereum eseguendo così l'azione richiesta.

Il CBM espone le seguenti REST API scritte seguendo le best practice basate sul concetto di risorsa REST. Le REST API sono costituite da 4 gruppi contenenti 5 chiamate REST per ciascuno. Ogni gruppo lavora su una risorsa o sotto risorsa specifica: impresa, fornitore registrato, fornitore non registrato e certificato.

impresa Gestione delle imprese

GET	/impresa/	ottiene la lista delle imprese
POST	/impresa/	crea una nuova impresa
GET	/impresa/{id_impresa}	ottiene i dettagli di una singola impresa cercando by ID
PUT	/impresa/{id_impresa}	aggiorna i dati di un'impresa esistente (by ID)
DELETE	/impresa/{id_impresa}	elimina un'impresa by ID

Il primo gruppo si occupa dell'interazione con la risorsa impresa, sulla risorsa è possibile eseguire 5 operazioni: registrare un'impresa, ottenere i dettagli di un'impresa, ottenere una lista di tutte le imprese registrate, modificare i dati di un'impresa e cancellare un'impresa.

fornitore registrato Gestione dei fornitori registrati

GET	/impresa/{id_impresa}/fornitoreregistrato	ritorna la lista delle imprese registrate
POST	/impresa/{id_impresa}/fornitoreregistrato	aggiunge un nuovo fornitore registrato ad un'impresa
GET	/impresa/{id_impresa}/fornitoreregistrato/{id_relation}	ritorna uno specifico fornitore registrato di un'impresa by ID
PUT	/impresa/{id_impresa}/fornitoreregistrato/{id_relation}	modifica un fornitore registrato di un'impresa
DELETE	/impresa/{id_impresa}/fornitoreregistrato/{id_relation}	invalida il fornitore registrato di un'impresa

fornitore non registrato Gestione dei fornitori non registrati

GET	/impresa/{id_impresa}/fornitorenonregistrato	ritorna la lista delle imprese non registrate
POST	/impresa/{id_impresa}/fornitorenonregistrato	aggiunge un nuovo fornitore non registrato ad un'impresa
GET	/impresa/{id_impresa}/fornitorenonregistrato/{id_relation}	ritorna uno specifico fornitore non registrato di un'impresa by ID
PUT	/impresa/{id_impresa}/fornitorenonregistrato/{id_relation}	modifica un fornitore non registrato di un'impresa
DELETE	/impresa/{id_impresa}/fornitorenonregistrato/{id_relation}	invalida il fornitore non registrato di un'impresa

Il secondo e terzo gruppo si occupano rispettivamente della sottorisorse fornitore registrato e fornitore non registrato. I sottogruppi sono sempre in relazione ad un'impresa identificata dall'id_impresa e hanno un identificativo che ne identifica la relazione (id_relation). Analogamente alle REST API dell'impresa, anche in questo caso abbiamo 5 operazioni possibili: registrazione di un fornitore, dettagli del fornitore, lista dei fornitori, modifica e cancellazione dei fornitori. All'atto della creazione di un fornitore, nel caso di fornitori già registrati viene creata una relazione tra l'impresa e il fornitore mentre nel caso di fornitori non registrati viene dapprima registrato il fornitore e poi creata la relazione

certificato Gestione dei certificati

GET	/impresa/{id_impresa}/certificato	ritorna la lista dei certificati
POST	/impresa/{id_impresa}/certificato	aggiunge un nuovo certificato ad un'impresa bt ID
GET	/impresa/{id_impresa}/certificato/{id_cert}	ritorna uno specifico certificato di un'impresa by ID
PUT	/impresa/{id_impresa}/certificato/{id_cert}	modifica un certificato di un'impresa by ID
DELETE	/impresa/{id_impresa}/certificato/{id_cert}	invalida il certificato di un'impresa

Il quarto gruppo invece si occupa della sotto-risorsa certificato che mantiene tutte le certificazioni legate all'impresa. Ciascun certificato è identificato da un identificatore univoco id_cert legato ad uno specifico id_impresa. Anche in questo caso i metodi offerti sono creazione, dettagli, cancellazione e modifica di un certificato e lista di tutti i certificati relativi a quell'impresa.

2.4. Smart contract Ethereum

2.4.1. Logiche di autorizzazione

Lo smart contract applica logiche di autorizzazione per tutte quelle operazioni che eseguono che inseriscono o modificano le strutture dati mantenute dallo smart contract, ossia che effettuano operazioni di scrittura. Le operazioni che necessitano di autorizzazione sono le seguenti:

Operazione	Policy
Registrazione di un'impresa	solo l'utente admin può inserire un'impresa
Modifica impresa	un'impresa può modificare solo i propri dati
Disabilitazione impresa	un'impresa può disabilitare solo sé stessa
Inserimento fornitore registrato	un'impresa può inserire fornitori solo per sé stessa
Inserimento fornitore non registrato	un'impresa può inserire fornitori non registrati solo per sé stessa
Modifica rapporto fornitura	un'impresa può modificare solo i propri rapporti di fornitura
Eliminazione rapporto fornitura	un'impresa può eliminare solo i propri rapporti di fornitura

Inserimento certificazione	un'impresa può inserire certificazioni solo per sé stessa
Modifica certificazione	un'impresa può modificare solo le proprie certificazioni
Eliminazione certificazione	un'impresa può eliminare solo le proprie certificazioni

Le logiche di autorizzazione elencate sono riassumibili come segue: solo imprese registrate possono effettuare operazioni di scrittura e possono modificare esclusivamente i propri dati.

Per implementare tutte le logiche di autorizzazione di scrittura da parte di un'impresa registrata si è adottata una rappresentazione delle strutture dati che include un identificativo dell'impresa a cui la struttura dati fa riferimento. Di seguito denotiamo l'impresa a cui fa riferimento una struttura dati come "proprietario", e denotiamo l'impresa che invoca una operazione di scrittura come "chiamante". La rappresentazione adottata permette di esprimere le logiche di autorizzazione nel seguente modo: se l'identificativo del chiamante non è uguale a quello del proprietario allora nega la scrittura, permetti altrimenti.

A titolo esemplificativo riportiamo la struttura dati che modella una certificazione, scritta in linguaggio Solidity:

```
struct Certification {
    uint id;
    string issuer;
    string certType;
    string object;
    uint validity;
    uint enterpriseId;
}
```

Figura 8

Il campo `id` riportato in Figura 8 è l'id del proprietario della certificazione. In fase di autorizzazione l'id del proprietario viene confrontato con l'id del chiamante. Se i due id coincidono, allora il chiamante può modificare i dati relativi alla certificazione.

Gli id delle imprese vengono stabiliti dallo smart contract al momento della registrazione di un'impresa e non possono essere modificati al fine di garantire l'integrità delle informazioni.

2.4.2. Gestione dati cifrati

Lo smart contract memorizza alcune informazioni riguardo le aziende che non devono essere pubblicamente accessibili: la partita IVA (*VATNumber*), il nome dell'azienda (*name*) e il luogo in cui opera (*location*). A tale scopo, lo smart contract supporta la memorizzazione di dati cifrati che vengono rappresentati come vettori opachi di byte. Nel linguaggio Solidity il tipo di dato adottato per memorizzare dati cifrati è `bytes`.

Come già evidenziato in Sezione 2.4.1, i dati cifrati non vengono utilizzati dallo smart contract per garantire le logiche di autorizzazione. Questo permette di considerarli come dati opachi su cui non vengono effettuati controlli.

3. Progettazione di dettaglio sistema Carpigiani

Nell'allegato tecnico D3.1 si è discusso della progettazione ad alto livello di un'infrastruttura che possa rispettare i requisiti evidenziati in fase di modellazione per il sistema Carpigiani (si veda l'allegato D2.2). A riguardo, il sistema progettato si occupa principalmente di visualizzare le informazioni inerenti ai refill delle macchine da gelato e di gestire le suddette macchine e la rete blockchain. Oltre a queste caratteristiche, è stata espressa la necessità di mantenere i dati presenti nelle transazioni riservati e di ridurre al minimo l'onere di gestione dell'infrastruttura nei confronti del personale afferente agli esercenti e alle aziende fornitrici di miscela.

L'infrastruttura progettata è stata realizzata per essere modulare, in modo che in fase implementativa possa risultare più versatile, permettere la suddivisione dei vari componenti su più server e garantire una loro replicazione al fine di favorire una maggiore scalabilità e resilienza del sistema, quindi consentendo di semplificare la manutenzione dell'infrastruttura nel suo complesso. Si vuole evidenziare, inoltre, che il nodo Blockchain, che in fase progettuale si pensava di inserire nella macchina da gelato, nella fase implementativa è stato spostato in Cloud. Le motivazioni di tale decisione verranno discusse nell'apposito allegato tecnico D4.3.

La Figura 3.1 mostra il prototipo dell'architettura implementata seguendo le linee guida di più alto livello presentate nel documento D3.1. Quest'ultimo riguarda la sua progettazione, la quale segue le tre classi di dominio applicativo dichiarate in fase di modellazione (utente, gestore e macchina). In aggiunta, nei rispettivi domini vengono anche illustrati i componenti software utilizzati in fase di realizzazione.

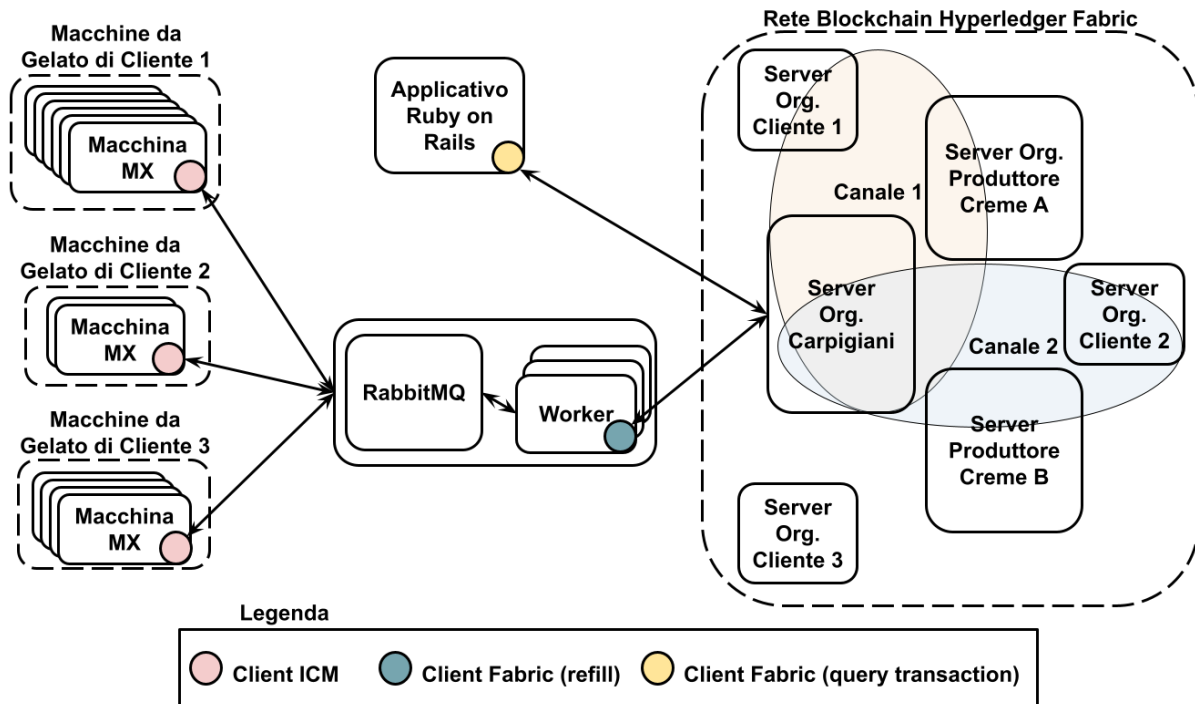


Figura 3.1 - Infrastruttura Blockchain per il refill di macchine da gelato.

In dettaglio, l'architettura implementata vede coinvolti i seguenti attori nei rispettivi domini applicativi:

- **Rete blockchain (dominio gestore).** I componenti di questa rete possono essere installati su differenti server in base alle diverse organizzazioni che partecipano alla rete, suddivisa logicamente in canali. Per tale componente si è utilizzato il software **Hyperledger Fabric**.
- **Broker di messaggistica (dominio gestore).** Questo strumento può essere installato su un server posto al di fuori della rete blockchain. Per la sua implementazione si è fatto uso del software **RabbitMQ** per diversi motivi funzionali, tra cui:
 - è una tra le applicazioni di messaggistica più usata e mantenuta, con una vasta community che fornisce: vari client implementati con linguaggi di programmazione differenti, plugin, guide, ecc;
 - garantisce prestazioni con alta affidabilità e servizi, come ad esempio persistenza, conferme di consegna e alta disponibilità;
 - fornisce un servizio di tracciamento per riconoscere un problema in fase di configurazione;
 - è risultata la scelta più leggera per gli aspetti relativi all'uso della memoria, del carico computazionale e dell'utilizzo della rete.
- **Applicativo Web (dominio gestore).** Per l'implementazione di tale componente è stato utilizzato il framework **Ruby on Rails** per i seguenti motivi:

- l'azienda Carpigiani aveva già esperienza nell'uso di tale framework in quanto era stato precedentemente utilizzato in altri contesti implementativi;
- agevola notevolmente la velocità di sviluppo ed è risultata la scelta più valida per le performance offerte.
- **Worker (dominio gestore).** Processo software che rimane in attesa dei messaggi provenienti dalle macchine, invocando la richiesta di transazione nell'opportuno canale presente nella rete Blockchain e restituendo, alla macchina da gelato, il risultato relativo dell'operazione effettuata.
- **Client Ice Cream Machine - ICM (dominio macchina).** Software proprietario che risiede nella macchina da gelato Carpigiani e ha il compito di inviare, tramite protocollo MQTT, un messaggio contenente le informazioni riguardanti l'operazione di refill e altri parametri relativi allo stato della macchina.

In Figura 3.2 è illustrato il diagramma di flusso tra i vari attori precedentemente descritti, partendo dalla richiesta di transazione contenente il refill, effettuata dalla macchina da gelato, fino alla notifica di avvenuta o fallita registrazione del suddetto refill. Si noti che in questo diagramma di flusso non è presente la componente di visualizzazione delle transazioni tramite Applicazione Web, in quanto viene considerata un'azione asincrona rispetto all'operazione di registrazione del refill sulla Blockchain.

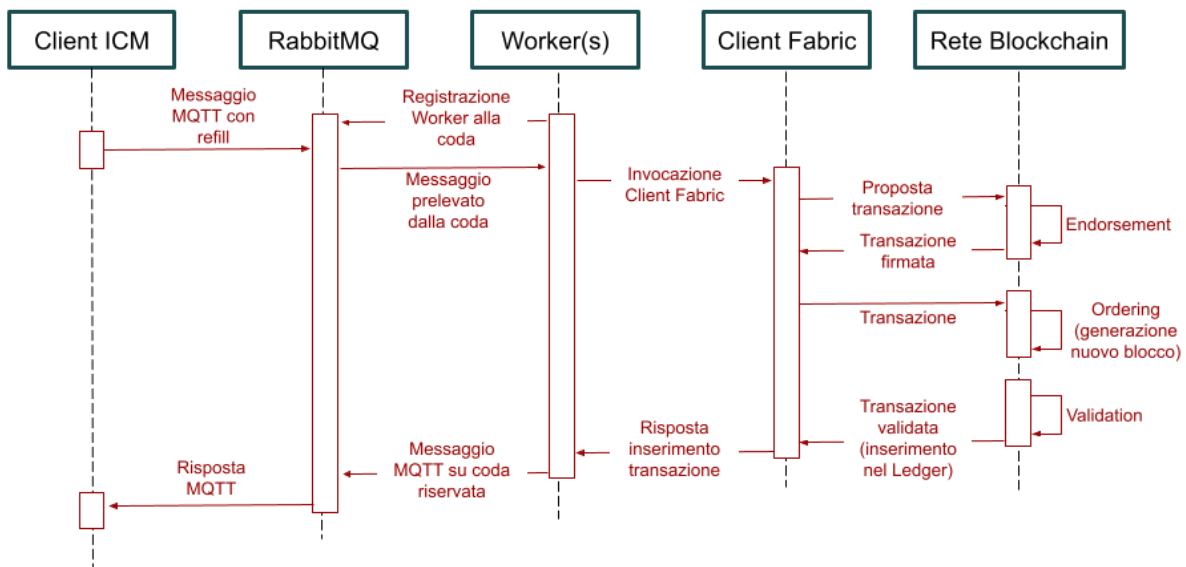


Figura 3.2 - Diagramma di sequenza per l'inserimento della transazione di refill.

In dettaglio, la figura mostra i seguenti passi:

1. il Client ICM crea un messaggio MQTT contenente le informazioni riguardanti il refill lo stato attuale della macchina;

2. il Client ICM effettua una mutua autenticazione presso RabbitMQ e, dopo aver inserito il messaggio di richiesta su di una coda MQTT, rimane in attesa di una risposta da parte del processo Worker. Nello specifico, la comunicazione tra Client e Worker avviene secondo il paradigma Remote Procedure Call (RPC)⁴ offerto dal broker. In particolare, il Client invia il messaggio in una coda comune a tutti gli altri Client ICM, specificando che la risposta al messaggio di richiesta avverrà su di una coda differente, temporanea e riservata. Questa viene indicata come parametro del messaggio e non possono accedere gli altri client. Questo meccanismo consente di avere maggiore sicurezza in quanto risulterebbe difficile per client esterni sapere in quale coda verrà inviata la risposta;
3. diversi processi Worker sono registrati sulla coda dei messaggi provenienti dai Client ICM e, una volta inserito un messaggio, il primo Worker disponibile si incarica di preparare la proposta di transazione da inserire nella blockchain per mezzo di un apposito software realizzato come Client Fabric;
4. a seconda dell'identificativo della Macchina da Gelato presente nel messaggio, il Worker individua il canale corretto composto dalla terna <ID_Cliente, ID_Produttore, ID_Carpigiani> relativa alle organizzazioni presenti nel canale. Per individuare la terna corretta, il processo Worker interroga un database contenente due diverse tabelle. La prima associa l'identificativo della macchina con i valori della terna <ID_Cliente, ID_Produttore, ID_Carpigiani>; la seconda associa la terna individuata in precedenza con il nome dello specifico canale e altri parametri indispensabili per l'invio della proposta di transazione sul canale della rete blockchain. Successivamente il Worker invoca il Client Fabric;
5. il Client Fabric, per accedere alla Rete Fabric, effettua l'autenticazione attraverso l'uso del certificato presso uno specifico peer afferente all'organizzazione Carpigiani e invia la proposta di transazione che dovrà essere approvata da tutte le organizzazioni presenti nel canale (fase di Endorsement). Successivamente attende le risposte degli endorser verificando che siano positive una volta ricevute;
6. ricevute le approvazioni da parte dei peer, il Client Fabric inserisce quest'ultime e la proposta in un messaggio di transazione, che viene poi inviato in broadcast al servizio di ordering affinché venga inserito nel blocco successivo (fase di Ordering);
7. al termine della fase di Ordering, ovvero quando il blocco viene scritto e certificato dagli orderer, questi lo inviano a tutti i peer appartenenti al canale, i quali si occuperanno di validare le transazioni presenti al suo interno (fase di Validation). Successivamente, la transazione contenente il refill originario viene registrata nel ledger (fase di Commit), entrando così a far parte del ledger della Blockchain;

⁴ RabbitMQ, Remote Procedure Call: <https://www.rabbitmq.com/tutorials/tutorial-one-javascript.html>

8. il peer a cui il client era autenticato notifica al Client Fabric il risultato dell'inserimento della transazione (successo o fallimento);
9. il Client Fabric notifica il risultato dell'operazione di approvazione della transazione al processo Worker, il quale è incaricato di inserire un messaggio con il risultato della richiesta di transazione all'interno della coda riservata indicatagli dal Client ICM;
10. il Client ICM, in attesa di risposta, riceve il messaggio che specifica se l'operazione di refill è stata inserita correttamente all'interno della Blockchain.

La fase di Ordering, indicata nel punto 6 dei passaggi precedentemente illustrati, può essere implementata secondo una delle tre modalità discusse nel documento D3.1. Al fine di avere una politica di consenso decentralizzata e con un elevato livello di resilienza, è stata scelta la tipologia di orderer **Raft**. Quest'ultimo, inoltre, permette di suddividere e distribuire differenti gruppi di orderer nella rete, migliorando la gestione e la velocità della piattaforma. I nodi orderer, che partecipano a tutte le operazioni sulla rete, possono ricevere molteplici transazioni inerenti a canali differenti e, considerando una media di due o tre refill effettuati al giorno, si è scelto di utilizzare cinque nodi per la gestione dell'intera rete; questo valore potrebbe aumentare in seguito in caso il carico della rete aumentasse. Riguardo la scelta del numero di orderer iniziali, si è considerato il trade-off, per la creazione dei blocchi, tra **resilienza**⁵ e **velocità**⁶.

Per la scelta del numero di peer appartenenti a ogni organizzazione, sono state fatte delle considerazioni sui seguenti temi:

- **Sicurezza.** Più committer partecipano al canale, più nodi detengono il ledger; questo aumenta la resistenza alla manomissione della Blockchain e, di conseguenza, la sicurezza di tutta l'infrastruttura.
- **Efficienza.** Un attore che partecipa a molti canali necessita diversi endorser e committer distribuiti sulla rete; in questo modo si suddivide il carico di lavoro globale dell'organizzazione, aumentando le prestazioni in termini di resilienza e di velocità nella fase di Endorsement.
- **Flooding dei messaggi.** È necessario non superare un certo numero di peer, poiché l'elevato numero di messaggi scambiati potrebbe portare alla congestione della rete.

3.1. Funzionalità chaincode

Sulla base di quanto discusso nell'allegato D3.1, in questa sezione si approfondirà la parte implementativa inerente al chaincode. Come anticipato, quest'ultimo è lo strumento che

⁵ **resilienza:** più orderer sono presenti e più la rete è resistente e pronta a situazioni di carico intenso.

⁶ **velocità:** più orderer partecipano al voto, più scambi di messaggi dovranno essere computati prima di raggiungere il quorum.

permette di memorizzare e recuperare le informazioni nella blockchain. In generale, si occupa di gestire la logica aziendale concordata dai membri della rete e lo stato del registro, precedentemente inizializzato, attraverso le transazioni inviate dalle applicazioni.

Il sistema implementato ha l'obiettivo di mantenere in memoria tutta la blockchain contenente le informazioni relative ai refill delle macchine da gelato. Ciò permette:

- il non ripudio delle operazioni compiute;
- la visualizzazione delle transazioni:
 - da parte degli utenti per controllare la quantità di miscela da gelato usata;
 - da parte di Carpigiani per accertarsi che non vi siano frodi.

Il chaincode realizzato per questo specifico caso d'uso rispetta quello definito in fase di progettazione dichiarato nell'allegato tecnico D3.1.

Difatti, inizialmente il codice verifica che siano stati inseriti tutti i parametri richiesti e, in caso contrario, termina l'esecuzione rilanciando un errore. A differenza di quanto detto in D3.1, si è deciso di non sviluppare il controllo dell'identità di chi ha invocato la proposta di transazione a livello di chaincode, bensì questo viene gestito a livello di configurazione della rete. Questo approccio rende più sicuro il controllo dei permessi poiché non basta andare a modificare il chaincode in tutti gli endorser, bensì modificare l'intera blockchain in tutti i peer (endorser e committer).

Se l'utente che ha invocato la proposta di transazione risulta autorizzato, il chaincode recupera le informazioni richieste, ovvero: l'identificatore della ricarica tramite il codice a barre, la quantità di prodotto di ricarica utilizzato, il timestamp di scansione del codice a barre e dell'operazione di refill, il numero di coni erogati fino a quel momento e l'identificativo del contratto stipulato.

Successivamente, il peer che esegue il chaincode verifica che la richiesta di registrazione del refill sia effettivamente una nuova transazione e che questa rispetti tutti i vincoli imposti a livello di contratto. Infine, aggiorna i parametri inerenti alla macchina e ne salva lo stato.

3.2. Funzionalità applicazione Web

Per quanto riguarda l'applicazione Web, si è deciso di svilupparla utilizzando il framework **Ruby on Rails**, implementando le funzionalità che sono state discusse nel D3.1.

È consentito l'accesso all'applicazione solo previo login con le proprie credenziali, inoltre, è stato stabilito che l'organizzazione Carpigiani possa essere la sola ad avere utenti con i massimi privilegi (utente amministratore).

Nella Figura 3.2.1 è mostrata la schermata relativa al login. Una volta inserite le credenziali, si accede alla schermata principale che cambia in base al tipo di utente loggato.

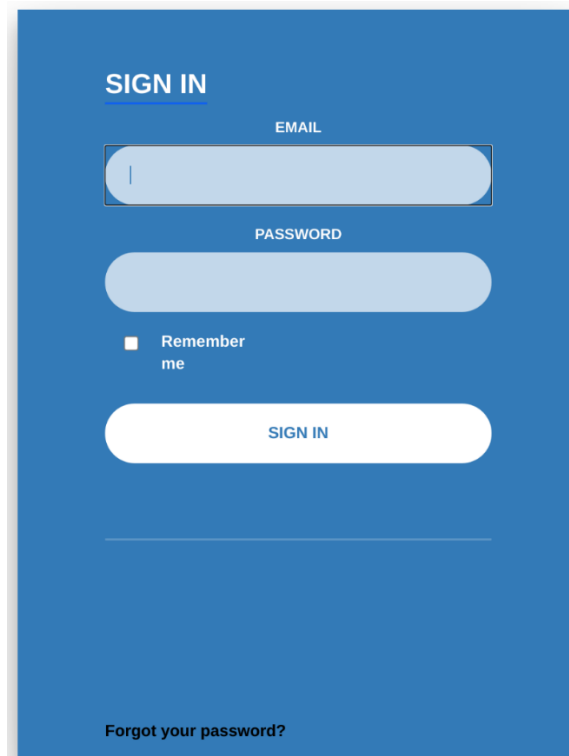


Figura 3.2.1 - Schermata di login.

Sono stati sviluppati due profili di utenti per l'utilizzo dell'applicazione Web: **utenti standard** e **amministratori** (utenti con maggiori privilegi rispetto agli utenti standard).

La funzionalità comuni a entrambe le tipologie di utenti sono:

- **Refills.** Consente la visualizzazione dei refill afferenti alle proprie macchine (caso utenti standard) o relative a tutte le macchine (caso utenti amministratori).
- **Profile.** Permette la modifica delle informazioni relative all'utente loggato.
- **Machines.** Mostra l'elenco delle macchine associate all'organizzazione dell'utente loggato (caso utente standard) o di tutte le macchine (nel caso di utente amministratore).

In aggiunta alle funzionalità presentate, l'utente amministratore può svolgere anche le seguenti funzionalità:

- **New User.** Inserire un nuovo utente.
- **New Machine.** Inserire una nuova macchina.
- **Blockchain.** Entrare nella sezione riservata alla modifica della blockchain.

Nella Figura 3.2.2 sono presentati i menu principali dell'interfaccia per l'utente standard (a sinistra) e per l'utente amministratore (a destra).

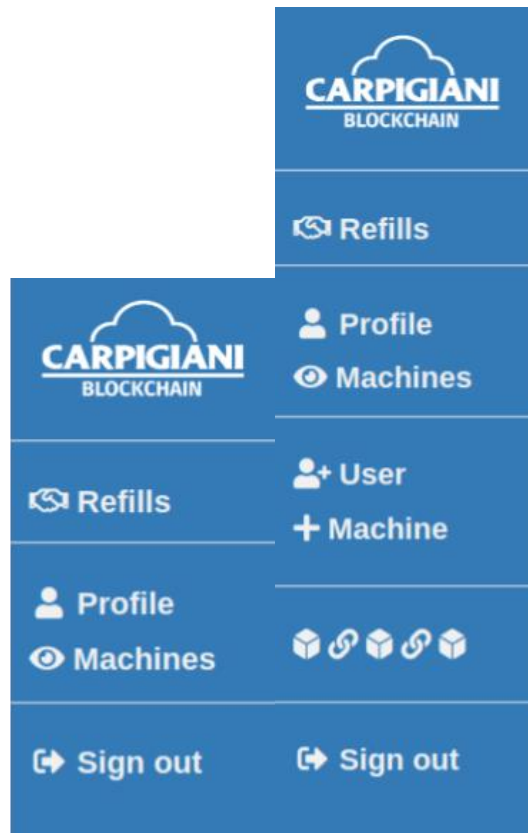


Figura 3.2.2 - Menu principali dell'interfaccia per l'utente standard (sx) e per l'amministratore (dx).

Una volta eseguito il login, un utente standard può visualizzare i canali nei quali la propria organizzazione risulta registrata (nella sezione Agreements), come mostrato in Figura 3.2.3.

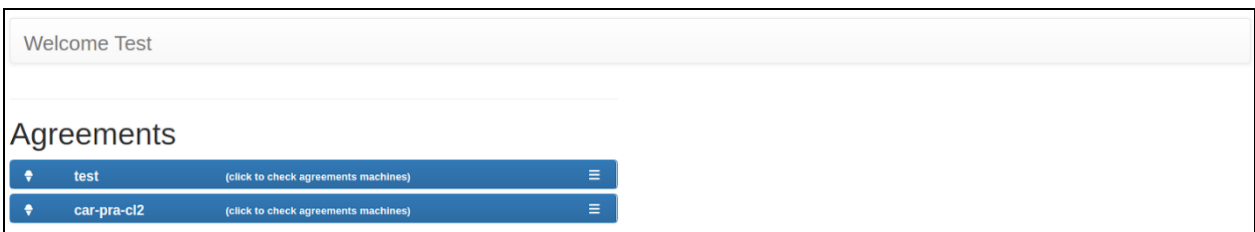


Figura 3.2.3 - Schermata principale di utente standard che mostra i canali della propria organizzazione.

Se l'utente a effettuare il login invece è un utente amministratore, questo può visualizzare l'elenco di tutti i canali presenti nel sistema (nella sezione Agreements) e l'elenco degli utenti registrati (sezione Users), come evidenziato nella Figura 3.2.4.

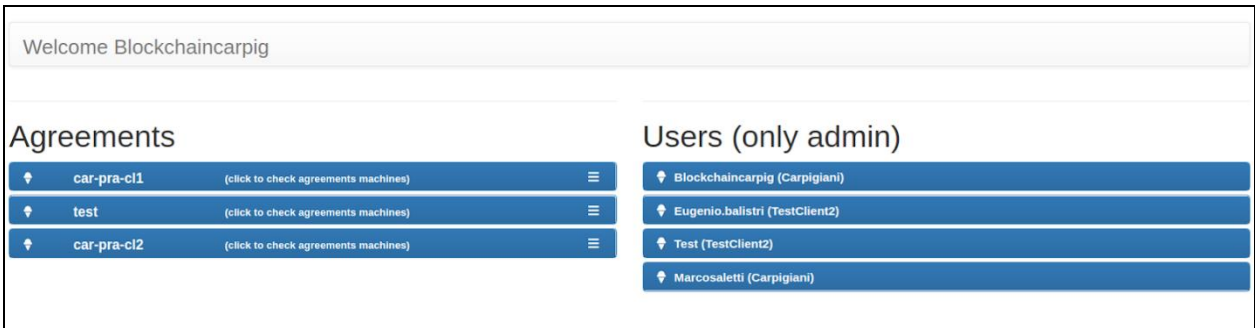


Figura 3.2.4 - Schermata principale di utente amministratore che mostra tutti i canali e gli utenti presenti nel sistema.

Accedendo al menu “Refills”, qualsiasi tipologia di utente può visionare i refill relativi alla macchina desiderata impostando correttamente l’intervallo temporale della ricerca e il nome della macchina. Un esempio di questa schermata è mostrato nella Figura 3.2.5.

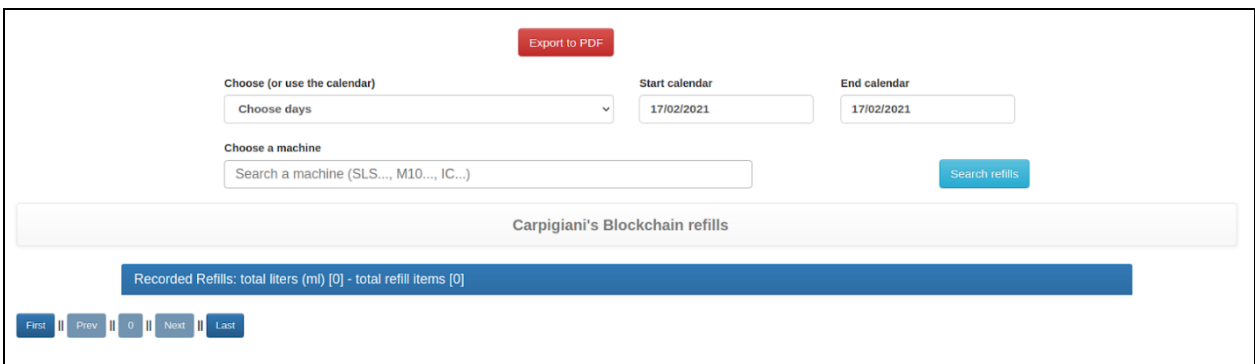


Figura 3.2.5 - Schermata principale per la visualizzazione dei refill di una macchina.

I dati relativi a un refill, invece, vengono mostrati in una sezione apposita come quella evidenziata in Figura 3.2.6, dove vengono visualizzati i campi principali:

- **Machine Refill.** Identifica la macchina che ha effettuato il refill.
- **Contract ID.** Si riferisce allo smart contract utilizzato.
- **Refill Quantity.** Indica la quantità di prodotto inserito nelle macchine.
- **Refill Timestamp.** Campo relativo all’istante nel quale avviene l’operazione di refill.
- **Barcode.** Identifica il codice a barre presente nella scatola di miscela da gelato.
- **Barcode Timestamp.** Si riferisce all’istante temporale nel quale viene letto il codice a barre.
- **No. of cones.** Indica il numero di coni gelato realizzati dalla macchina fino a quel momento.

La funzionalità che permette la visualizzazione di tutte le macchine possedute dall’utente loggato è stata poi implementata e, come impostazione di default, Carpigiani possiede tutte le macchine presenti a sistema. Questa funzione crea un’unica vista che risulta utile per il monitoraggio e la visualizzazione delle informazioni delle macchine da gelato. La Figura 3.2.7 mostra la schermata appena descritta.

Carpigiani's Blockchain refills

Recorded Refills: total liters (ml) [4000] - total refill items [2]

- Machine refill: SoftServe-M1000001
- Contract ID: contract_v1
- Refill quantity (ml): 2000
- Refill timestamp: 25 Jun 2020 17:03:32
- Barcode: Unknown
- Barcode timestamp: 25 Jun 2020 17:03:32
- No of cones: 0

- Machine refill: SoftServe-M1000001
- Contract ID: contract_v1
- Refill quantity (ml): 2000
- Refill timestamp: 25 Jun 2020 17:03:31
- Barcode: Unknown
- Barcode timestamp: 25 Jun 2020 17:03:31
- No of cones: 0



Figura 3.2.6 - Porzione di schermata dettagliata inerente ai refill della macchina "M1000001".

Machines			
M1000001 (Client1)	refills	info	Fabric: M1000001
M1000002 (Client1)	refills	info	Fabric: M1000002
test (TestClient2)	refills	info	Fabric: test

Figura 3.2.7 - Schermata di visualizzazione di tutte le macchine possedute dall'utente Carpigiani

Passando alla visualizzazione dell'interfaccia Blockchain, cioè quella che consente la gestione della rete Fabric, nella Figura 3.2.8 viene presentato il menù relativo alla sezione di modifica della blockchain.

Le funzionalità previste dal menu di modifica della blockchain sono:

- **New channel.** Per la creazione di un nuovo canale.
- **New org.** Per la creazione di una nuova organizzazione.
- **Add org.** Per inserire una organizzazione già esistente in un canale.
- **Install SC.** Per l'installazione di un nuovo Smart Contract (chaincode) su un canale.

La Figura 3.2.9 mostra la home page della sezione di gestione della blockchain. A questa si accede unicamente con l'account di tipo amministratore e si visualizzano le informazioni relative alle organizzazioni che compongono un singolo canale; come ad esempio: il nome dell'Endpoint per ogni organizzazione, la lunghezza del Ledger e il nome dello Smart Contract installato.

Nella figura 3.2.10 viene rappresentata l'interfaccia atta alla creazione di un nuovo canale, sia a livello applicativo, che nella rete Fabric. Risulta quindi necessario inserire nel form i dati inerenti al peer dell'organizzazione che ne farà parte e al nome del canale stesso.

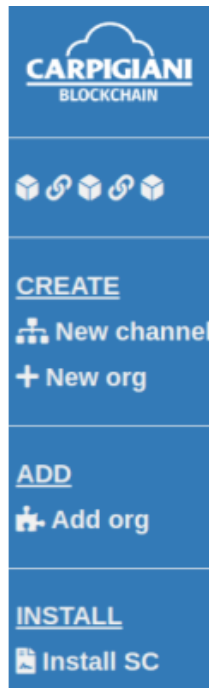


Figura 3.2.8 - Menu relativo alla sezione di modifica della blockchain.

Channels

Choose a channel

MSPID: CarpigianiMSP

- Endpoint: peer0.carpigiani.carpinet.com:7051
- LedgerHeight: 995
- Chaincodes:
 - refill
 - _lifecycle

MSPID: Client1MSP

- Endpoint: peer0.client1.carpinet.com:10051
- LedgerHeight: 995
- Chaincodes:
 - refill
 - _lifecycle

MSPID: ProdAMSP

- Endpoint: peer0.prodA.carpinet.com:9051
- LedgerHeight: 995
- Chaincodes:
 - refill
 - _lifecycle

Figura 3.2.9 - Schermata principale del canale selezionato.

NEW CHANNEL

CHANNEL NAME

PEER

peer0

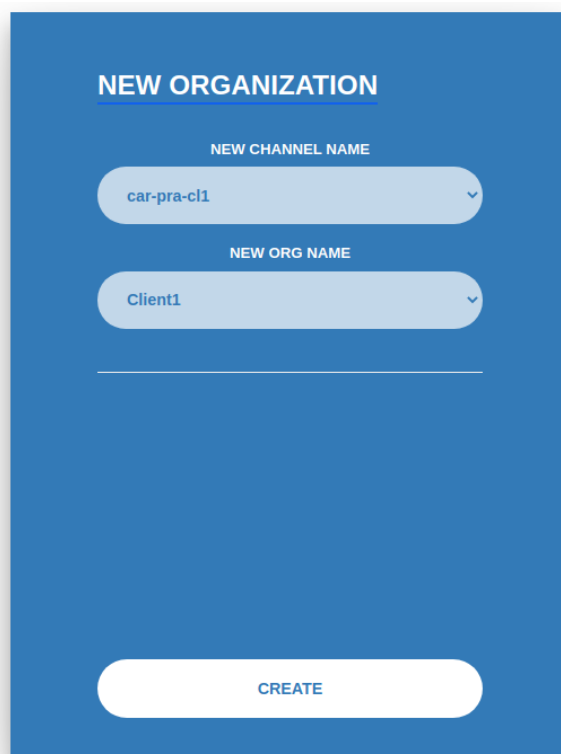
PORT

7051

CREATE

Figura 3.2.10 - Schermata di creazione di un nuovo canale.

Come mostrato in Figura 3.2.11, l'amministratore ha anche la possibilità di inserire un'organizzazione, già esistente nella piattaforma, in un nuovo canale sia a livello applicativo che a quello della rete Fabric.



The screenshot shows a form titled "NEW ORGANIZATION" on a blue background. It features two dropdown menus. The first dropdown, labeled "NEW CHANNEL NAME", has "car-pra-cl1" selected. The second dropdown, labeled "NEW ORG NAME", has "Client1" selected. Below the dropdowns is a horizontal line, and at the bottom of the form is a white "CREATE" button.

Figura 3.2.11 - Schermata di aggiunta della organizzazione "Client1" nel canale "car-pra-cl1".

Come già anticipato, è stata implementata anche la funzionalità inerente la creazione di una nuova organizzazione, in cui vengono specificate lato applicazione non solo le caratteristiche inerenti all'organizzazione, ma anche altre informazioni riguardanti il canale in cui essa dovrà essere inserita e i relativi peer. La Figura 3.2.12 mostra il dettaglio della schermata relativa a questa funzionalità.

Infine, l'ultima funzionalità a livello di gestione blockchain consiste nell'installazione dello smart contract nel canale selezionato (Figura 3.2.13). Questo genere di operazione è importante perché permette all'amministratore di creare un nuovo chaincode o di aggiornare quello attuale, se il contratto con il cliente e il produttore dovesse subire delle variazioni.

NEW ORGANIZATION

NEW ORG NAME	ANCHOR PEER (CARPIGIANI)
<input type="text"/>	<input type="text" value="peer0"/>
ORG VAT	HOW MANY PEERS
<input type="text"/>	<input type="text" value="1"/>
ORG HEAD OFFICE	WHAT START FROM?
<input type="text"/>	<input type="text" value="0"/>
CHANNEL NAME	HOW MANY USERS?
<input type="text" value="Please select a channel"/>	<input type="text" value="1"/>
NEW PEER PORT	ORGANIZATION CA PORT
<input type="text"/>	<input type="text"/>

Figura 3.2.12 - Schermata di creazione di una nuova organizzazione.

INSTALL CHAINCODE

CHANNEL NAME

CHAINCODE

VERSION

Figura 3.2.13 - Schermata di installazione di una nuova versione del chaincode "refill" nel canale "car-pra-cl1".