



# FILIERA-TOKEN-SHOP

Umberto Della Monica  
Gerardo Leone





# TABELLA DEI CONTENUTI

**01**

**INTRODUZIONE**

**02**

**ARCHITETTURA**

**03**

**SMART  
CONTRACTS**

**04**

**SICUREZZA**

**05**

**FUNZIONALITA'  
PRINCIPALI**

**06**

**CONCLUSIONI**





# 01

## INTRODUZIONE



# LA FILIERA



# LA FILIERA: PARTITA DI LATTE



- Identificativo
- Scadenza
- Quantità (in litri)
- Prezzo (FLT)

# LA FILIERA: BLOCCO DI FORMAGGIO



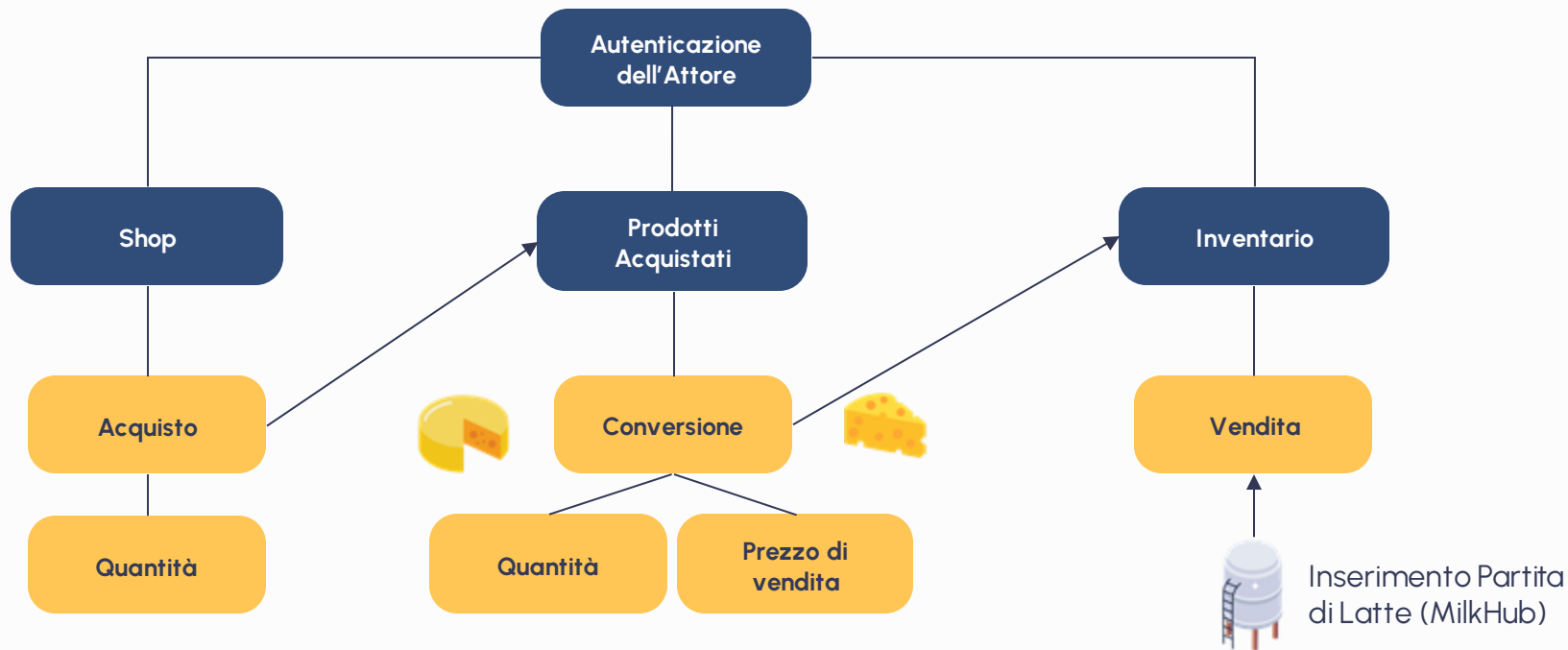
- Identificativo
- Marchio di qualità D.O.P.
- Quantità (in Kg)
- Prezzo (FLT)

# LA FILIERA: PEZZO DI FORMAGGIO



- Identificativo
- Quantità (in Kg)
- Prezzo (FLT)

# NAVIGAZIONE E INTERAZIONE





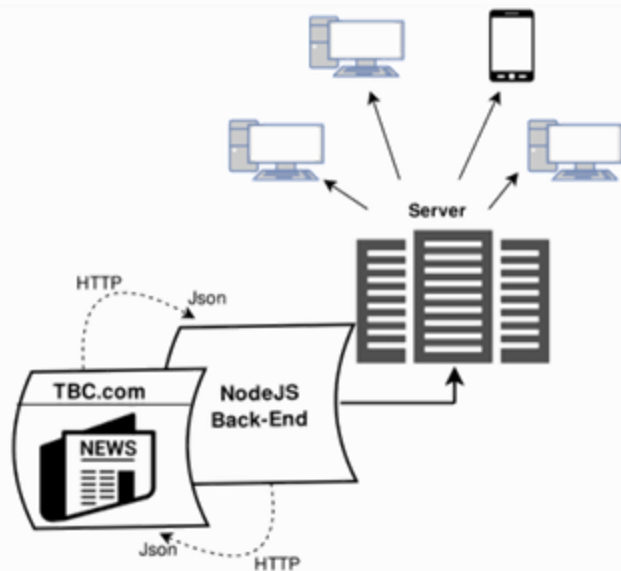


# 02

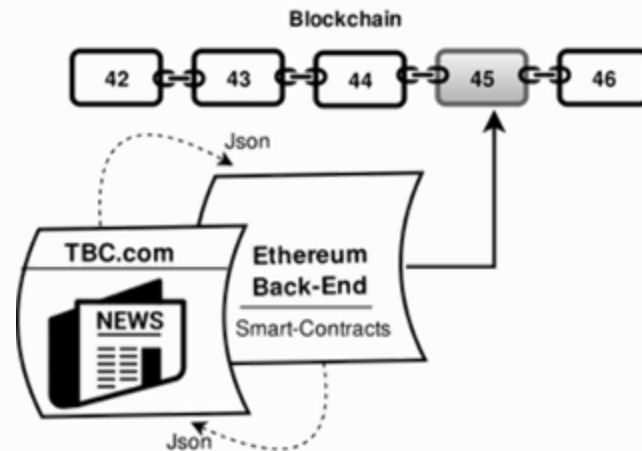
## ARCHITETTURA



# WEB 2.0 vs WEB 3.0

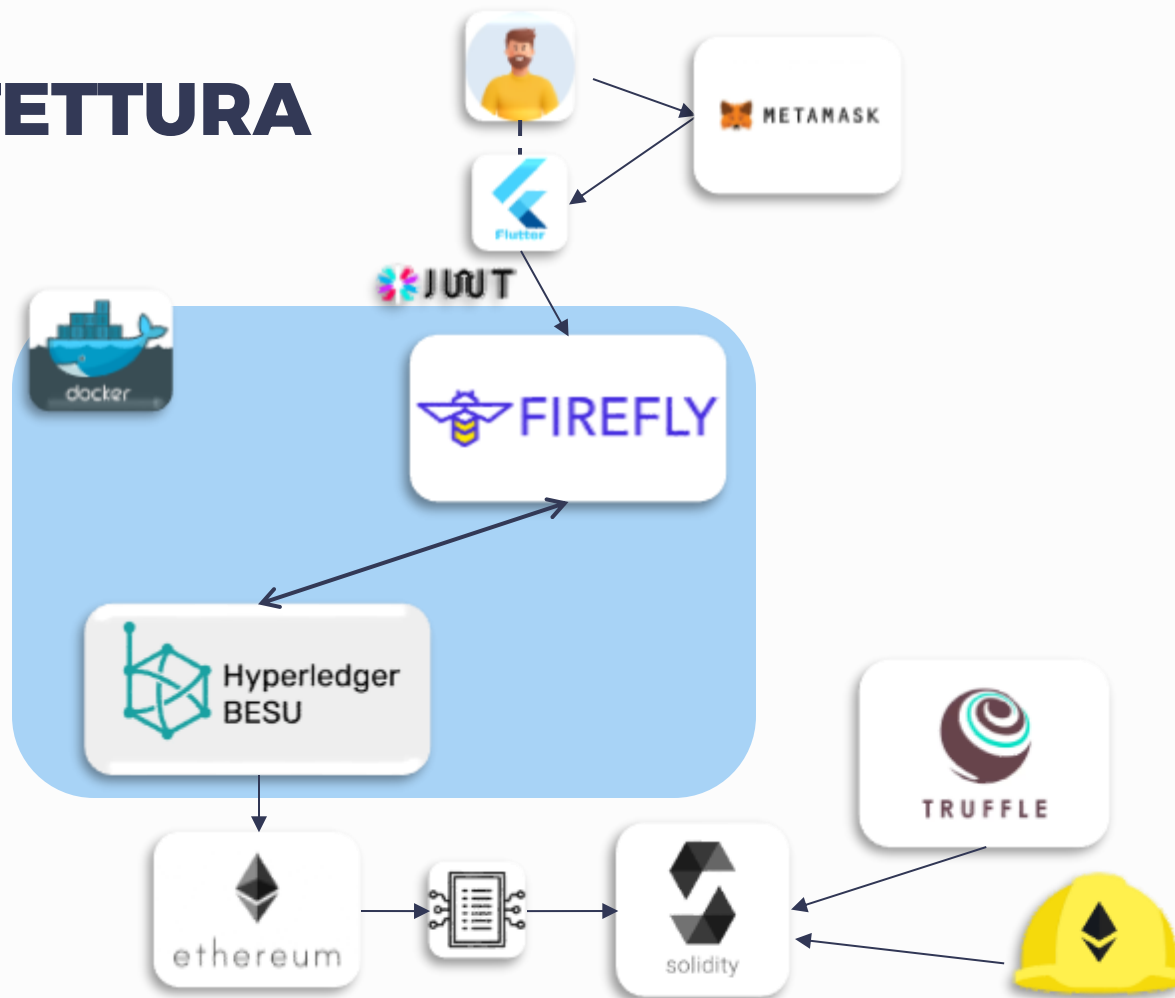


**Web 2.0**  
**Architettura centralizzata**

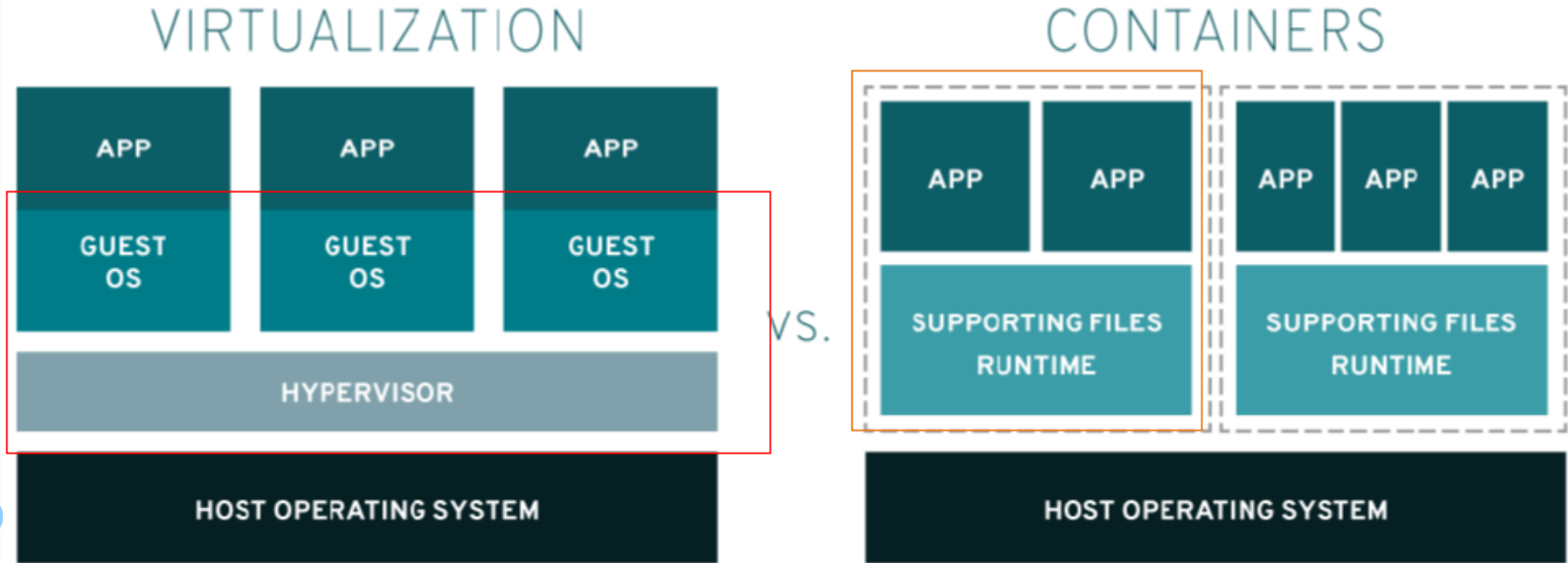


**Web 3.0**  
**Architettura decentralizzata**

# ARCHITETTURA



# DOCKER: UN AMBIENTE SICURO



# DOCKER

## Client

Docker pull

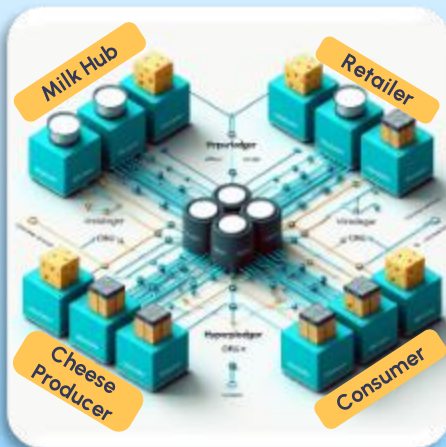
Docker build

Docker run

## Docker Host



### Docker daemon



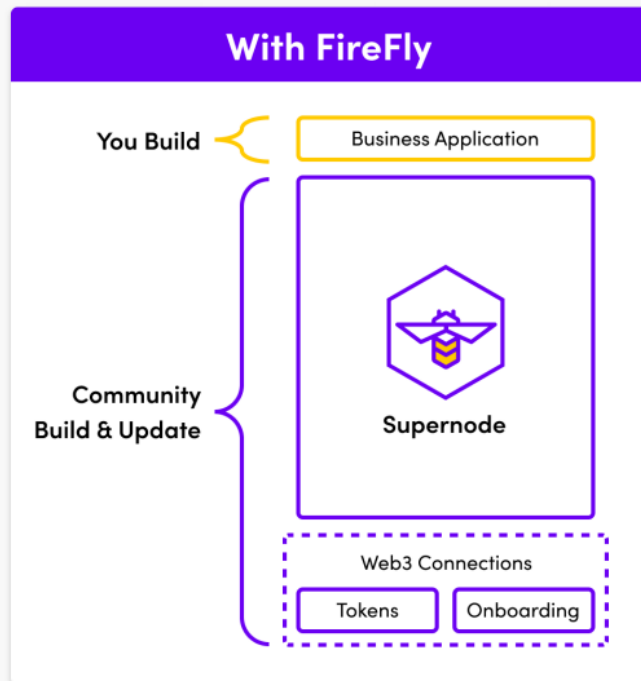
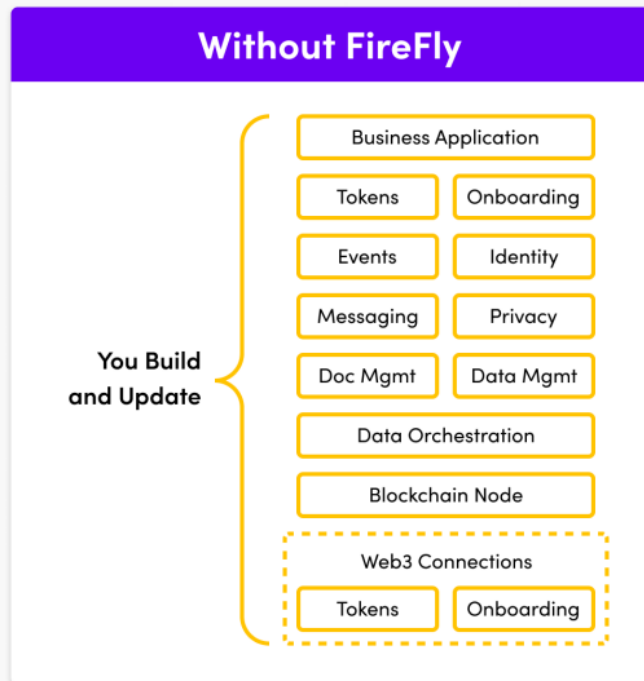
## Registry



```
ff init ethereumfiliera-token4 --block-period 2 --blockchain-connector "evmconnect" --blockchain-node "remote-rpc" --chain-id 1337 --contract-address "0xb9A219631Aed55eBC3D998f17C3840B7eC39C0cc" --remote-node-url "http://host.docker.internal:8545/" --org-name MilkHub_Org --node-name MilkHub_Node --org-name CheeseProducer_Org --node-name CheeseProducer_Name --org-name Retailer_Org --node-name Retailer_Node --org-name Consumer_Org --node-name Consumer_Node
```



# HYPERLEDGER FIREFLY



# HYPERLEDGER FIREFLY: API

The screenshot shows the Hyperledger FireFly API interface in a web browser. The browser's address bar displays the URL `127.0.0.1:5109/home?action=contracts.interface`. The interface is divided into several sections:

- Messages:** A list of messages with columns for the message ID and actions (download and copy).
- Tokens:** A section for managing tokens.
- Contracts:** A section for managing contracts, including a "Deploy a Contract" button and a "Define a Contract Interface" section. The "Define a Contract Interface" section includes a dropdown for "Interface Format" (set to "ABI - Solidity Application Binary Interface"), a text input for "Name", and a text input for "Version". Below these is a "Schema" section with a JSON schema for an ABI.
- Code:** A section for writing and running code. It includes a "Run" button and a "Server Response" section.
- Events:** A section for viewing events submitted to FireFly.

The interface is dark-themed and includes a sidebar on the right with various icons for navigation. The top of the interface shows the Hyperledger FireFly logo and the word "Sandbox".

# HYPERLEDGER FIREFLY: API

The screenshot shows the Hyperledger FireFly API interface in a web browser. The browser's address bar displays the URL `127.0.0.1:5109/home?action=contracts.interface`. The interface is divided into several sections:

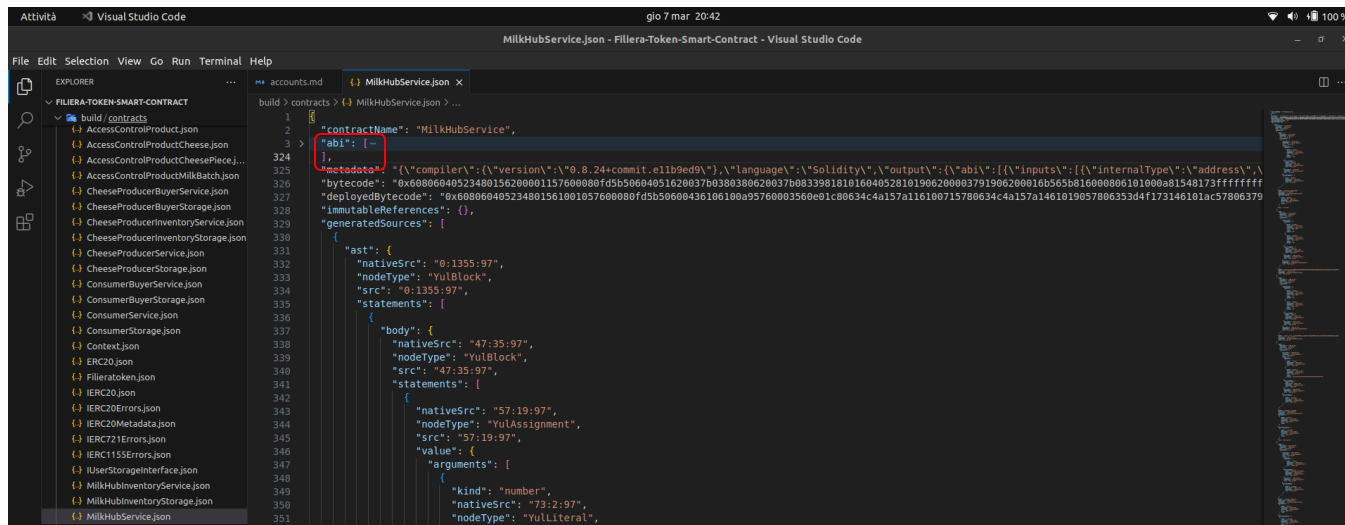
- Messages, Tokens, Contracts:** A top navigation bar with tabs for Messages, Tokens, and Contracts. The Contracts tab is currently selected.
- Contracts List:** A list of contracts with columns for Name, ABI, and Actions. Two contracts are visible: `/api/contracts/api/MilkHubInventoryService` and `/api/contracts/api/MilkHubService`.
- Deploy a Contract:** A button to "Deploy a Contract" with the description "Compile and deploy your contract to the blockchain".
- Define a Contract Interface:** A section highlighted with a red box, containing a form to define a contract interface. It includes a dropdown for "Interface Format" (set to "ABI - Solidity Application Binary Interface"), a note "Either FFI or Solidity ABI format", and input fields for "Name" and "Version". Below these is a "Schema" field with a JSON schema for an anonymous function.
- Code:** A section titled "Application Code" with a description: "Build an action using the panels on the left. Click 'Run' to execute the action via the backend server. FireFly events related to the transaction will appear on the right." It contains a code editor with the following JavaScript code:

```
1 const ffi = await firefly.generateContractInterface({
2   name: undefined,
3   version: undefined,
4   input: {
5     abi: [{"anonymous": true}],
6   },
7 });
8 const result = await firefly.createContractInterface(ffi);
9 return { type: 'message', id: result.message_id };
```

A "Run" button is located at the bottom right of the code editor.
- Server Response:** A section titled "Server Response" with a text area containing the response: `{}`.
- Events:** A section titled "Events" with a description: "Events submitted to FireFly will show here."



# HYPERLEDGER FIREFLY: API



The screenshot displays the Visual Studio Code editor with the 'MilkHubService.json' file open. The Explorer sidebar on the left shows the project structure under 'FILIERA-TOKEN-SMART-CONTRACT', with 'contracts' expanded. The main editor area shows the JSON content of the file, with the 'abi' field highlighted by a red box. The JSON includes contract metadata, bytecode, and a list of ABI functions.

```
{
  "contractName": "MilkHubService",
  "abi": [
    {
      "name": "constructor",
      "inputs": [
        {
          "name": "contractAddress",
          "type": "address"
        }
      ],
      "outputs": [],
      "stateMutability": "nonpayable"
    },
    {
      "name": "getContractAddress",
      "inputs": [],
      "outputs": [
        {
          "name": "contractAddress",
          "type": "address"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "getContractName",
      "inputs": [],
      "outputs": [
        {
          "name": "contractName",
          "type": "string"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "getContractBytecode",
      "inputs": [],
      "outputs": [
        {
          "name": "contractBytecode",
          "type": "bytes"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "getContractMetadata",
      "inputs": [],
      "outputs": [
        {
          "name": "contractMetadata",
          "type": "string"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "getContractImmutableReferences",
      "inputs": [],
      "outputs": [
        {
          "name": "contractImmutableReferences",
          "type": "string"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "getGeneratedSources",
      "inputs": [],
      "outputs": [
        {
          "name": "contractGeneratedSources",
          "type": "string"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "ast",
      "inputs": [
        {
          "name": "nativeSrc",
          "type": "string"
        },
        {
          "name": "nodeType",
          "type": "string"
        },
        {
          "name": "src",
          "type": "string"
        },
        {
          "name": "statements",
          "type": "array"
        }
      ],
      "outputs": [
        {
          "name": "body",
          "type": "object"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "nativeSrc",
      "inputs": [
        {
          "name": "nativeSrc",
          "type": "string"
        },
        {
          "name": "nodeType",
          "type": "string"
        },
        {
          "name": "src",
          "type": "string"
        },
        {
          "name": "value",
          "type": "object"
        }
      ],
      "outputs": [
        {
          "name": "arguments",
          "type": "array"
        }
      ],
      "stateMutability": "view"
    },
    {
      "name": "kind",
      "inputs": [
        {
          "name": "kind",
          "type": "string"
        },
        {
          "name": "nativeSrc",
          "type": "string"
        },
        {
          "name": "nodeType",
          "type": "string"
        }
      ],
      "outputs": [
        {
          "name": "value",
          "type": "string"
        }
      ],
      "stateMutability": "view"
    }
  ],
  "bytecode": "0x60806040523480156200001157600080fd5b50604051620037b083398181016040520101906200003791906200016b505b616000806101000a81548173ffffffff",
  "deployedBytecode": "0x60806040523480156200001157600080fd5b50604051620037b083398181016040520101906200003791906200016b505b616000806101000a81548173ffffffff",
  "immutableReferences": {},
  "generatedSources": [
    {
      "ast": {
        "nativeSrc": "0:1355:97",
        "nodeType": "YulBlock",
        "src": "0:1355:97",
        "statements": [
          {
            "body": {
              "nativeSrc": "47:35:97",
              "nodeType": "YulBlock",
              "src": "47:35:97",
              "statements": [
                {
                  "nativeSrc": "57:19:97",
                  "nodeType": "YulAssignment",
                  "src": "57:19:97",
                  "value": {
                    "arguments": [
                      {
                        "kind": "number",
                        "nativeSrc": "73:2:97",
                        "nodeType": "YulLiteral",

```

# HYPERLEDGER FIREFLY: API

The screenshot shows the Hyperledger Firefly API interface in a web browser. The browser's address bar displays the URL `127.0.0.1:5109/home?action=contracts.api`. The interface is divided into several sections:

- Messages:** A list of messages, with two entries highlighted by a red box: `/api/contracts/api/MilkHubInventoryService` and `/api/contracts/api/MilkHubService`.
- Contracts:** A section containing several actions:
  - Deploy a Contract:** Compile and deploy your contract to the blockchain.
  - Define a Contract Interface:** Defines all methods and events of your contract to FireFly.
  - Register a Contract API:** Generates an interactive HTTP API from your contract. This section is highlighted by a red box and contains a dropdown menu for "Contract Interface" (set to "TransactionBuyCheesePieceServiceInterface - 1.0"), a "Name" field, and an "Address" field.
  - Register a Contract Listener:** Instructs FireFly to listen for and index events from your contract.
- Code:** A section for application code, showing a code editor with the following code:

```
1 const ffi = await firefly.generateContractInterface({
2   name: undefined,
3   version: undefined,
4   input: {
5     abi: [{"anonymo ... unction"}],
6   },
7 });
8 const result = await firefly.createContractInterface(ffi);
9 return { type: 'message', id: result.message };
```
- Events:** A section for events, showing a message: "Events submitted to FireFly will show here."

The interface also includes a top navigation bar with "Messages", "Tokens", and "Contracts" tabs, and a right sidebar with various icons and a "Connected" status indicator.

# SWAGGER

Attività Microsoft Edge gio 7 mar 20:38

MilkHub FireFly Explorer FireFly Sandbox Swagger UI

127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService/api

Aggiungi i preferiti a questa area di lavoro posizionandoli sulla barra Preferiti. [Gestire i preferiti ora](#)

Swagger OPENAPI /api/v1/namespaces/default/apis/MilkHubService/api/swagger.yaml **Explore**

**MilkHubServiceInterface** 1.0 **OAS3**  
[/api/v1/namespaces/default/apis/MilkHubService/api/swagger.yaml](#)

Servers  
<http://127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService>

- GET /interface
- POST /invoke/deleteMilkHub
- POST /invoke/getListAddressMilkHub
- POST /invoke/getMilkHubBalance
- POST /invoke/getMilkHubData
- POST /invoke/getMilkHubEmail
- POST /invoke/getMilkHubFullName
- POST /invoke/getMilkHubId
- POST /invoke/isUserPresent
- POST /invoke/login
- POST /invoke/registerMilkHub
- POST /invoke/updateMilkHubBalance

# SWAGGER

Attività Microsoft Edge gio 7 mar 20:39

MilkHub FireFly Explorer FireFly Sandbox Swagger UI

127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService/api#/invoke\_registerMilkHub

Aggiungi i preferiti a questa area di lavoro posizionandoli sulla barra Preferiti. [Gestire i preferiti ora](#)

## POST /invoke/registerMilkHub

Additional smart contract details:

Key	Key
stateMutability	nonpayable

### Parameters

Name	Description
Request-Timeout	Server-side request timeout (milliseconds, or set a custom suffix like 10s)
string (header)	Default value : 2m0s

2m0s

Request body application/json

Example Value Schema

```
{  "idempotencyKey": "string",  "input": {    "email": "string",    "fullName": "string",    "password": "string",    "walletMilkHub": "string"  },  "key": "string",  "options": {}}
```

### Responses

Code	Description	Links
200	Success	No links

Media type application/json

# SWAGGER

Attività Microsoft Edge gio 7 mar 20:39

MilkHub FireFly Explorer FireFly Sandbox Swagger UI

127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService/api#/query\_getMilkHubId

Aggiungi i preferiti a questa area di lavoro posizionandoli sulla barra Preferiti. [Gestire i preferiti ora](#)

POST /query/getMilkHubBalance

POST /query/getMilkHubData

POST /query/getMilkHubEmail

POST /query/getMilkHubFullName

POST /query/getMilkHubId

Additional smart contract details:

Key	Key
stateMutability	view

Parameters Try it out

Name	Description
Request-Timeout string (header)	Server-side request timeout (milliseconds, or set a custom suffix like 10s) Default value : 2m0s <input type="text" value="2m0s"/>

Request body application/json

Example Value | Schema

```
{  "idempotencyKey": "string",  "input": {    "walletMilkHub": "string"  },  "key": "string",  "options": {}}
```

Responses

# HYPERLEDGER BESU

Client Ethereum 



## Reti pubbliche

Sono accessibili a chiunque e non richiedono permessi per partecipare. Chiunque può leggere, inviare e confermare transazioni.



## Reti private

Una specifica organizzazione definisce gli utenti autorizzati ad accedere alla rete e a partecipare alle attività di lettura e scrittura.

# HYPERLEDGER BESU

Client Ethereum 



## Reti pubbliche

Sono accessibili a chiunque e non richiedono permessi per partecipare. Chiunque può leggere, inviare e confermare transazioni.

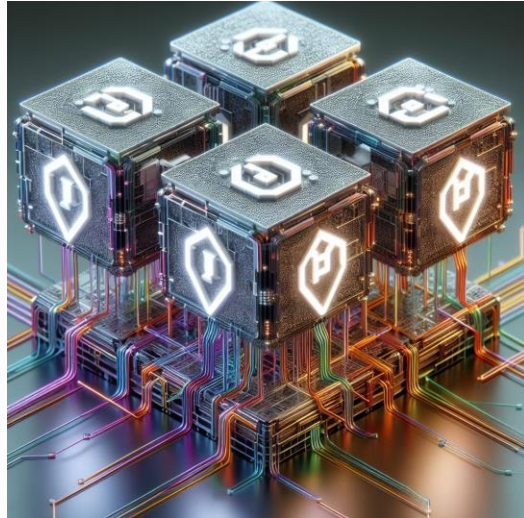
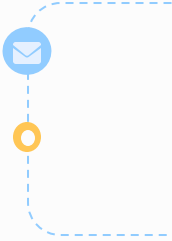


## Reti private

Una specifica organizzazione definisce gli utenti autorizzati ad accedere alla rete e a partecipare alle attività di lettura e scrittura.

# HYPERLEDGER BESU: QBFT

Client Ethereum 



- **QBFT sta per Qorum Byzantine Fault Tolerance**

- **Principali Caratteristiche :**

- **1. Tolleranza ai guasti**
- **2. Finalità Rapida**
- **3. Equilibrio tra Velocità di Consenso e Sicurezza**

- **Applicazioni :**

- **1. Eseguito all'interno delle Reti Private**
- **2. Adatto per Applicazioni che richiedono Affidabilità e Velocità**





# HYPERLEDGER BESU: QBFT

## Docker Build-up



```
1 {
2   "genesis": {
3     "config": {
4       "chainId": 1337,
5       "berlinBlock": 0,
6       "qbft": {
7         "blockperiodseconds": 2,
8         "epochlength": 30000,
9         "requesttimeoutseconds": 4
10      },
11     },
12     "nonce": "0x0",
13     "timestamp": "0x58ee40ba",
14     "gasLimit": "0x1fffffffffffff",
15     "difficulty": "0x1",
16     "mixhash": "0x63746963616c2062797a616e74696e65206661756c742074666572616e6365",
17     "coinbase": "0x00000000000000000000000000000000",
18     "alloc": {
19       "fe3b557e8fb62b89f4916b721be55ceb828dbd73": {
20         "privateKey": "8fa55949038a9610f50fb23b5883af3b4ecb33b5792cbcfbd1542c692be63",
21         "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
22         "balance": "0xad778ebc5ac200000"
23       },
24       "627386690aba8346e1400e9345bc66c78a88ef57": {
25         "privateKey": "c87509a1c067bde78beb793e6fa7653bb61324c4241e5e4adeceba9f44dc0d3",
26         "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
27         "balance": "90000000000000000000000000000000"
28       },
29       "f17f52151E8F6C7334FAD088C5704D77216b732": {
30         "privateKey": "a6aeae5ccbf0b4590405997ee2d52d2b3387261370875053c36d94e974d162f",
31         "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
32         "balance": "90000000000000000000000000000000"
33       }
34     },
35   },
36   "blockchain": {
37     "nodes": {
38       "generate": true,
39       "count": 4
40     }
41   }
42 }
```

Config File

```
# Blockchain

bootnode:
  hostname: bootnode
  ports:
    - '8545:8545'
    - '30303'
    - '30303/udp'
  networks:
    chain_net:
      ipv4_address: 172.4.0.111
  image: 'hyperledger/besu'
  user: "root:root"
  volumes:
    - bootnode_data:/opt/besu/database
    - ./networkFiles/config:/etc/besu/config
    - ./networkFiles/keys/bootnode:/etc/besu/keys
  command:
    --data-path=/opt/besu/database
    --genesis-file=/etc/besu/config/genesis.json
    --node-private-key-file=/etc/besu/keys/key
    --min-gas-price=0
    --rpc-http-enabled
    --rpc-http-host=0.0.0.0
    --rpc-http-api=ETH,NET,QBFT,ADMIN,WEB3
    --host-allowlist=""
    --rpc-http-cors-origins="all"
    --logging=INFO
```

Docker Config File

# HYPERLEDGER BESU: QBFT

## Boot-Node URL



```
#####  
#  
# Besu version 24.1.0  
#  
# Configuration:  
# Network: Custom genesis file  
# /etc/besu/config/genesis.json  
# Network Id: 1337  
# Data storage: Forest  
# Sync mode: Full  
# RPC HTTP APIs: ETH_NET_QBFT_ADMIN_WEB3  
# RPC HTTP port: 8545  
# Using LAYERED transaction pool implementation  
# Using STACKED worldstate update mode  
#  
# Host:  
# Java: openjdk-java-17  
# Maximum heap size: 1.92 GB  
# OS: linux-x86_64  
# glibc: 2.35  
# jemalloc: 5.2.1-0-gea6b3e973b477b8061e0876bb257dbd7f3faa756  
# Total memory: 7.68 GB  
# CPU cores: 12  
#  
# Plugins:  
# TOTAL = 0 of 0 plugins successfully loaded  
# from /opt/besu/plugins  
#  
#####  
2024-03-06 18:39:14.939+00:00 | main | INFO | Besu | Security Module: localfile  
2024-03-06 18:39:15.161+00:00 | main | INFO | Besu | Using the native implementation of alt bn128  
2024-03-06 18:39:15.259+00:00 | main | INFO | Besu | Using the native implementation of modexp  
2024-03-06 18:39:15.260+00:00 | main | INFO | Besu | Using the native implementation of the signature algorithm  
2024-03-06 18:39:15.274+00:00 | main | INFO | Besu | Using the native implementation of the blake2bf algorithm
```

- Recuperiamo l'URL dell'enode.
- Lo settiamo come variabile di environment all'interno degli altri nodi che partecipano al meccanismo di consenso
- Avviamo il container di Besu per sincronizzare i VALIDATORI e cominciare a creare i blocchi all'interno della nostra rete
- Validatori disponibili tramite (Header - Smart Contract )

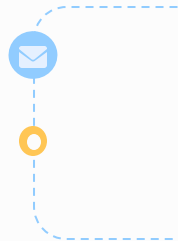
```
network > besu-private-qbft-network-docker > .env
```

```
# Configuration env vars
```

```
BOOTNODE_ID=ca960da0249d5a68fa943b4369e13b4ba2f15eb10500657ee417e4a9fc695597fa55a9bf2ed9bee3752c8685965a438a724e65c46473d8c0c0691d9176ee3a09
```

# HYPERLEDGER BESU: QBFT

## Setting-up



```
node2:
  hostname: node2
  ports:
    - '8547:8545'
    - '30303'
    - '30303/udp'
  networks:
    chain_net:
      ipv4_address: 172.4.0.202
  image: 'hyperledger/besu'
  user: "root:root"
  volumes:
    - node2_data:/opt/besu/database
    - ./networkFiles/config:/etc/besu/config
    - ./networkFiles/keys/node2:/etc/besu/keys
    - ./single-node/.env:/etc/besu/env

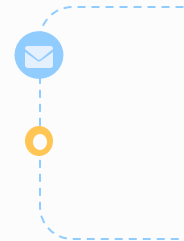
  command:
    --bootnodes=enode://${BOOTNODE_ID}@172.4.0.111:30303
    --data-path=/opt/besu/database
    --genesis-file=/etc/besu/config/genesis.json
    --node-private-key-file=/etc/besu/keys/key
    --min-gas-price=0
    --rpc-http-enabled
    --rpc-http-host=0.0.0.0
    --rpc-http-api=ETH,NET,QBFT,ADMIN,WEB3
    --host-allowlist=""
    --rpc-http-cors-origins="all"
    --logging=INFO
```

- Esponiamo le varie porte sul nostro sistema locale
- - Porta: 8545 all' 8548
  - Protocollo UDP porta: 30303
  - Protocollo TCP porta: 30303
  - Protocollo P2P porta: 30303
- In questo caso specifico, per far comunicare i container tra di loro, creiamo una bridge network e assegnamo ad ogni nodo un'indirizzo IP :
  - BOOTNODE : 172.4.0.111
  - NODE 1 : 172.4.0.201
  - NODE 2 : 172.4.0.202
  - NODE 3 : 172.4.0.203

```
networks:
  chain_net:
    driver: bridge
    ipam:
      config:
        - subnet: 172.4.0.0/24
```



# HYPERLEDGER BESU e FireFly Smart Contract Batch Pin



```
// SPDX-License-Identifier: Apache-2.0

pragma solidity >=0.6.0 <0.9.0;

import "./IBatchPin.sol";

UnitTest stub | dependencies | uml | funcSigs | draw.io
contract Firefly is IBatchPin {
    ftrace | funcSig | selector | natspec
    function pinBatchData(bytes calldata data) public override {
        bytes32 uuids;
        bytes32 batchHash;
        string memory payloadRef;
        bytes32[] memory contexts;
        (uuids, batchHash, payloadRef, contexts) = abi.decode(
            data,
            (bytes32, bytes32, string, bytes32[])
        );
        emit BatchPin(
            tx.origin,
            block.timestamp,
            "firefly:contract_invoke_pin",
            uuids,
            batchHash,
            payloadRef,
            contexts
        );
    }

    ftrace | funcSig | selector | natspec
    function pinBatch(
    ) public override {
    }

    ftrace | funcSig | selector | natspec
    function networkAction(string memory action!, string memory payload!) public {
    }

    ftrace | funcSig | selector | natspec
    function networkVersion() public pure returns (uint8) {
        return 2;
    }
}
```

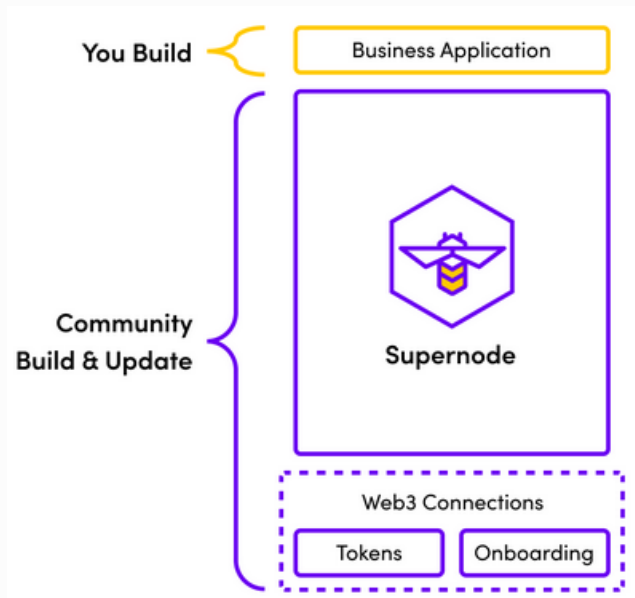
- Compilazione dello Smart contract
- Deploy sulla rete di Besu con Truffle o HardHat
- Salvataggio dell'Address dello Smart Contract deployato
- Inizializzazione degli Stack di FireFly

Parametri da tenere in considerazione :

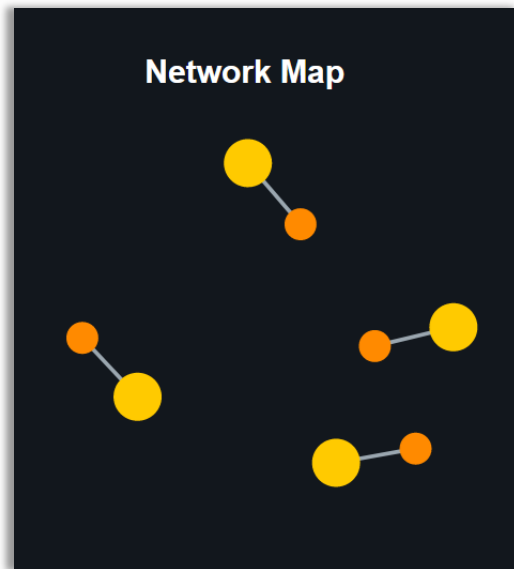
- **Blockchain-node** "remote-rpc"
- **Chain-id, Contract-Address**
- **Remote-node-Url**

ff init ethereum filiera-token 4 --block-period 2 --blockchain-connector "evmconnect" --blockchain-node "remote-rpc" --chain-id 1337 --contract-address "0xb9A219631Aed55eBC3D998f17C3840B7eC39C0cc" --remote-node-url "http://host.docker.internal:8545/"

# LA NOSTRA RETE



Stack Firefly



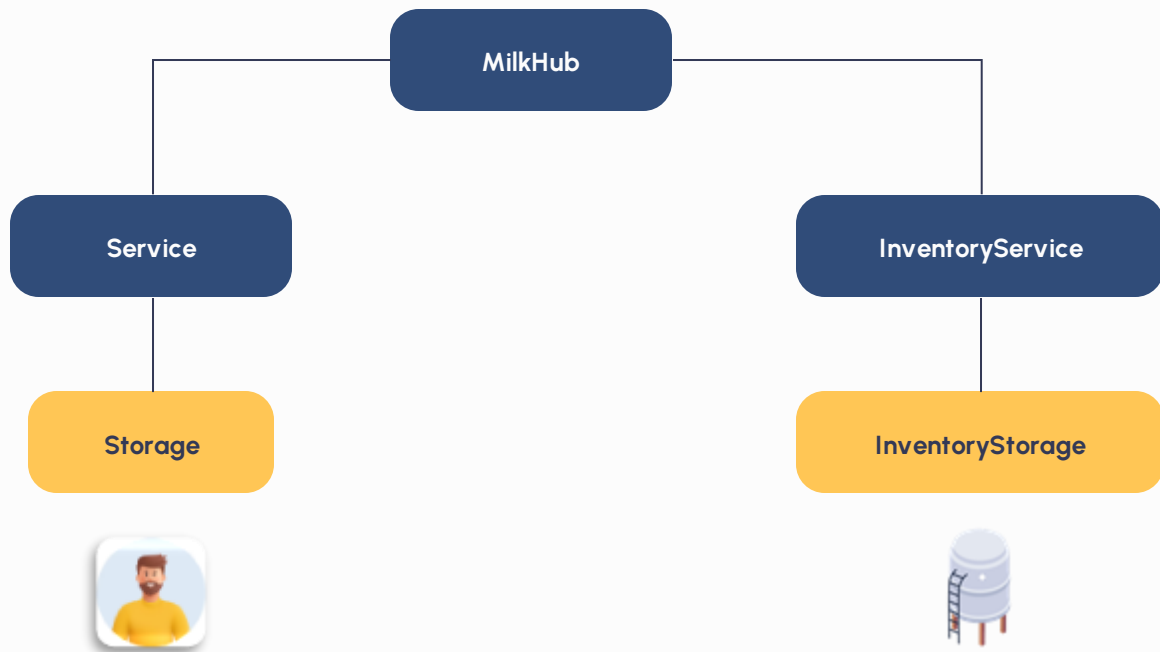


# 03

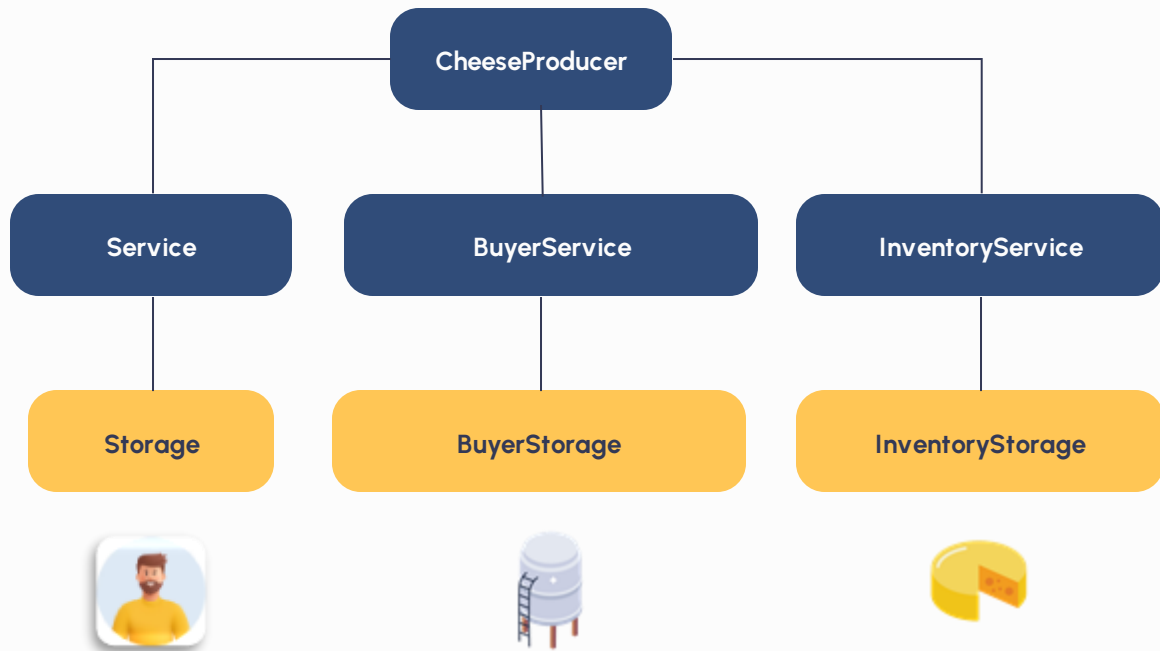
## SMART CONTRACTS



# MICROSERVIZI: MILKHUB

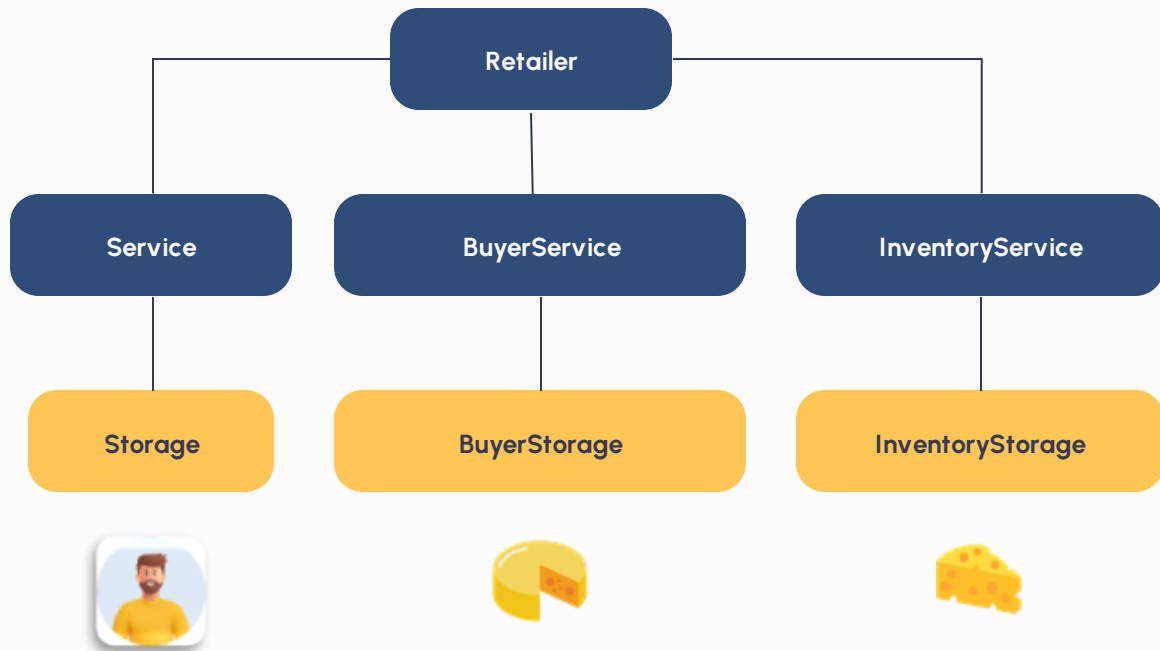


# MICROSERVIZI: CHEESEPRODUCER

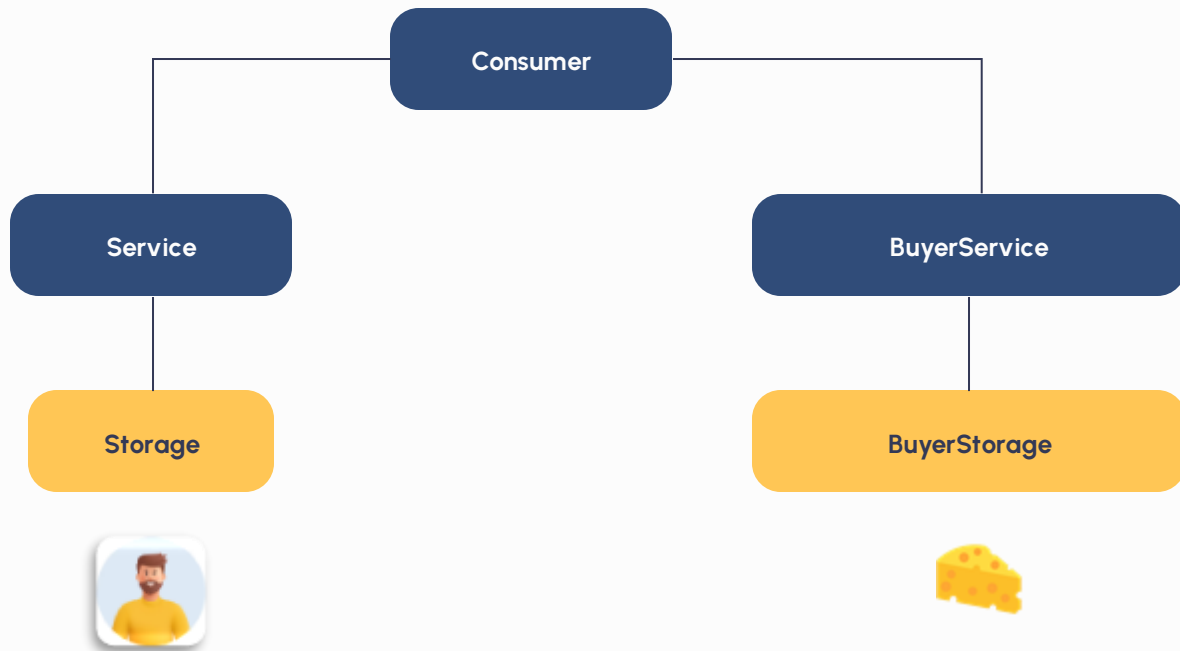




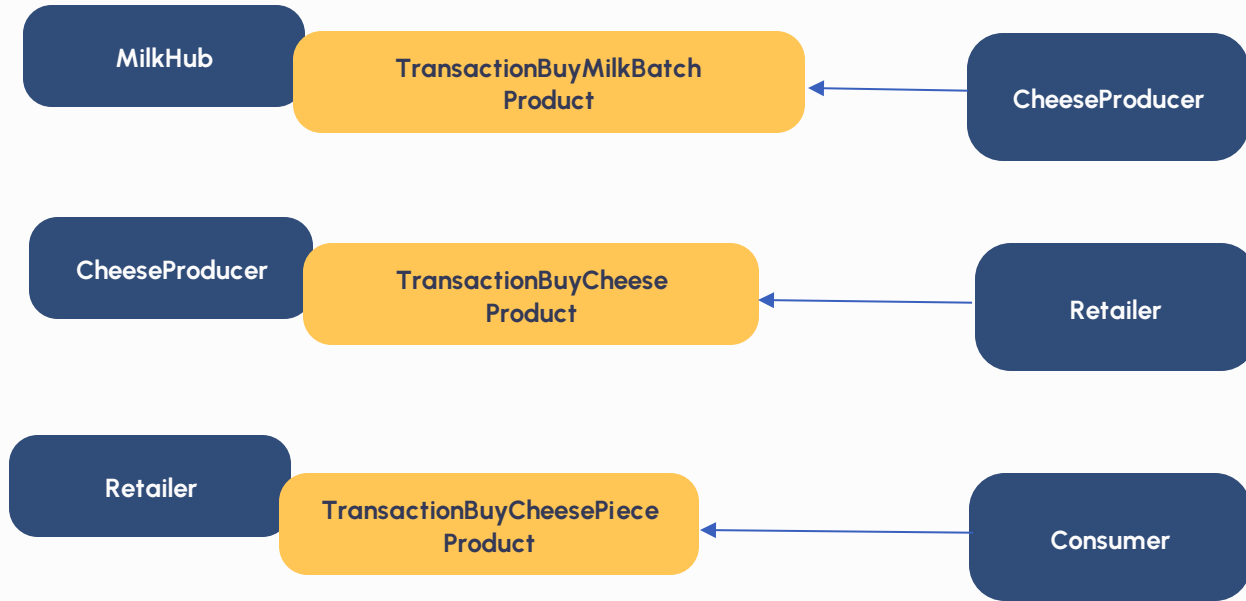
# MICROSERVIZI: RETAILER



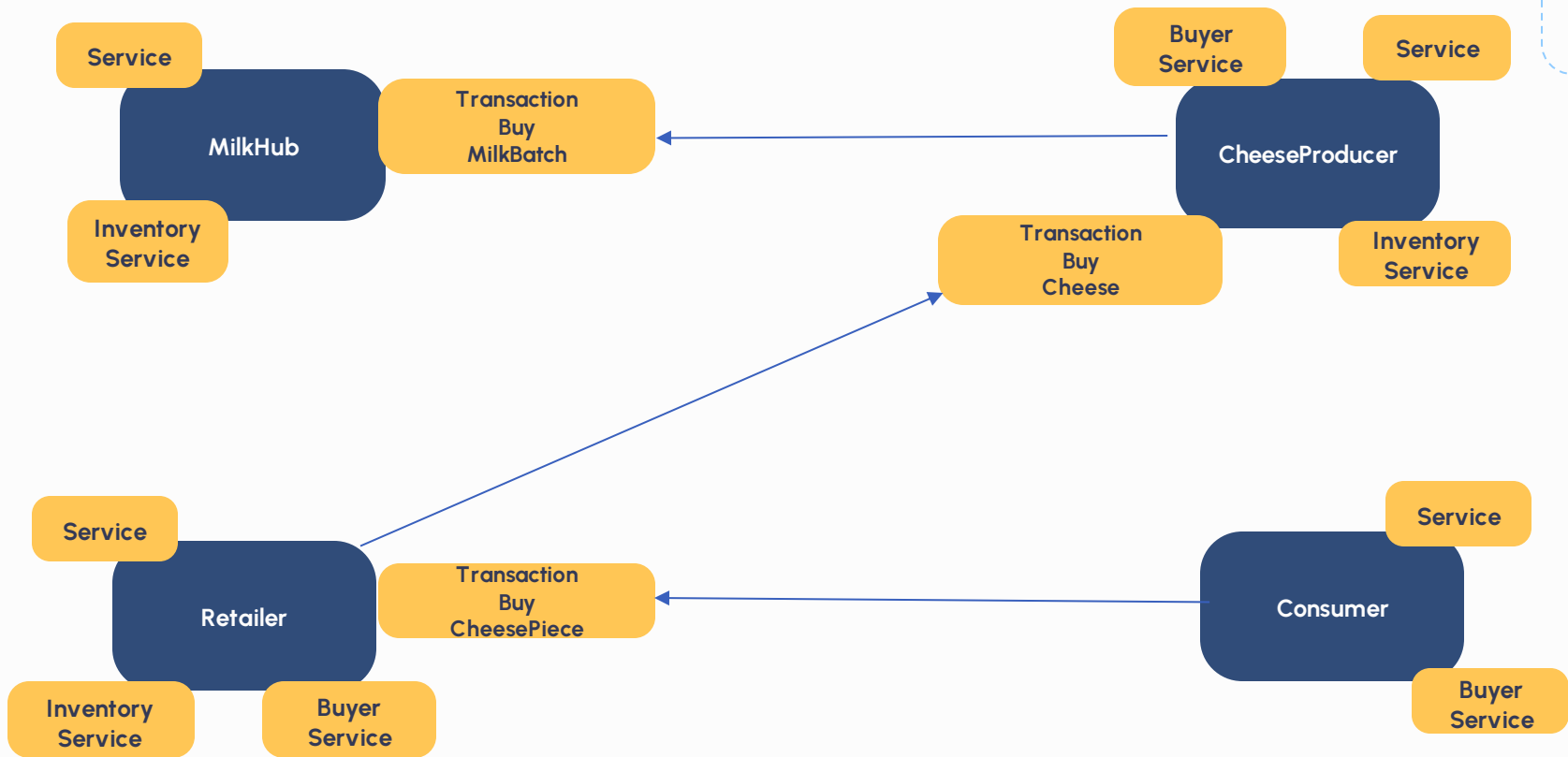
# MICROSERVIZI: CONSUMER



# MICROSERVIZI: Transaction Manager



# DApp: FILIERA-TOKEN



# DESIGN PATTERN: ETERNAL STORAGE

Service



Storage





# 04

## SICUREZZA

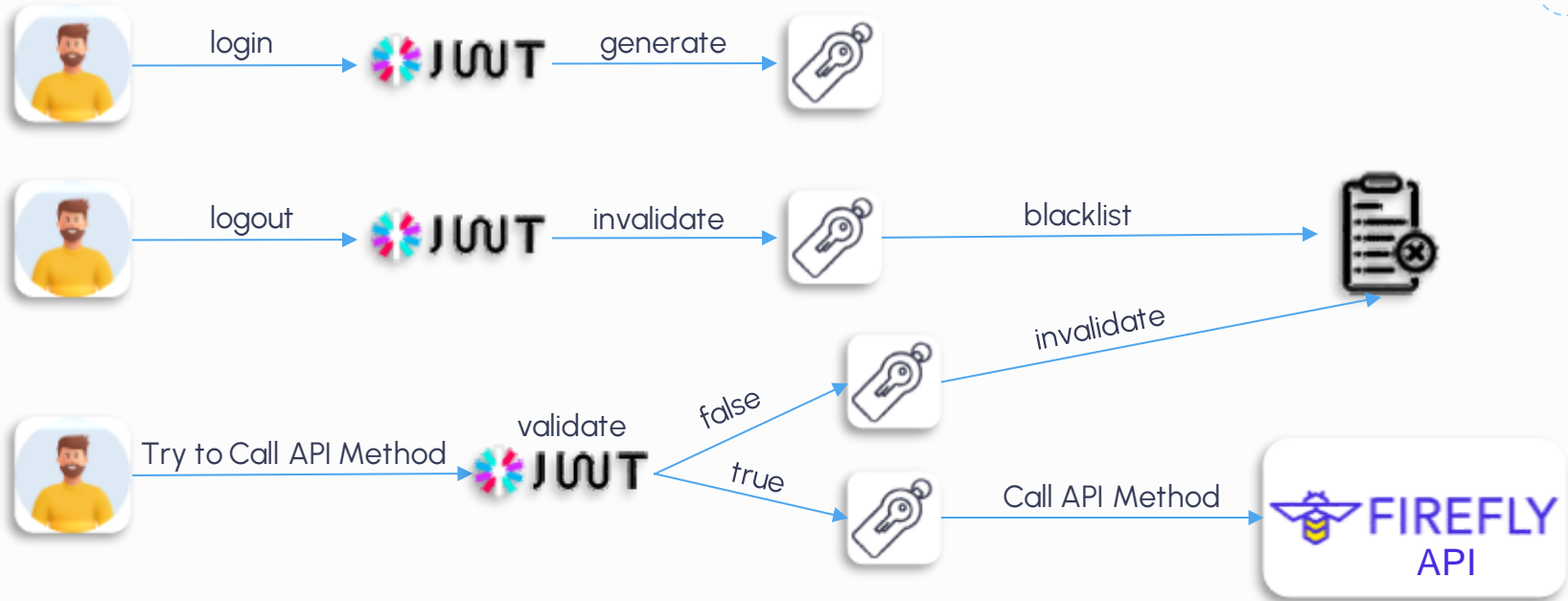


# DESIGN PATTERN: SECURE ETHER TRANSFER



```
contracts > Service > Filieratoken.sol
06 contract Filieratoken is ERC20 {
07     /-
71     function approve(address spender, uint256 amount) public override returns (bool) {
72         return super.approve(spender, amount);
73     }
74
75     /**
76     * @dev Trasferisce 'amount' token Filiera dall'account del mittente all'account del destinatario per l'acquisto di un prodotto.
77     *
78     * @param from L'indirizzo del mittente dei token.
79     * @param to L'indirizzo del destinatario dei token.
80     * @param amount Il numero di token Filiera da trasferire.
81     *
82     * @return 'true' se il trasferimento ha avuto successo, 'false' in caso contrario.
83     */
84     function transferTokenBuyProduct(address from, address to, uint256 amount) public returns (bool) {
85         require(from != address(this) && to != address(this), "Transazione non valida per acquisto prodotto");
86         super.transfer(from, to, amount);
87         return true;
88     }
89
90
91     /**
92     * @dev Assegna 'amount' token Filiera all'account dell'utente specificato come incentivo alla registrazione.
93     *
94     * @param to L'indirizzo dell'utente che viene registrato.
95     * @param amount Il numero di token Filiera da assegnare all'utente.
96     *
97     * @return 'true' se la registrazione ha avuto successo, 'false' in caso contrario.
98     */
99     function registerUserWithToken(address to, uint256 amount) public returns (bool) {
100         require(msg.sender != address(this), "Solo il contratto FilieraToken puo' registrare utenti");
101         super.transfer(address(this), to, amount);
102         return true;
103     }
104
105
106 }
107 }
```

# JWT: JSON Web Token





# BCrypt con Salt nascosto

```
String salt = "\$2a\$10\$Gs.PmaGJQtm0ThQF3VkJ2u";
```



```
String _hashPassword(String password) {  
    final hashedPassword = BCrypt.hashpw(password, salt);  
    return hashedPassword;  
}
```

# 05

## **FUNZIONALITA' PRINCIPALI**




# Transazione: Selezione prodotti

Visual Studio Code  
gio 7 mar 18:11


FilieraToken-Shop

localhost:42389/#/home-page-user/CheeseProducer/115192126750668270867982429930761125630048164786147622271849641239322942242816


Filiera-Token-Hompage




**Partita di Latte 5FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 10-10-2024




**Partita di Latte 2FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 11-11-2025




**Partita di Latte 2FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 10-11-2027




**Partita di Latte 1FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 30-11-2025



**Partita di Latte 1FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza:



**Partita di Latte 1FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 10-11-2000



**Partita di Latte 2FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 20-04-2025


Saldo: 49FTL

# Transazione: Acquisto


Visual Studio Code | gio 7 mar 18:12

FilieraToken-Shop | localhost:42389/#/home-page-user/CheeseProducer/11519212675066827086798242993076112563004816478614762227184964123932942242816


### Filiera-Token-Homepage





**Partita di Latte 5FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 10-10-2024




**Partita di Latte 2FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 11-11-2025











**Partita di Latte 1FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 10-11-2000



**Partita di Latte 2FTL**  
Silos contenente latte, disponibile all'acquisto immediato.  
Scadenza: 20-04-2025



ID: 945883788680512548360561859  
34430039545907642761904769158  
638728979713668546178


Scadenza: 10-10-2024

Quantità disponibile: 40L

Prezzo: 5FTL

Quantità richiesta (L)

Back Buy



**Dati owner**

\* Address :  
0x7ddc959b89472a1812ace5b2d2a  
e6f2925c0aabd

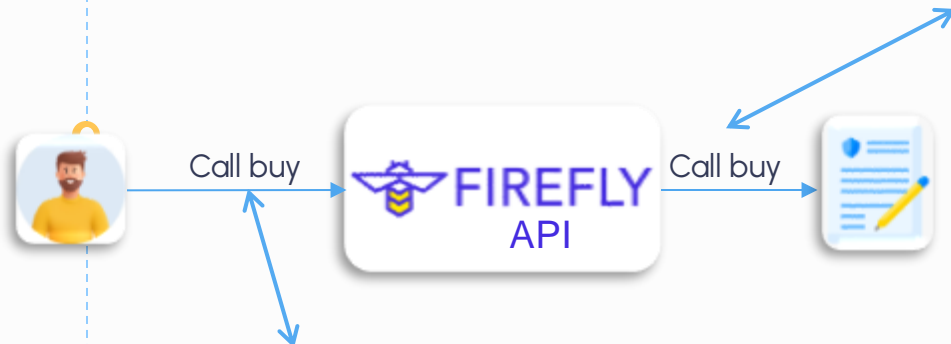
Saldo: 49FTL



# Transazione: Acquisto



TransactionBuyMilkBatch  
Product



```
Future<bool> buyMilkBatchProduct(
    String milkBatchId,
    String quantityToBuy,
    String buyer,
    String ownerMilkBatch,
    String totalPrice,
) async {

    var url = Uri.parse(API.buildURL(API.CheeseProducerNodePort, API.TransactionBuyMilkBatchService , API.Invoke, "BuyMilkBatchProduct"));

    var body = jsonEncode(API.buyMilkBatchProductBody(milkBatchId, quantityToBuy, buyer, ownerMilkBatch, totalPrice));

    try {
        var response = await http.post(
            url,
            headers: API.getHeaders(),
            body: body,
        );
    }

    if (response.statusCode == 200 || response.statusCode == 202) {
        // Transazione è stata effettuata con successo
        print('Risposta: ${response.body}');
        return true;
    } else {
        print('Errore: ${response.statusCode}');
        print('Messaggio di errore: ${response.body}');
        return false;
    }
} catch (e) {
    print('Errore durante la richiesta: $e');
    return false;
}
```

```
function BuyMilkBatchProduct(
    address buyer,
    address ownerMilkBatch,
    uint256 _id MilkBatch,
    uint256 _quantityToBuy,
    uint256 totalPrice)
    external {

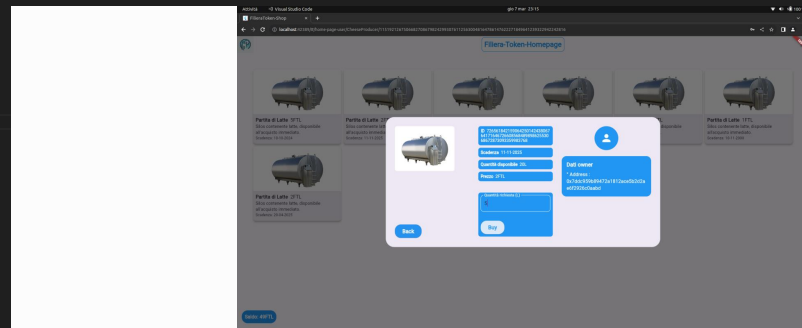
    // Verifica degli address con controlli generici
    require(buyer != address(0), "Invalid sender address");
    require(ownerMilkBatch != address(0), "Invalid owner address");
    require(buyer != ownerMilkBatch, "Cannot buy from yourself");
    // Verifica della presenza del Prodotto
    require(milkhubInventoryService.isMilkBatchPresent(ownerMilkBatch, _id MilkBatch), "Product not found");
    // Verifica della quantità da acquistare rispetto alla quantità totale
    require(_quantityToBuy <= milkhubInventoryService.getMilkBatchQuantity(ownerMilkBatch, _id MilkBatch), "Invalid quantity");
    // Verifica del saldo dell'acquirente
    require(filieraTokenService.balanceOf(buyer) >= totalPrice, "Insufficient balance");

    // Acquisto
    require(filieraTokenService.transferTokenBuyProduct(buyer, ownerMilkBatch, totalPrice), "Acquisto non andato a buon fine!");

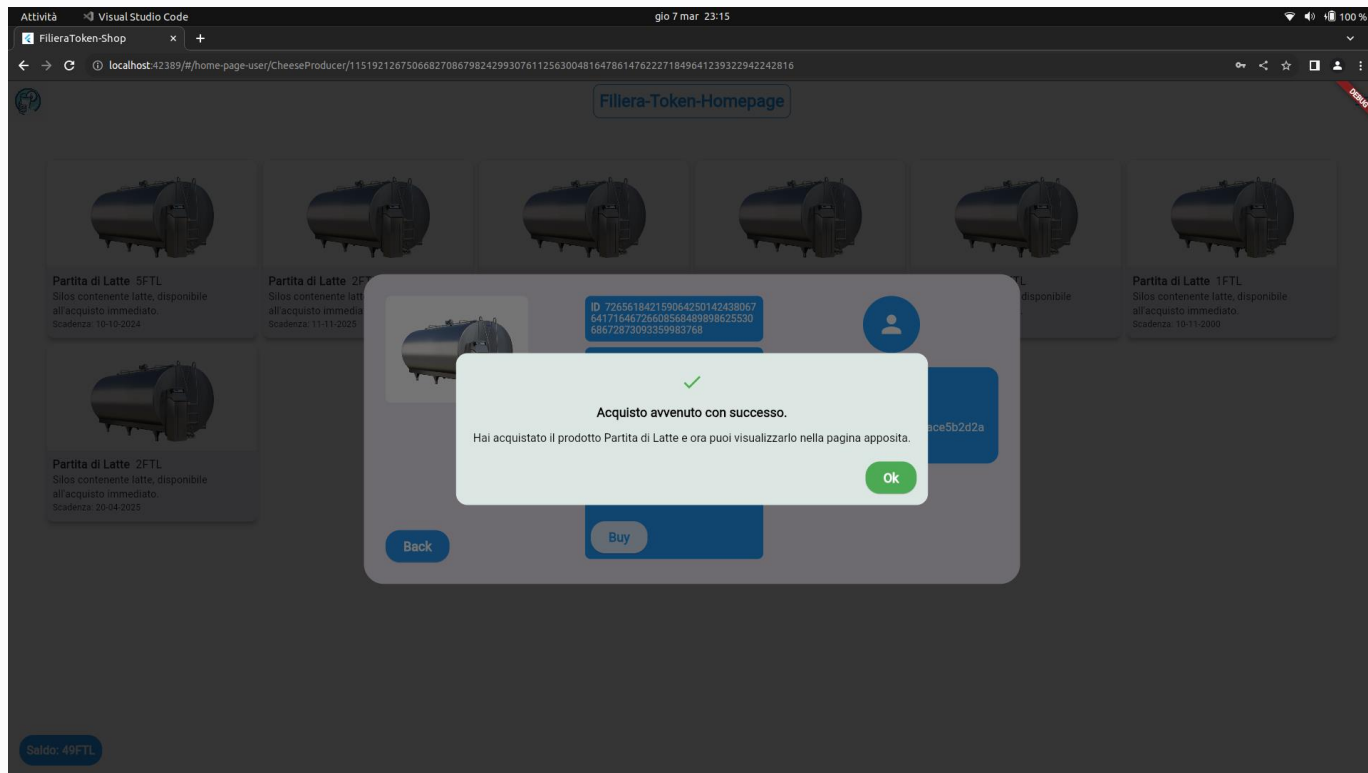
    // Aggiornamento del saldo del MilkHub
    uint256 newMilkHubBalance = filieraTokenService.balanceOf(ownerMilkBatch);
    milkhubService.updateMilkHubBalance(ownerMilkBatch, newMilkHubBalance);
    // Aggiornamento del saldo del CheeseProducer
    uint256 newCheeseProducerBalance = filieraTokenService.balanceOf(buyer);
    cheeseProducerService.updateCheeseProducerBalance(buyer, newCheeseProducerBalance);

    // Riduzione della quantità nel MilkHubInventory
    uint256 currentQuantity = milkhubInventoryService.getMilkBatchQuantity(ownerMilkBatch, _id MilkBatch);
    milkhubInventoryService.updateMilkBatchQuantity(ownerMilkBatch, _id MilkBatch, currentQuantity - _quantityToBuy);

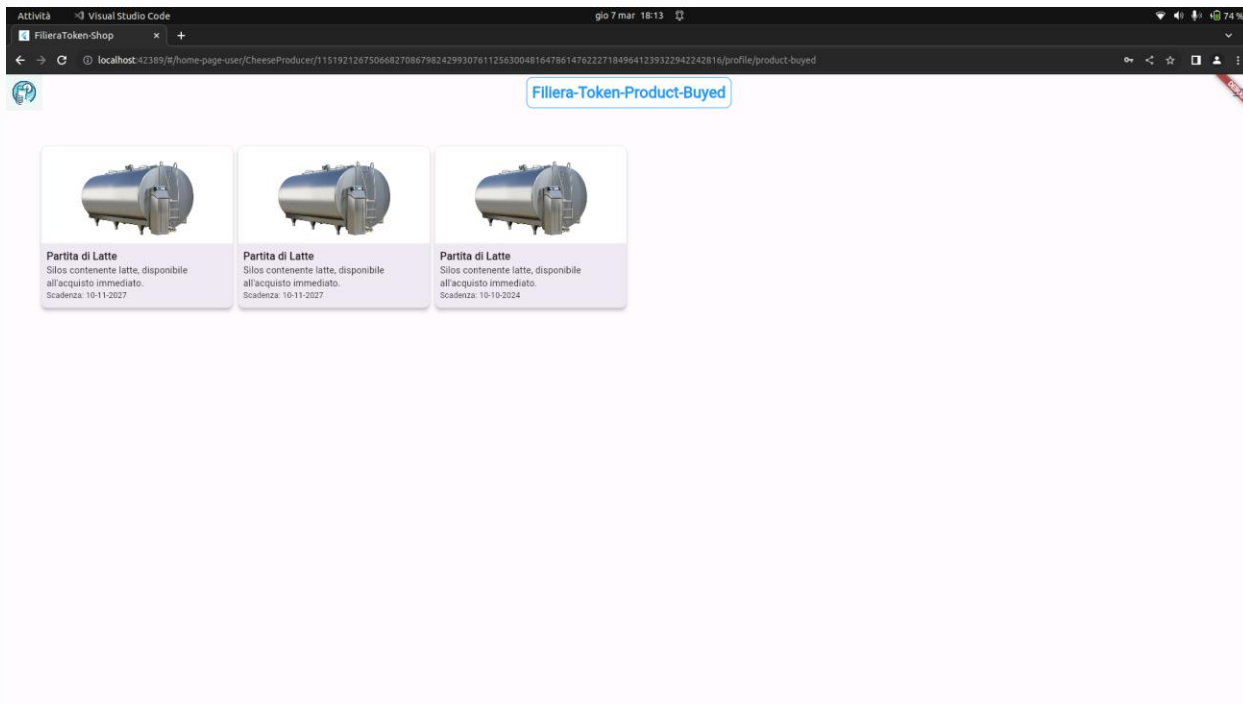
    // Aggiunta del MilkBatch nell'inventario del CheeseProducer
    cheeseProducerBuyerService.addMilkBatch(
        ownerMilkBatch,
        buyer,
        _id MilkBatch,
        milkhubInventoryService.getMilkBatchExpirationDate(ownerMilkBatch, _id MilkBatch),
        _quantityToBuy);
}
```



# Transazione: Acquisto

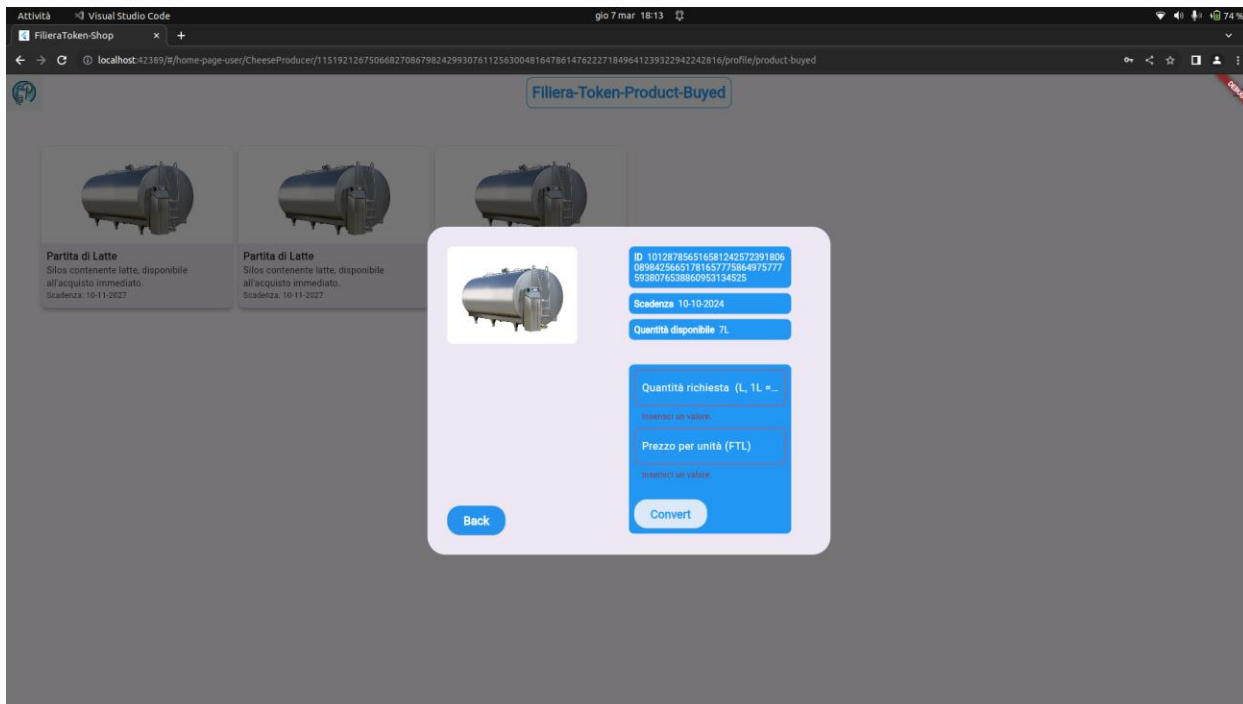


# Conversione prodotti: Selezione

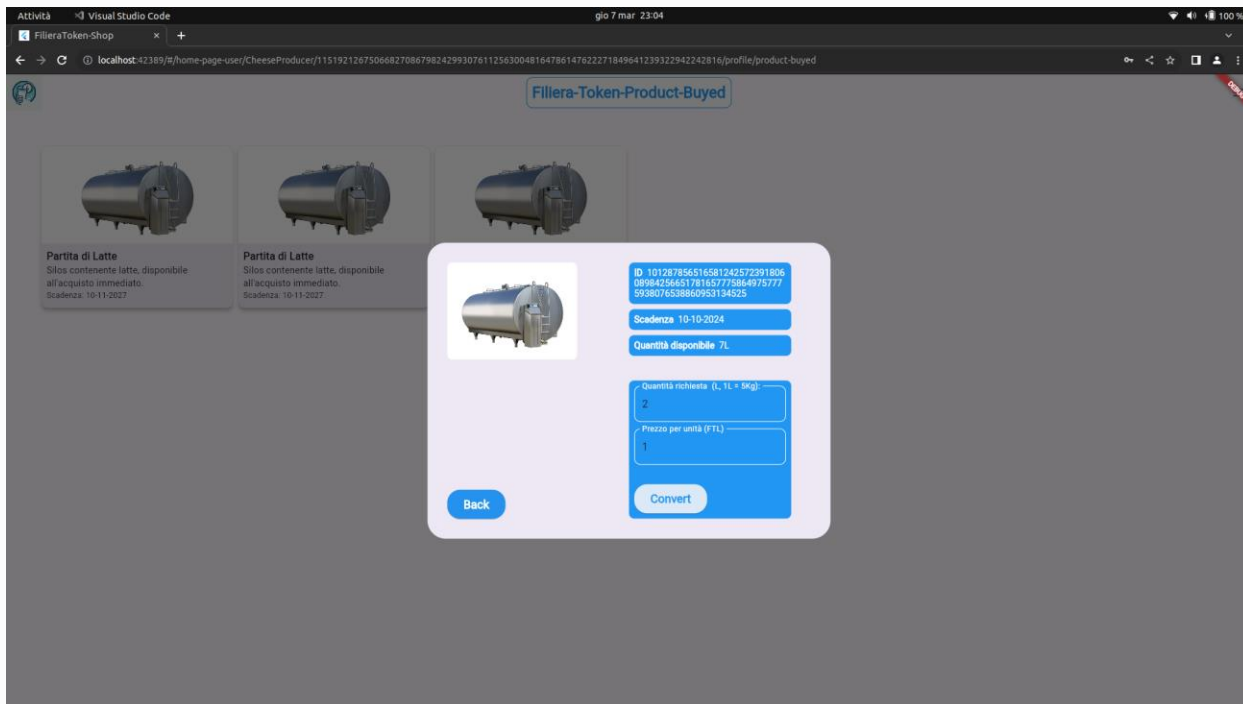




# Conversione prodotti: Selezione



# Conversione prodotti: Conversione



# Conversione prodotti: conversione

```
// Funzione per trasformare i vari milkBatch in CheeseBlock
function transformMilkBatch(address walletCheeseProducer, uint256 idMilkBatch, uint256 quantityToTransform, uint256 pricePerKg, string memory dop) external {

    //Verifico che il prodotto esiste, che la quantità richiesta da trasformare non ecceda il massimo consentito
    require(cheeseProducerBuyerService.isMilkBatchPresent(walletCheeseProducer, idMilkBatch), "Prodotto non presente!");
    // Verifico la quantità
    // Verifico che quella che devo trasformare sia inferiore o uguale a quella che ho acquistato
    require(quantityToTransform <= cheeseProducerBuyerService.getQuantity(walletCheeseProducer, idMilkBatch), "Quantità da trasformare non valida!");

    uint256 newQuantity = cheeseProducerBuyerService.getQuantity(walletCheeseProducer, idMilkBatch) - quantityToTransform;

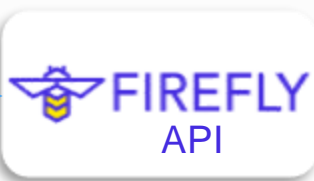
    cheeseProducerBuyerService.updateMilkBatchQuantity(walletCheeseProducer, idMilkBatch, newQuantity);

    uint256 weight = quantityToTransform * 5;
    addCheeseBlock(walletCheeseProducer, dop, weight, pricePerKg);
}
```

CheeseProducerInventoryService



Call transform



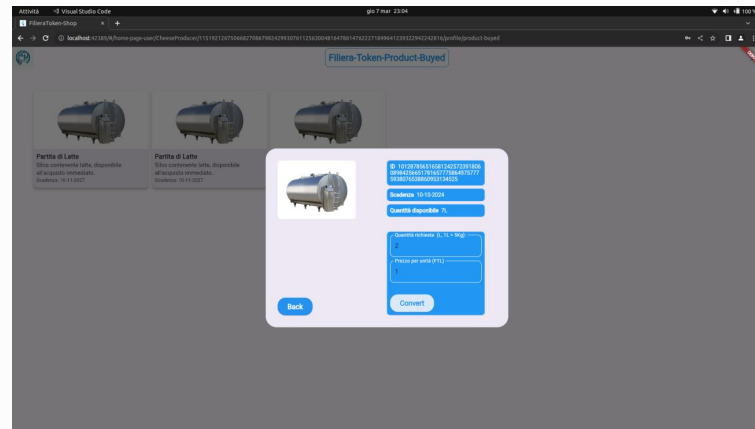
Call transform



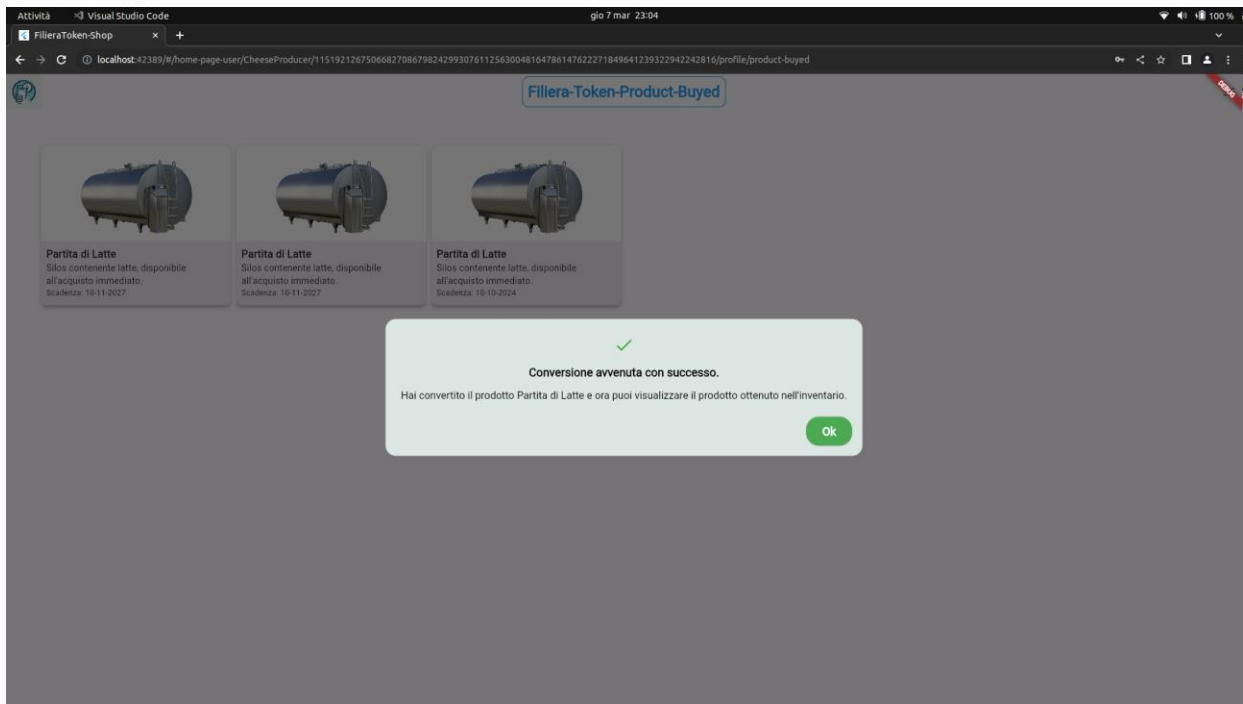
```
Future<bool> transformMilkBatch(String wallet, String idMilkBatch, String quantityToTransform, String pricePerKg, String dop) async {
    String url = API.buildURL(API.CheeseProducerNodePort, API.CheeseProducerInventoryService, API.Invoke, "transformMilkBatch");
    final headers = API.getHeaders();
    final body = jsonEncode(API.getTransformCheeseProducerBody(dop, idMilkBatch, pricePerKg, quantityToTransform, wallet));

    print(url);
    print(headers);
    print(body);
    try {
        final response = await http.post(Uri.parse(url), headers: headers, body: body);

        if (response.statusCode == 200 || response.statusCode == 202) {
            return true;
        } else {
            throw Exception('Failed to transform MilkBatch in CheeseBlock: ${response.statusCode}');
        }
    } catch (error) {
        print('Error transforming MilkBatch in CheeseBlock: $error');
        rethrow;
    }
}
```



# Conversione prodotti: conversione



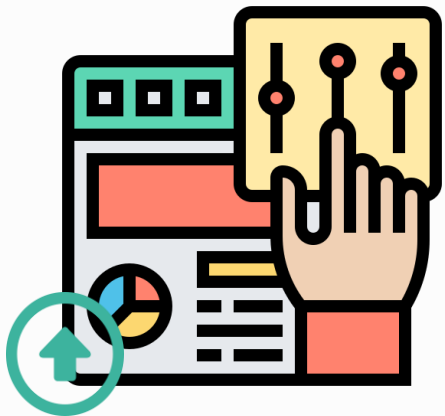
# 06

## CONCLUSIONI

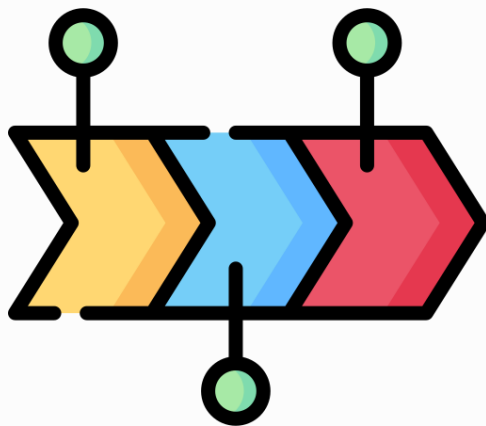


# Sviluppi futuri

- Miglioramento della UI/UX



- Cronologia Eventi





# GRAZIE PER L'ATTENZIONE!

Umberto Della Monica  
Gerardo Leone

