FILIERA-TOKEN-SHOP

Umberto Della Monica
Gerardo Leone

LinearIT
A HYA HOLDING COMPANY
UNIVERSITÀ DEGLI STUDI DI SALERNO

TABELLA DEI CONTENUTI

01

INTRODUZIONE

LA FILIERA

Milk Hub

Centro di raccolta e trasformazione del latte

Cheese Producer

Produttore di formaggio

Retailer

Venditore al dettaglio

Consumer

Consumatore

Partita di latte

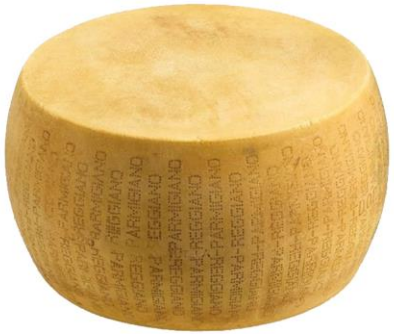
Blocco di Formaggio

Pezzo di formaggio

LA FILIERA: PARTITA DI LATTE

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

LA FILIERA: BLOCCO DI FORMAGGIO

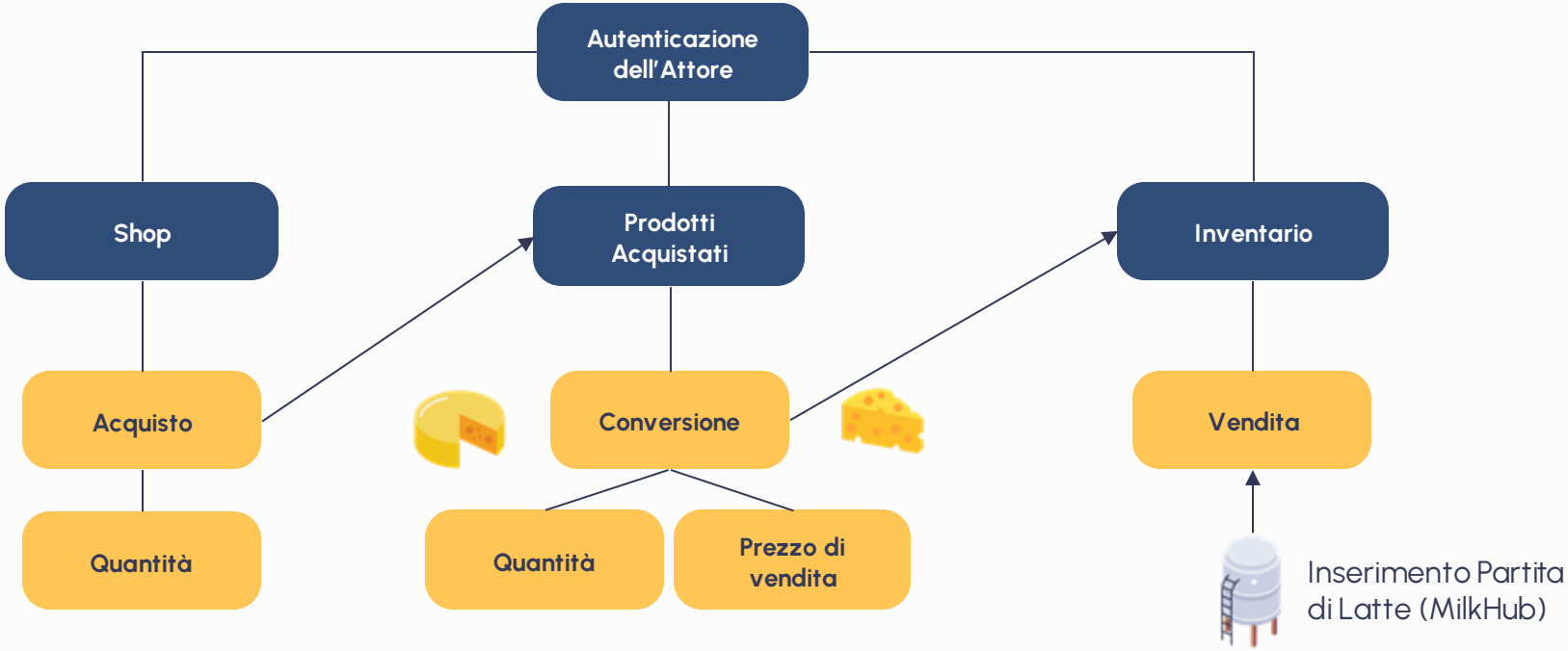


- Identificativo
- Marchio di qualità D.O.P.
- Quantità (in Kg)
- Prezzo (FLT)

LA FILIERA: PEZZO DI FORMAGGIO

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

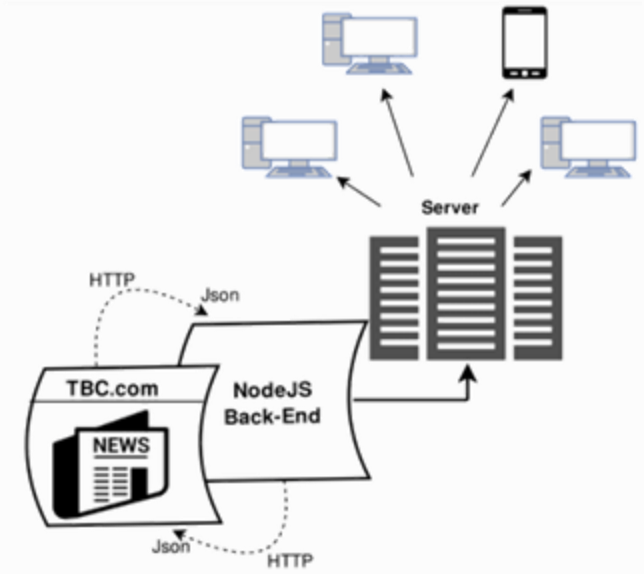
NAVIGAZIONE E INTERAZIONE



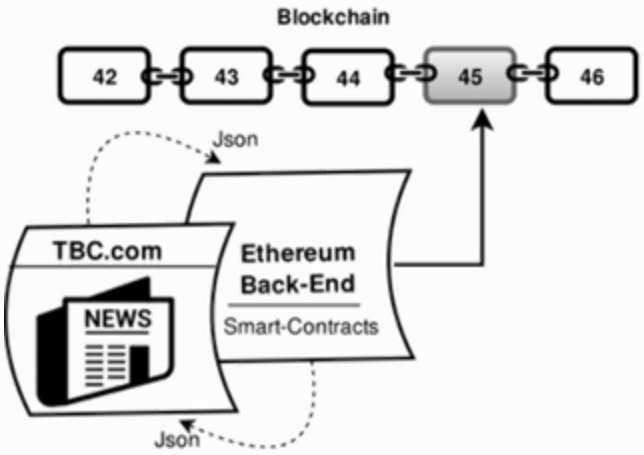
02

ARCHITETTURA

WEB 2.0 vs WEB 3.0

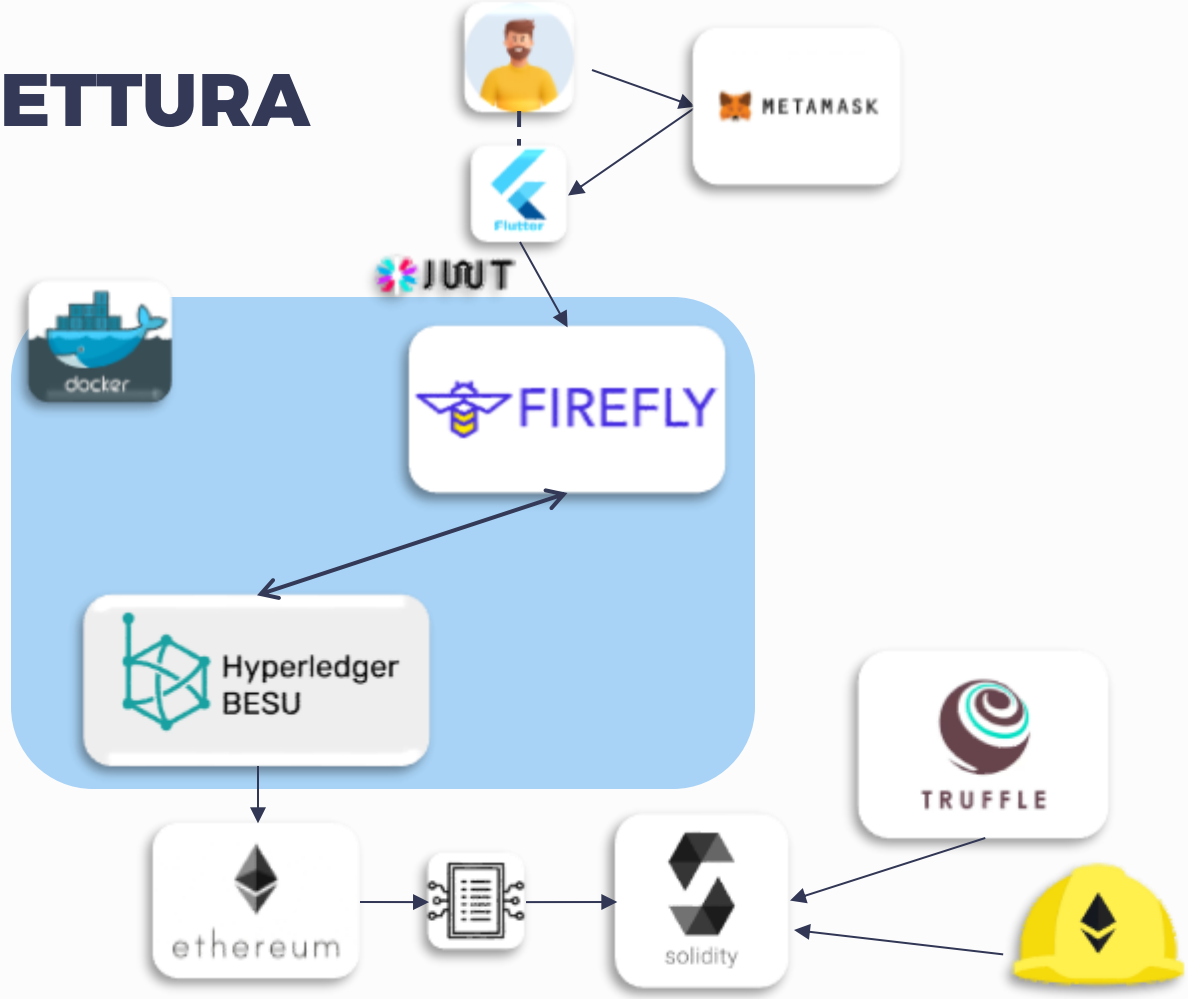


Web 2.0
Architettura centralizzata

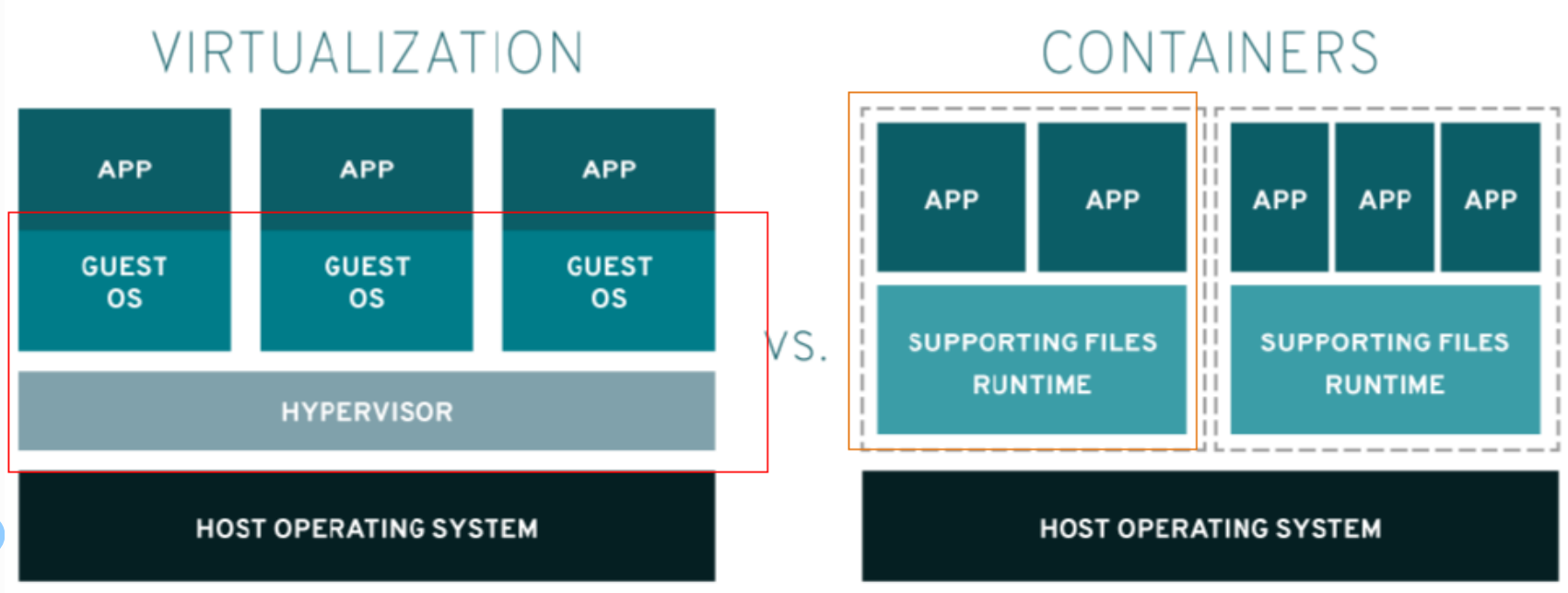


Web 3.0
Architettura decentralizzata

ARCHITETTURA



DOCKER: UN AMBIENTE SICURO



DOCKER

Client

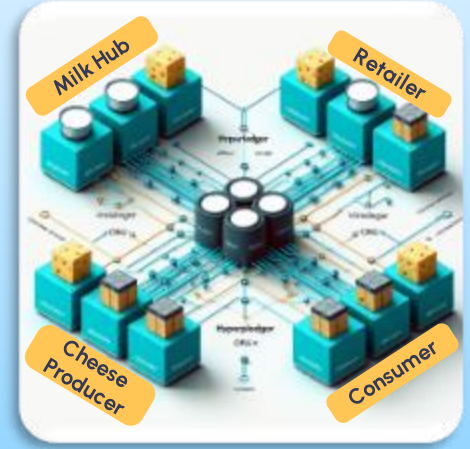
Docker pull

Docker build

Docker run

Docker Host

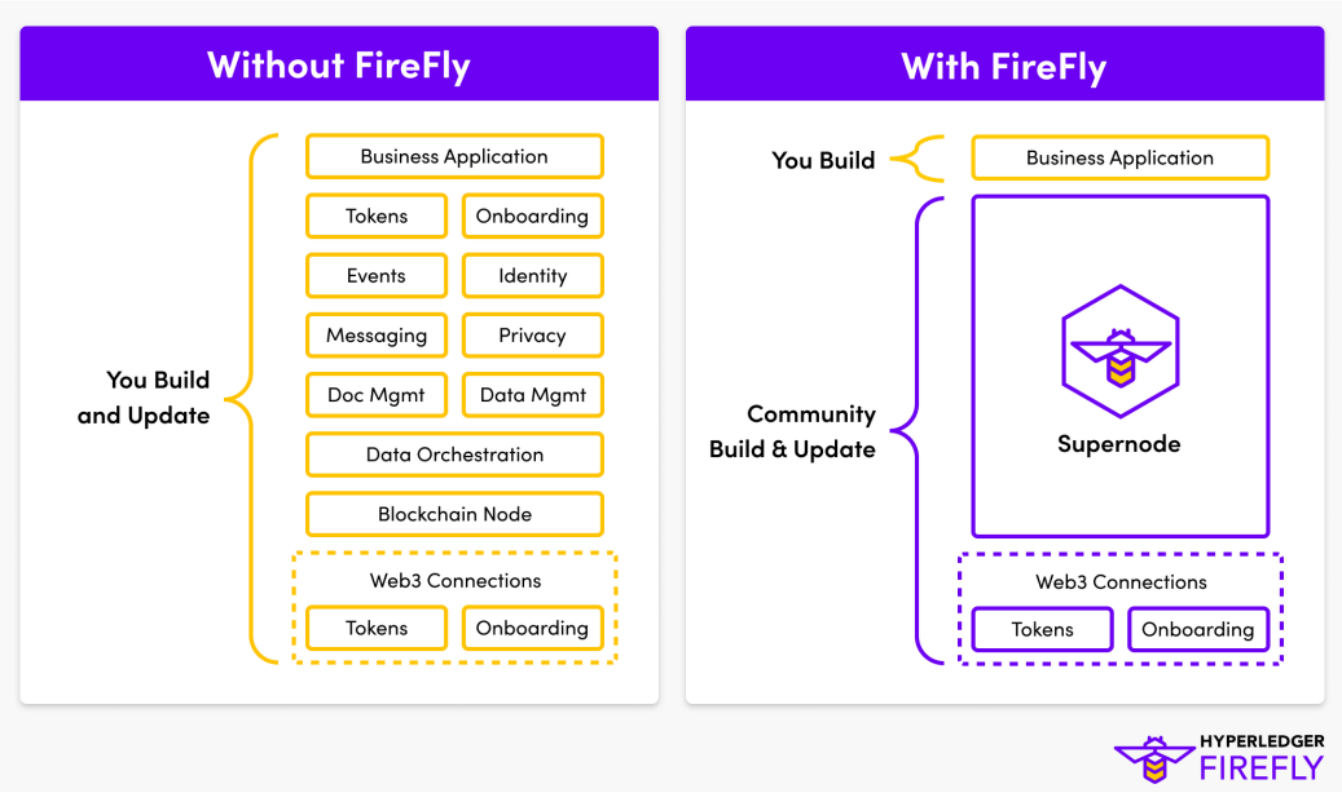
Docker daemon



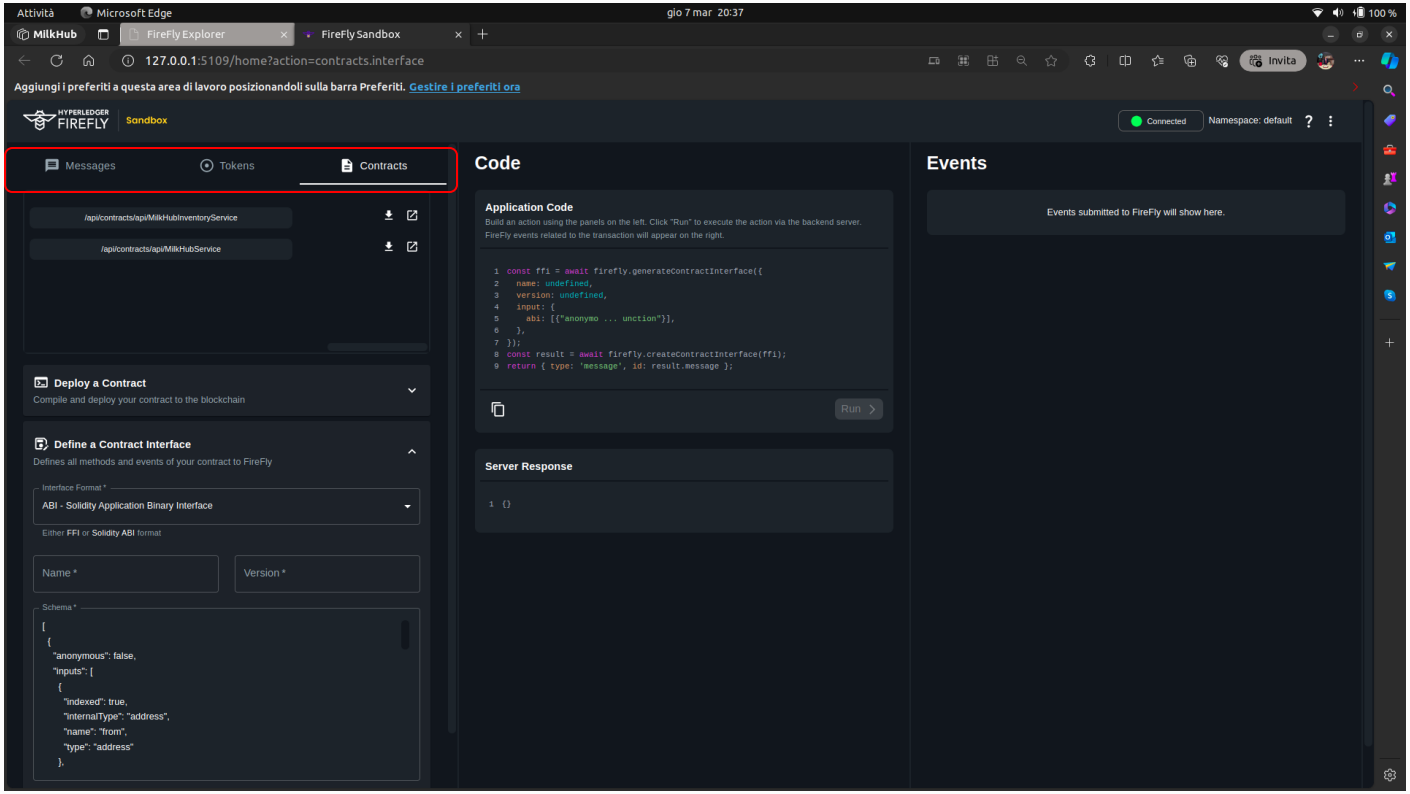
Registry

```
ff init 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/" --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
```

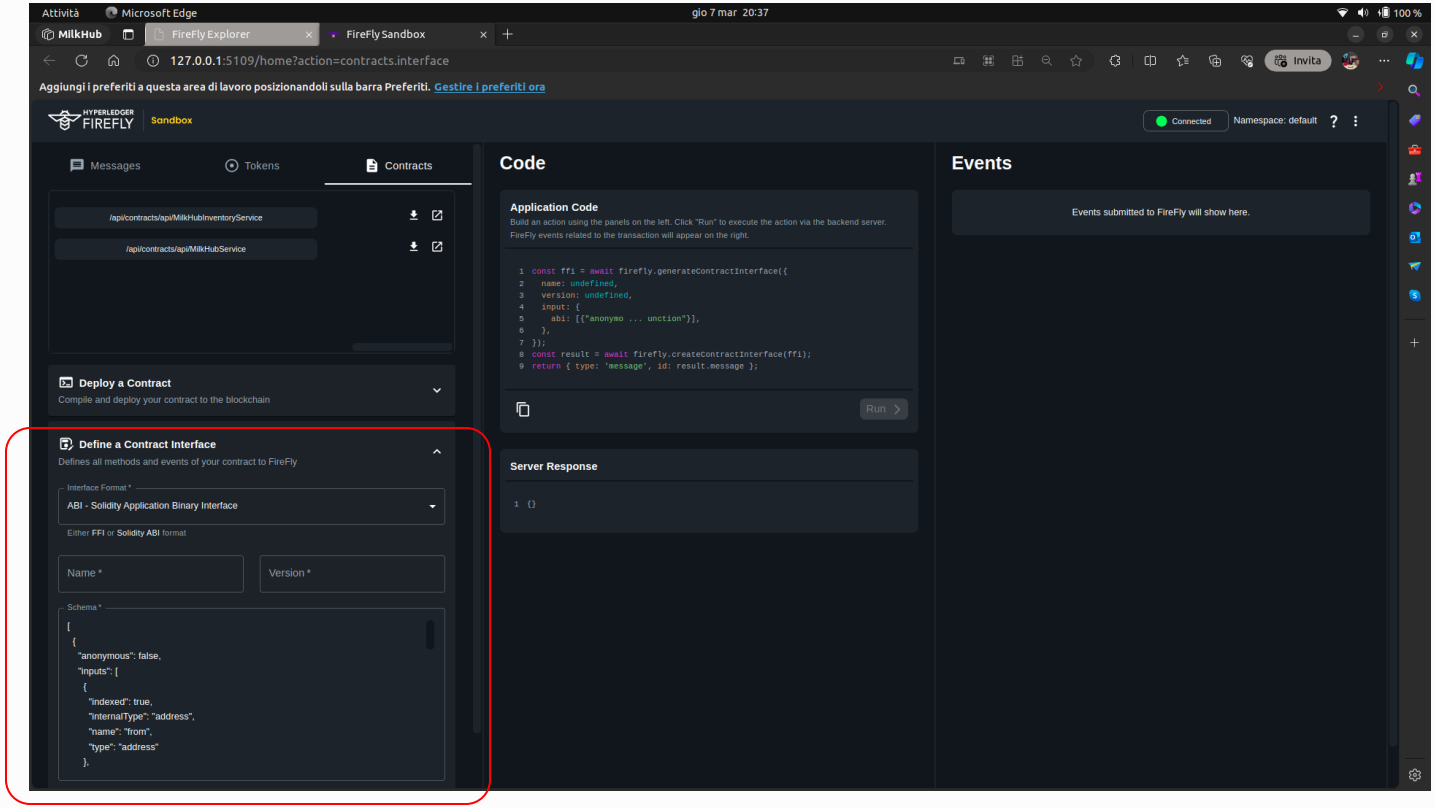
HYPERLEDGER FIREFLY



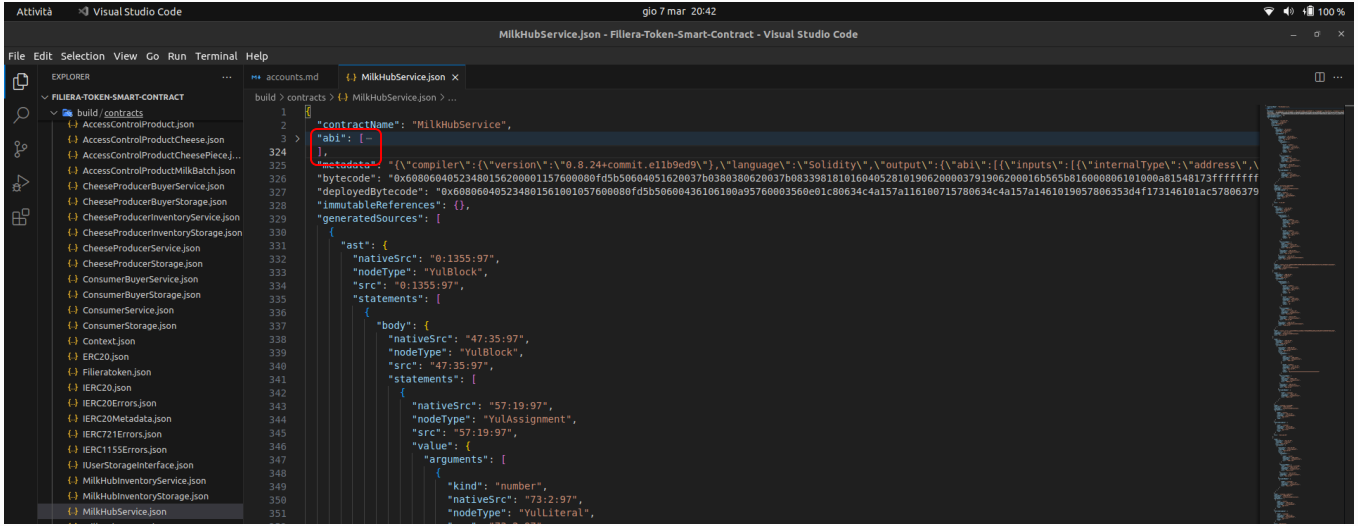
HYPERLEDGER FIREFLY: API



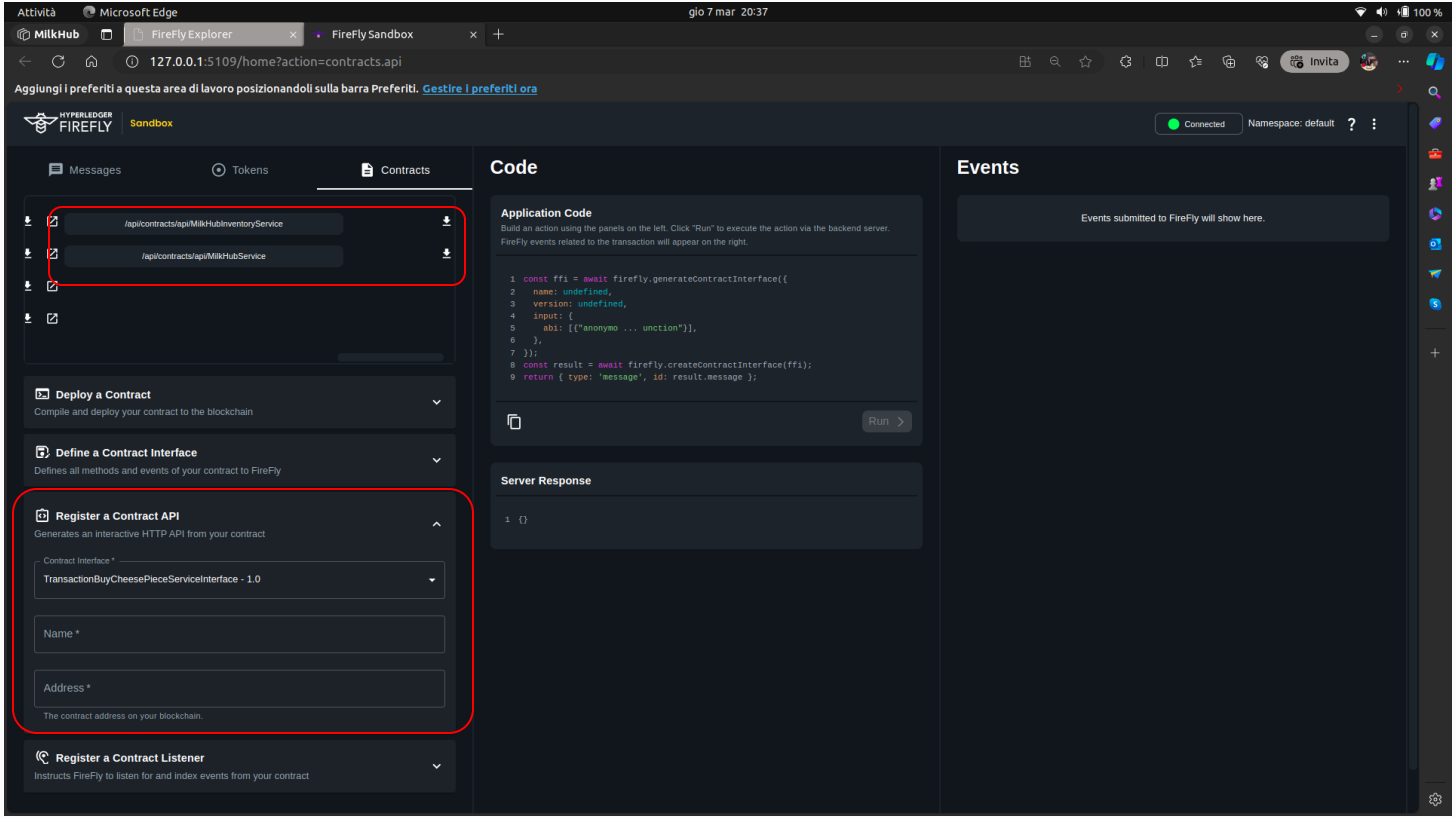
HYPERLEDGER FIREFLY: API



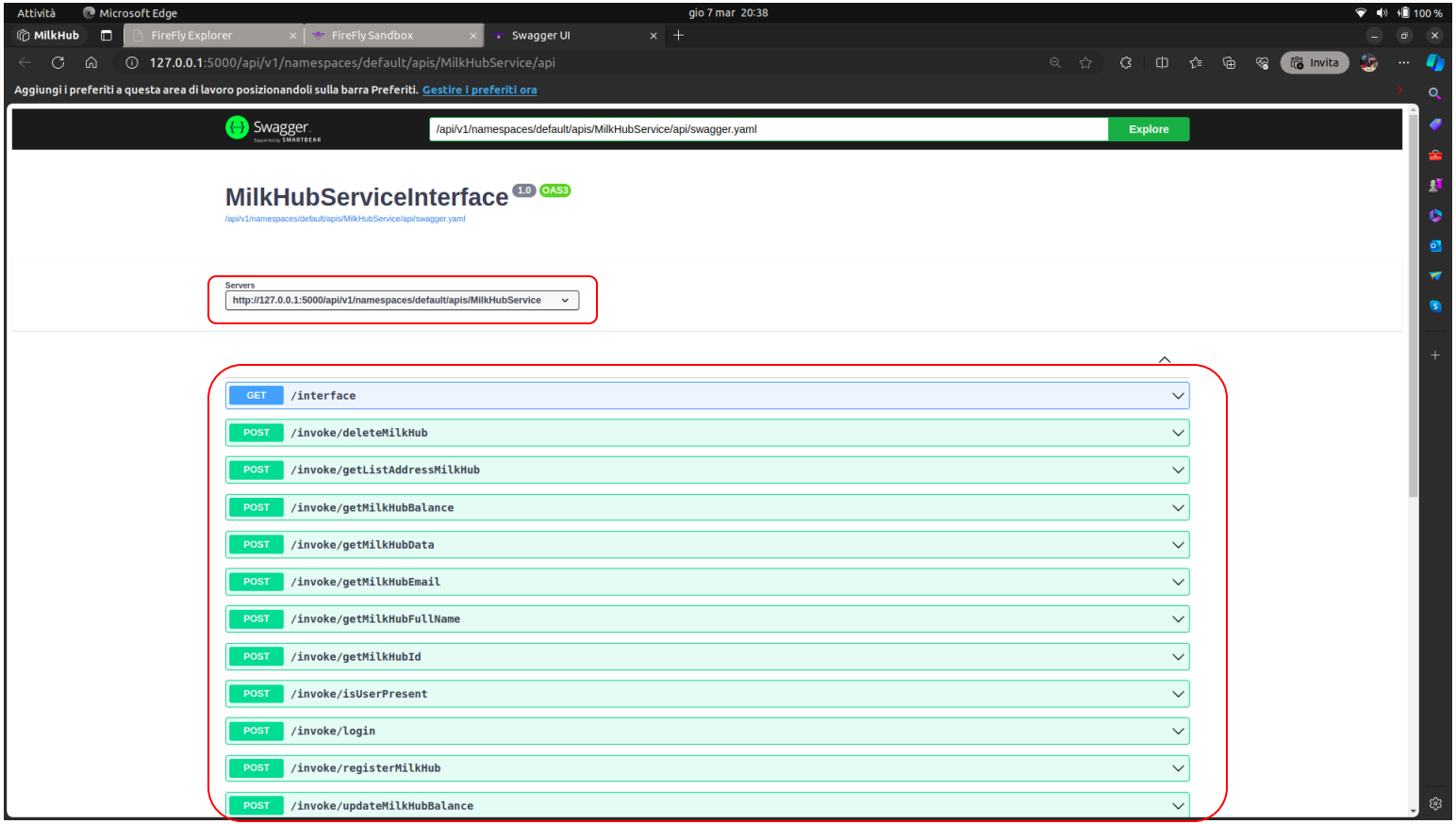
HYPERLEDGER FIREFLY: API



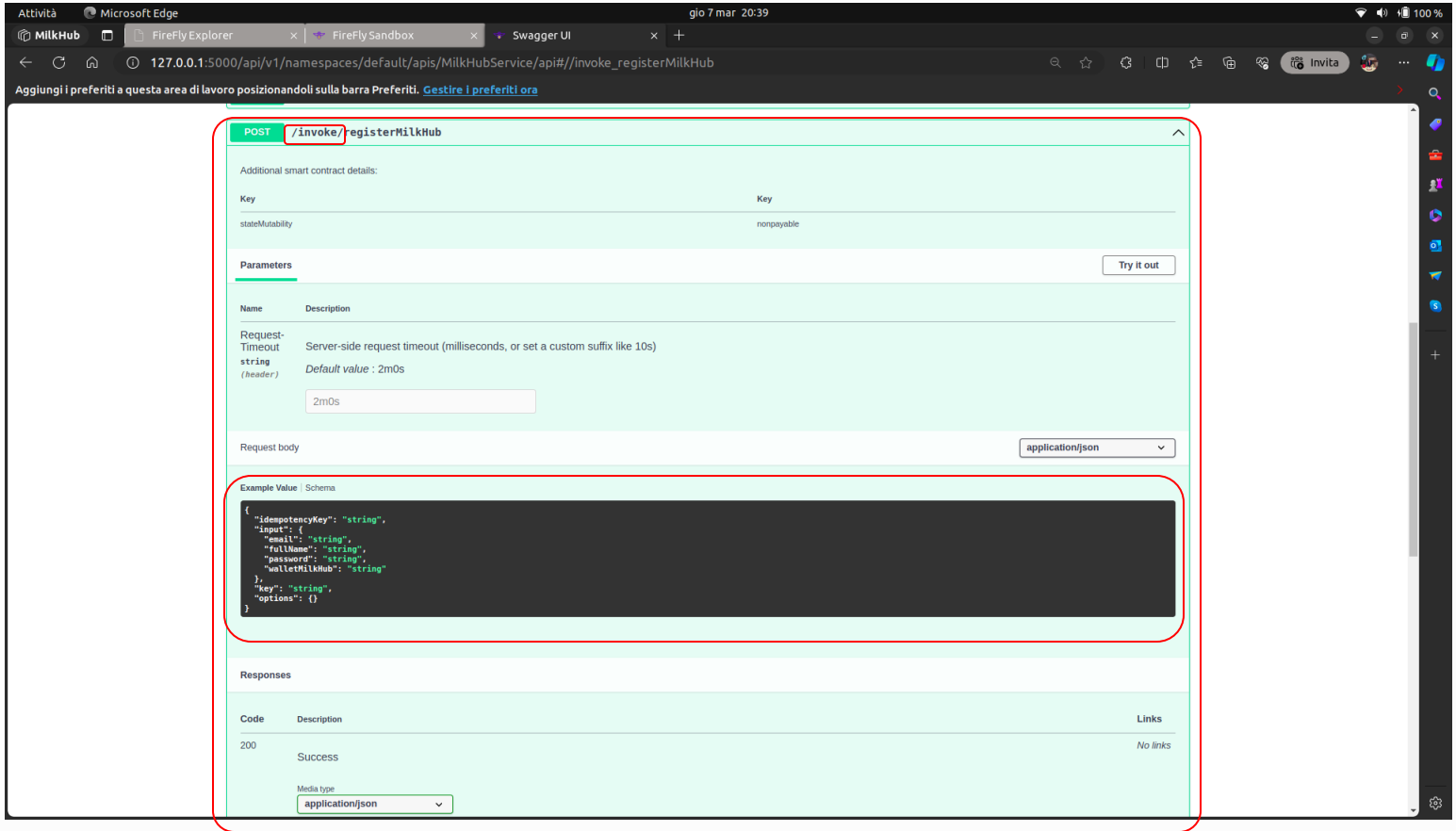
HYPERLEDGER FIREFLY: API



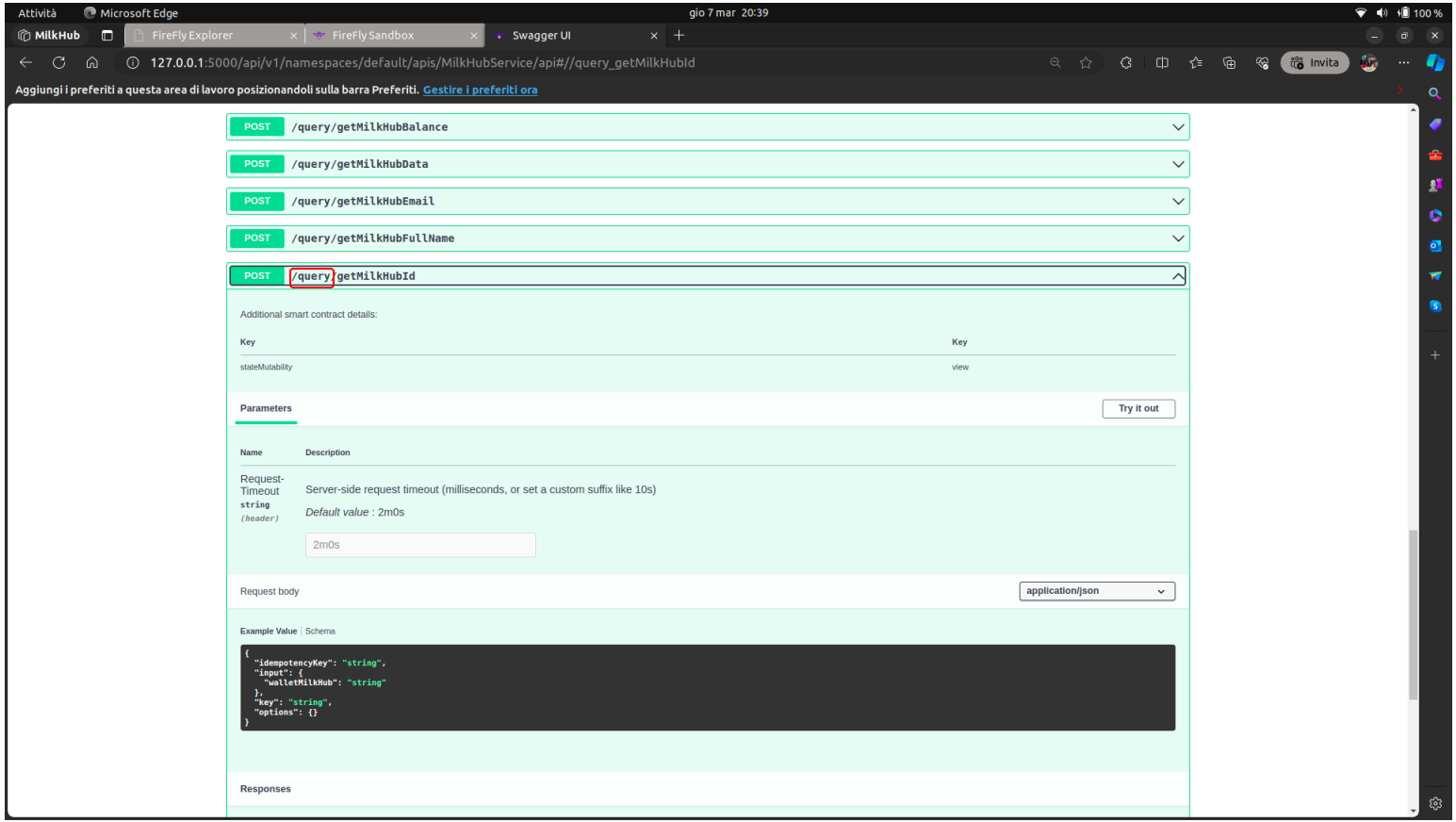
SWAGGER


Attività
Microsoft Edge
gio 7 mar 20:38
100 %
MilkHub
FireFly Explorer
FireFly Sandbox
Swagger UI
127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService/api
Invita
Aggiungi i preferiti a questa area di lavoro posizionandoli sulla barra Preferiti. Gestire i preferiti ora
Swagger
/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
100 %
MilkHub
FireFly Explorer
FireFly Sandbox
Swagger UI
127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService/api#/invoke_registerMilkHub
Invita
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
Try it out
Name
Description
Request-Timeout
string
(header)
Server-side request timeout (milliseconds, or set a custom suffix like 10s)
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
100 %
MilkHub
FireFly Explorer
FireFly Sandbox
Swagger UI
127.0.0.1:5000/api/v1/namespaces/default/apis/MilkHubService/api#//query_getMilkHubId
Invita
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
stateMutability
Key
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
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

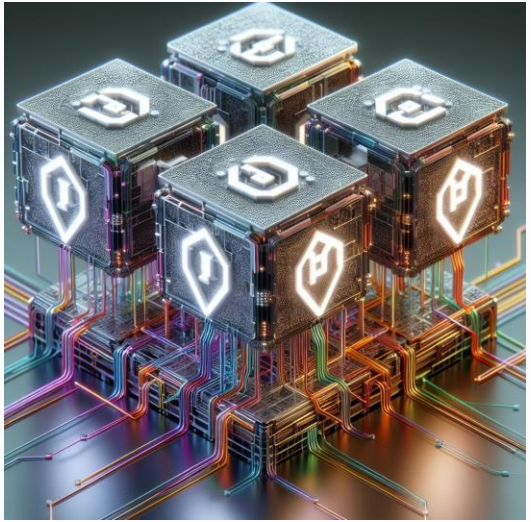
Sono accessibili a chiunque e non richiedono permessi per partecipare. Chiunque può leggere, inviare e confermare transazioni.

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

```jsonc
{} config.json ×
QBFT-network > besu-private-qbft-network-docker > {} config.json > {} blockchain
{
  "genesis": {
    "config": {
      "chainId": 1337,
      "berlinBlock": 0,
      "qbft": {
        "blockperiodseconds": 2,
        "epochlength": 30000,
        "requesttimeoutseconds": 4
      }
    },
    "nonce": "0x0",
    "timestamp": "0x58ee40ba",
    "gasLimit": "0x1ffffffffffffff",
    "difficulty": "0x1",
    "mixHash": "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
    "coinbase": "0x0000000000000000000000000000000000000000",
    "alloc": {
      "fe3b557e8fb62b89f4916b721be55ceb828dbd73": {
        "privateKey": "8f2a55949038a9610f50fb23b5883af3b4ecb3cbb792cbcefbd1542c692be63",
        "comment": "private key and this comment are ignored.  In a real chain, the private key should NOT be stored",
        "balance": "0xad78ebc5ac6200000"
      },
      "627306090abaB3A6e1400e9345bC60c78a8BEf57": {
        "privateKey": "c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",
        "comment": "private key and this comment are ignored.  In a real chain, the private key should NOT be stored",
        "balance": "90000000000000000000000"
      },
      "f17f52151EbEF6C7334FAD080c5704D77216b732": {
        "privateKey": "ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f",
        "comment": "private key and this comment are ignored.  In a real chain, the private key should NOT be stored",
        "balance": "90000000000000000000000"
      }
    }
  },
  "blockchain": {
    "nodes": {
      "generate": true,
      "count": 4
    }
  }
}
```

Config File

```yaml
# 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-gea6b3e973b477b8061e0076bb257dbd7f3faa756                                    #
# 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 degl'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

```yaml
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

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

HYPERLEDGER BESU e FireFly Smart Contract Batch Pin

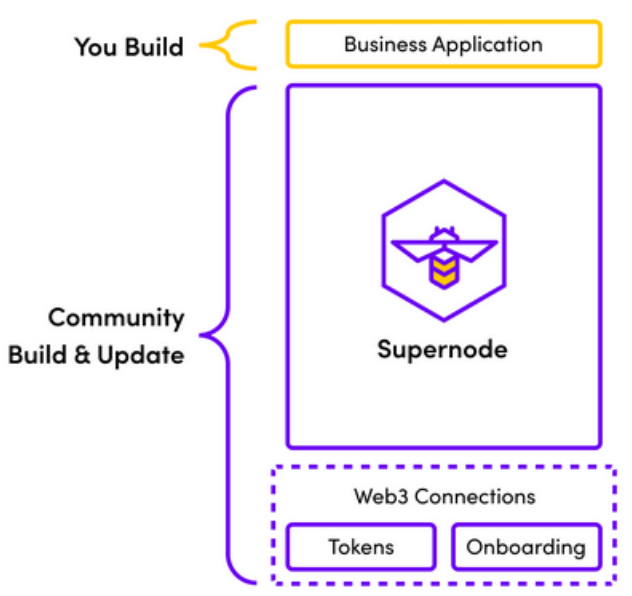
```
// SPDX-License-Identifier: Apache-2.0

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

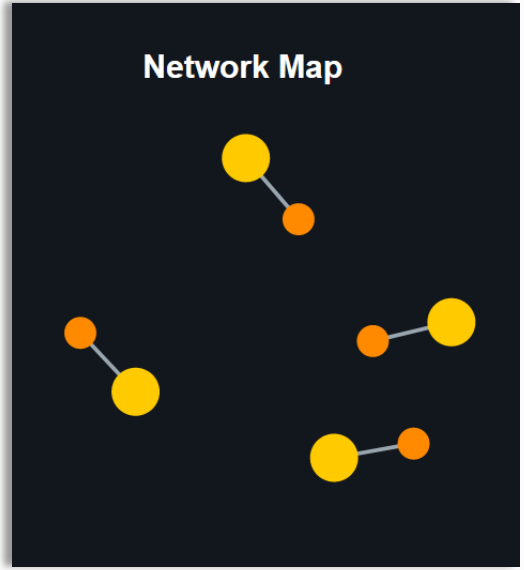
Parametri da tenere in considerazione :
- **Blockchain-node** "remote-rpc"
- **Chain-id**, **Contract-Address**
- **Remote-node-Url**

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



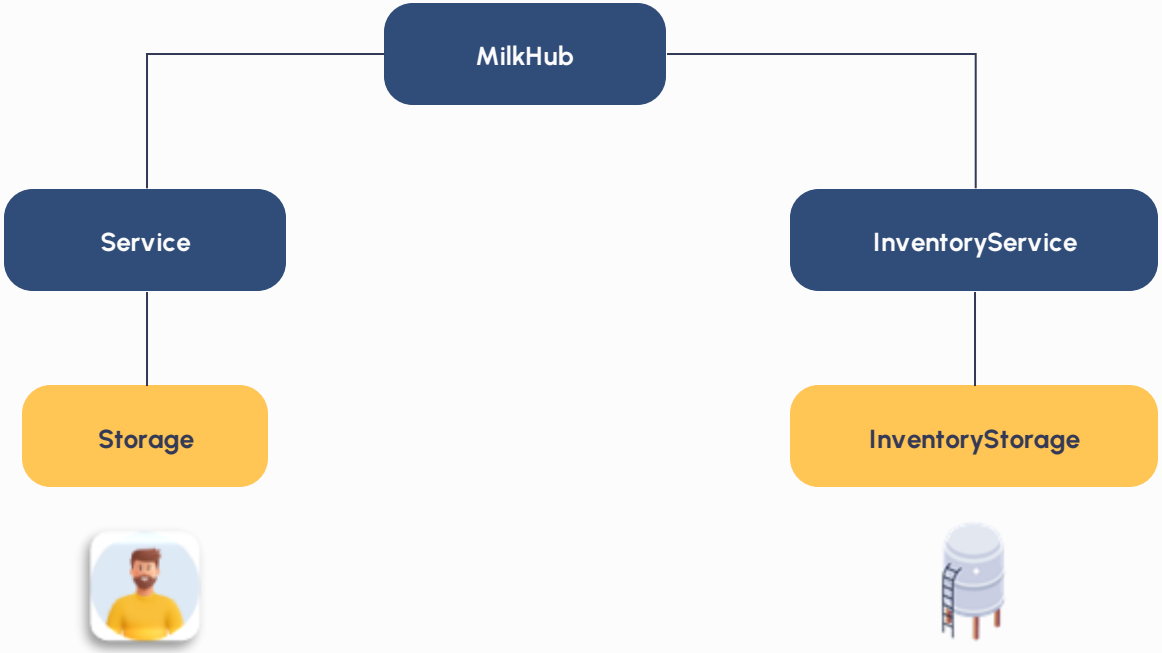
Stack Firefly



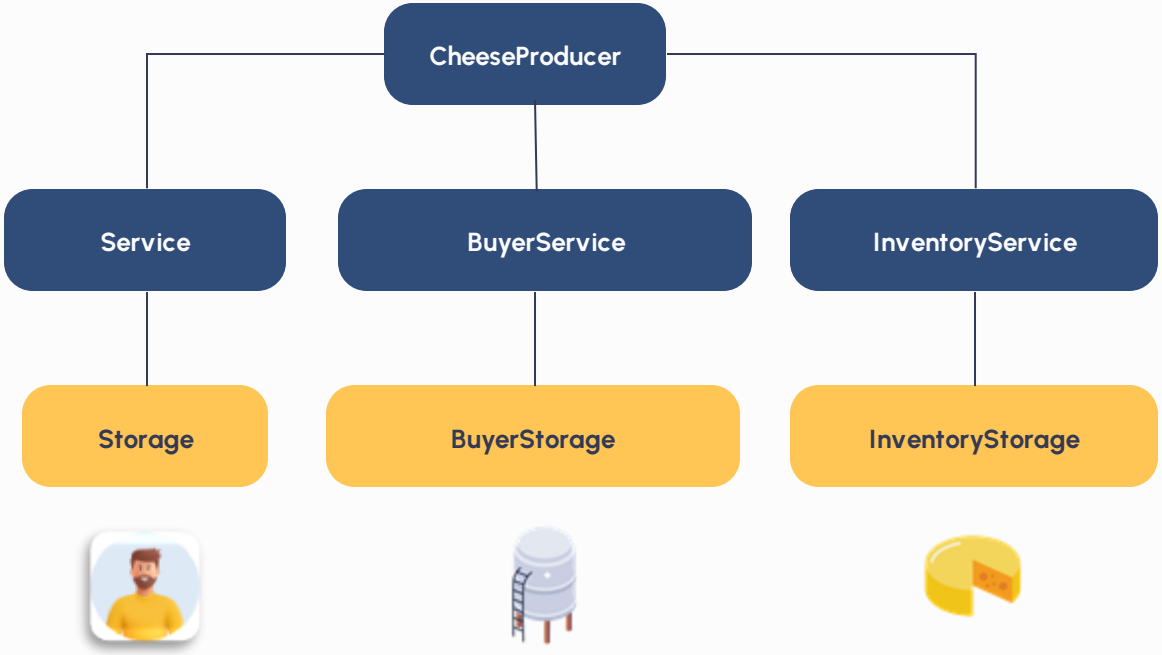
03

SMART CONTRACTS

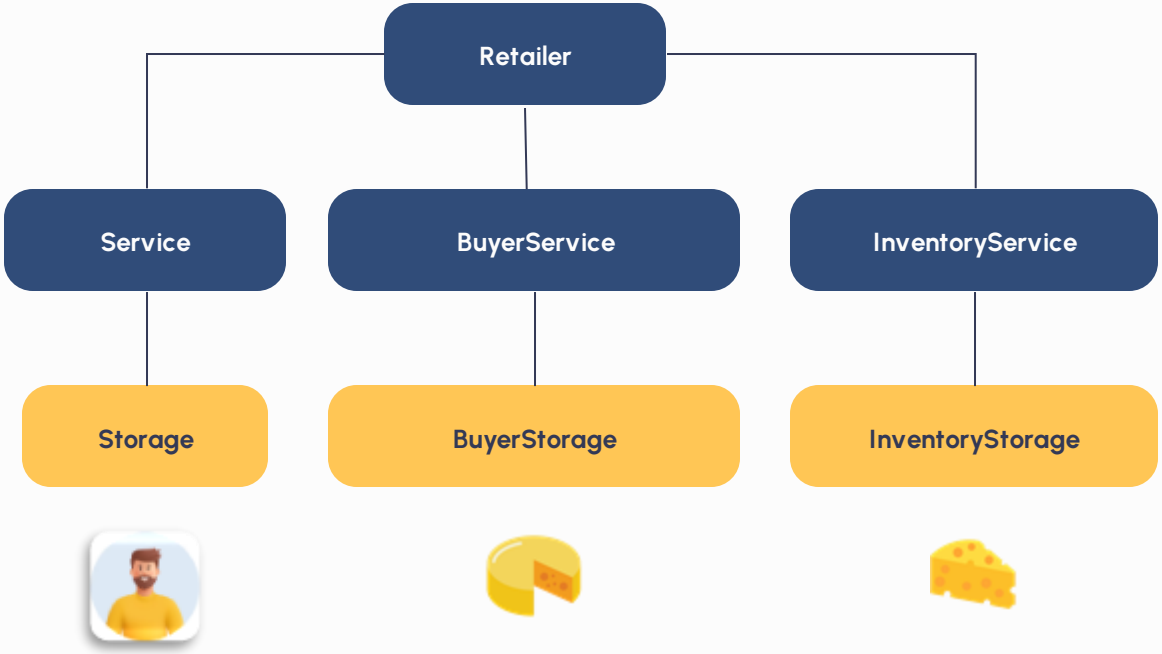
MICROSERVIZI: MILKHUB



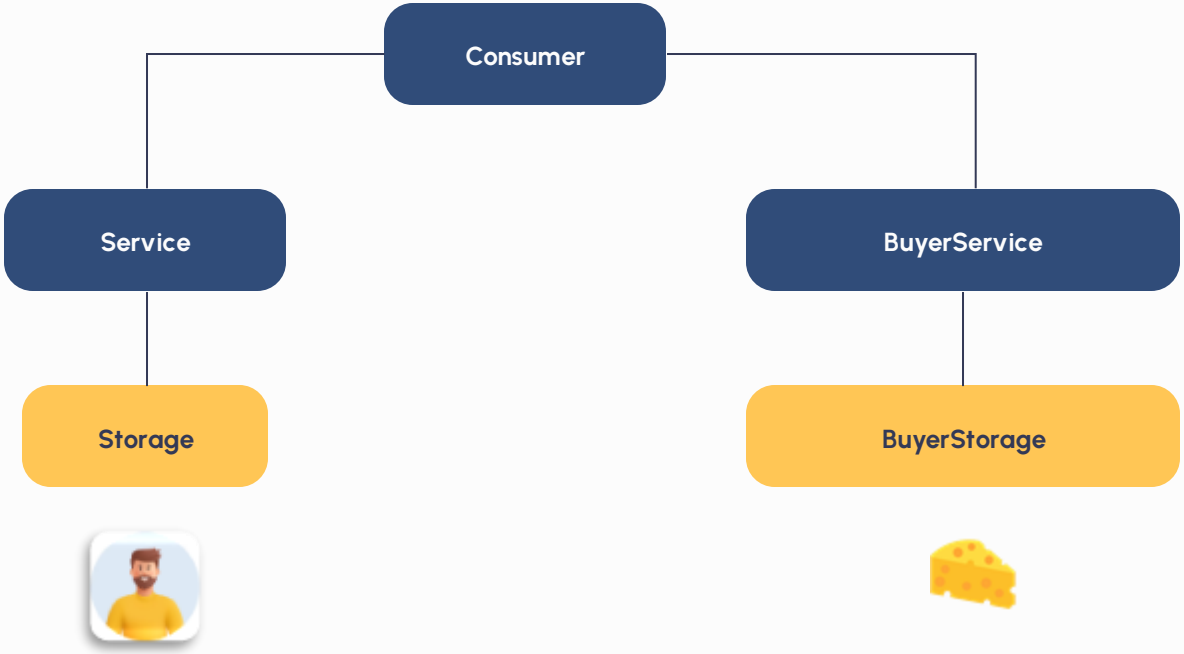
MICROSERVIZI: CHEESEPRODUCER



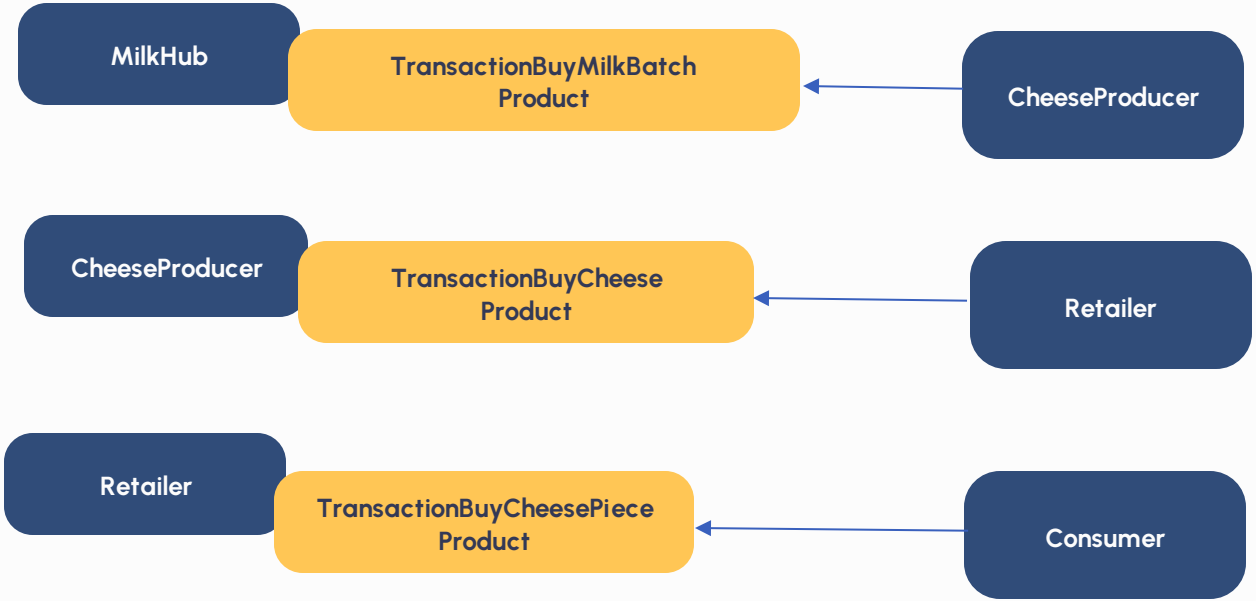
MICROSERVIZI: RETAILER



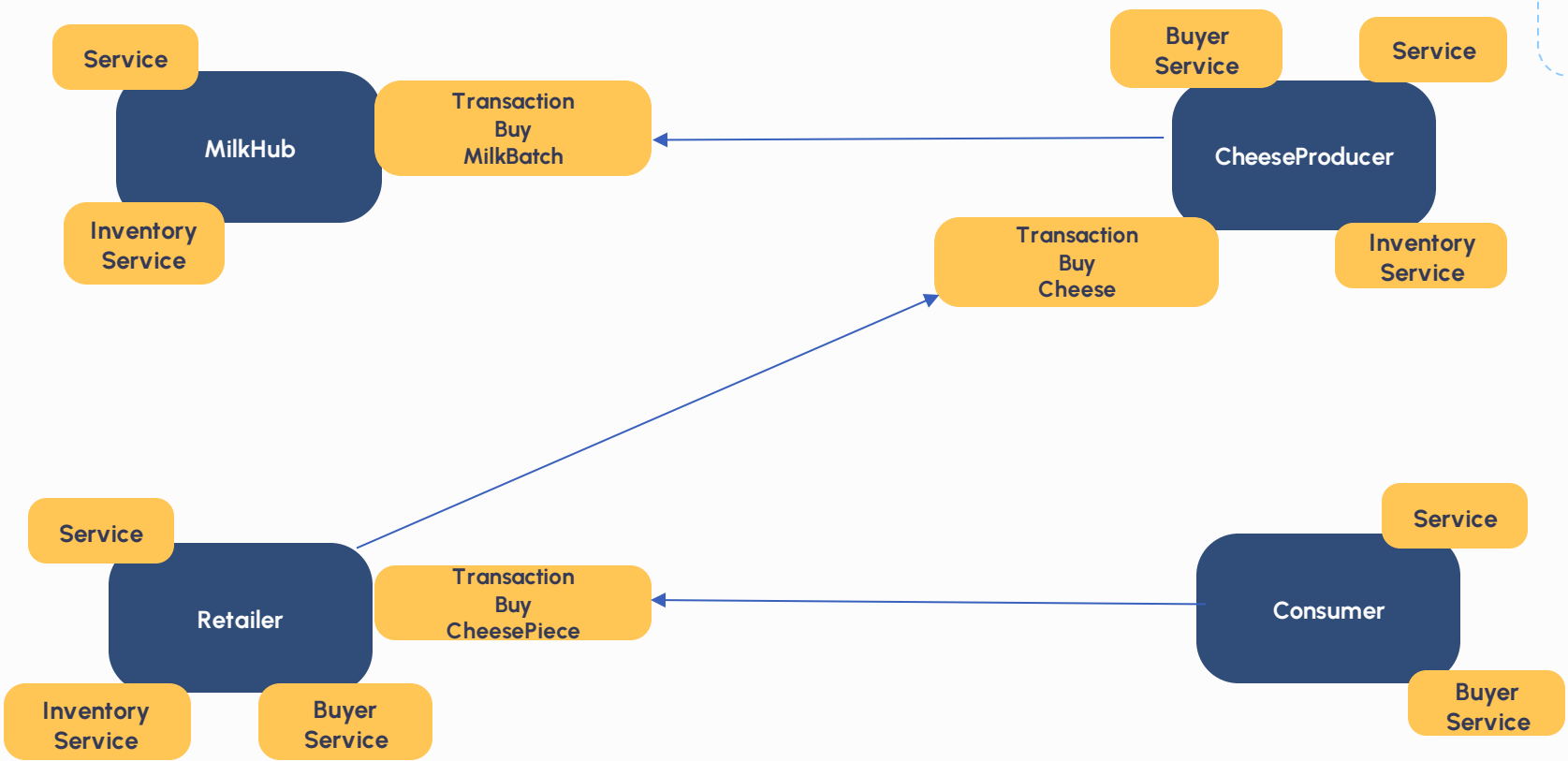
MICROSERVIZI: CONSUMER



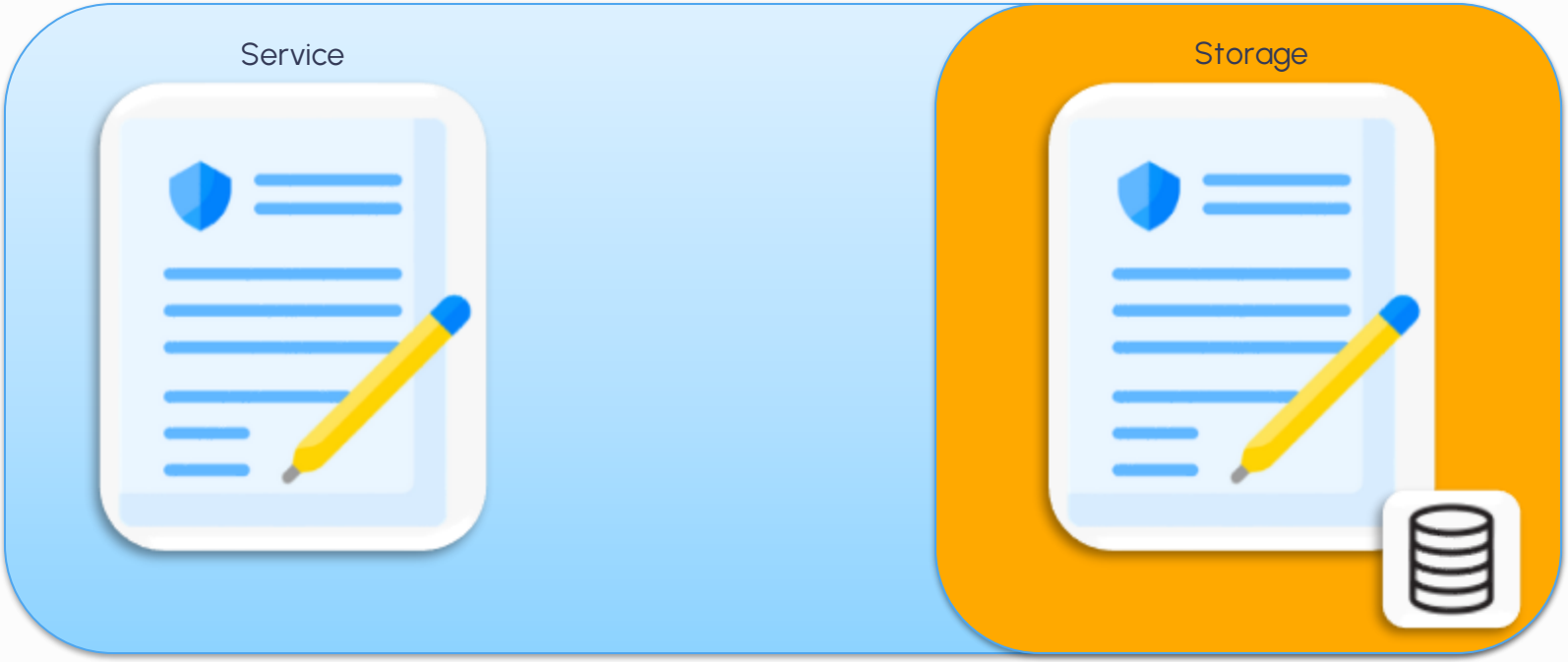
MICROSERVIZI: Transaction Manager



DApp: FILIERA-TOKEN



DESIGN PATTERN: ETERNAL STORAGE



04

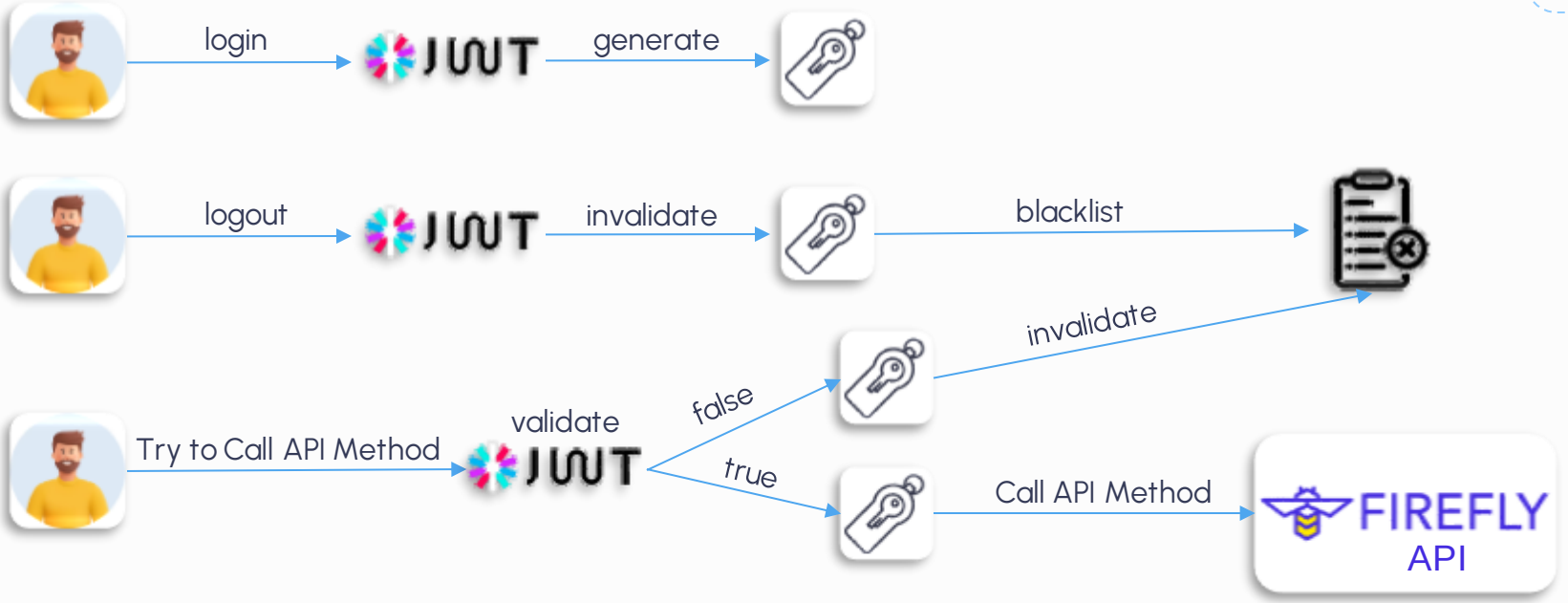
SICUREZZA

DESIGN PATTERN: SECURE ETHER TRANSFER

contracts > Service > Filieratoken.sol

```
contract Filieratoken is ERC20 {
    function approve(address spender, uint256 amount) public override returns (bool) {
        return super.approve(spender, amount);
    }

    /**
     * @dev Trasferisce `amount` token Filiera dall'account del mittente all'account del destinatario per l'acquisto di un prodotto.
     *
     * @param from L'indirizzo del mittente dei token.
     * @param to L'indirizzo del destinatario dei token.
     * @param amount Il numero di token Filiera da trasferire.
     *
     * @return `true` se il trasferimento ha avuto successo, `false` in caso contrario.
     */
    function transferTokenBuyProduct(address from, address to, uint256 amount) public returns (bool) {
        require(from != address(this) && to != address(this), "Transazione non valida per acquisto prodotto");
        super._transfer(from, to, amount);
        return true;
    }


    /**
     * @dev Assegna `amount` token Filiera all'account dell'utente specificato come incentivo alla registrazione.
     *
     * @param to L'indirizzo dell'utente che viene registrato.
     * @param amount Il numero di token Filiera da assegnare all'utente.
     *
     * @return `true` se la registrazione ha avuto successo, `false` in caso contrario.
     */
    function registerUserWithToken(address to, uint256 amount) public returns (bool) {
        require(msg.sender != address(this), "Solo il contratto FilieraToken puo' registrare utenti");
        super._transfer(address(this), to, amount);
        return true;
    }


}
```

JWT: JSON Web Token



BCrypt con Salt nascosto

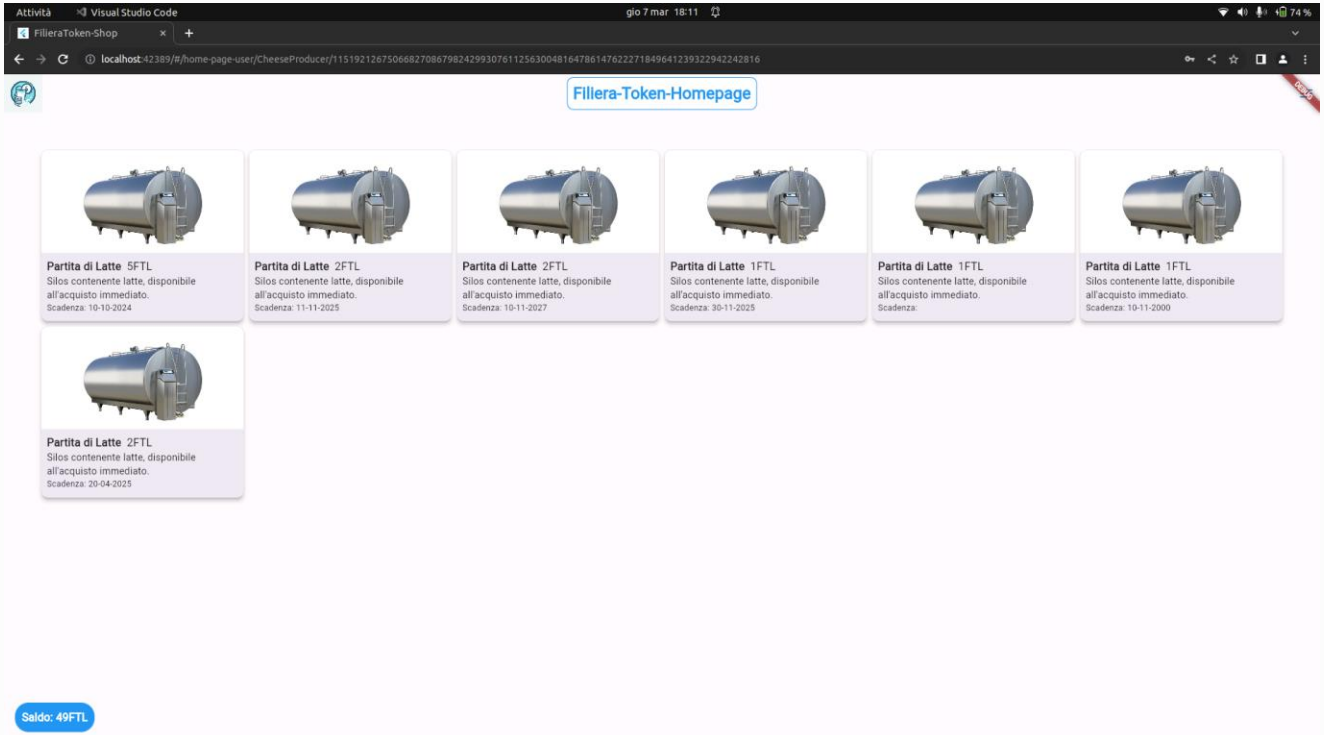
```
String salt = "\$2a\$10\$Gs.PmaGJQtm0ThQF3VkX2u";
```


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

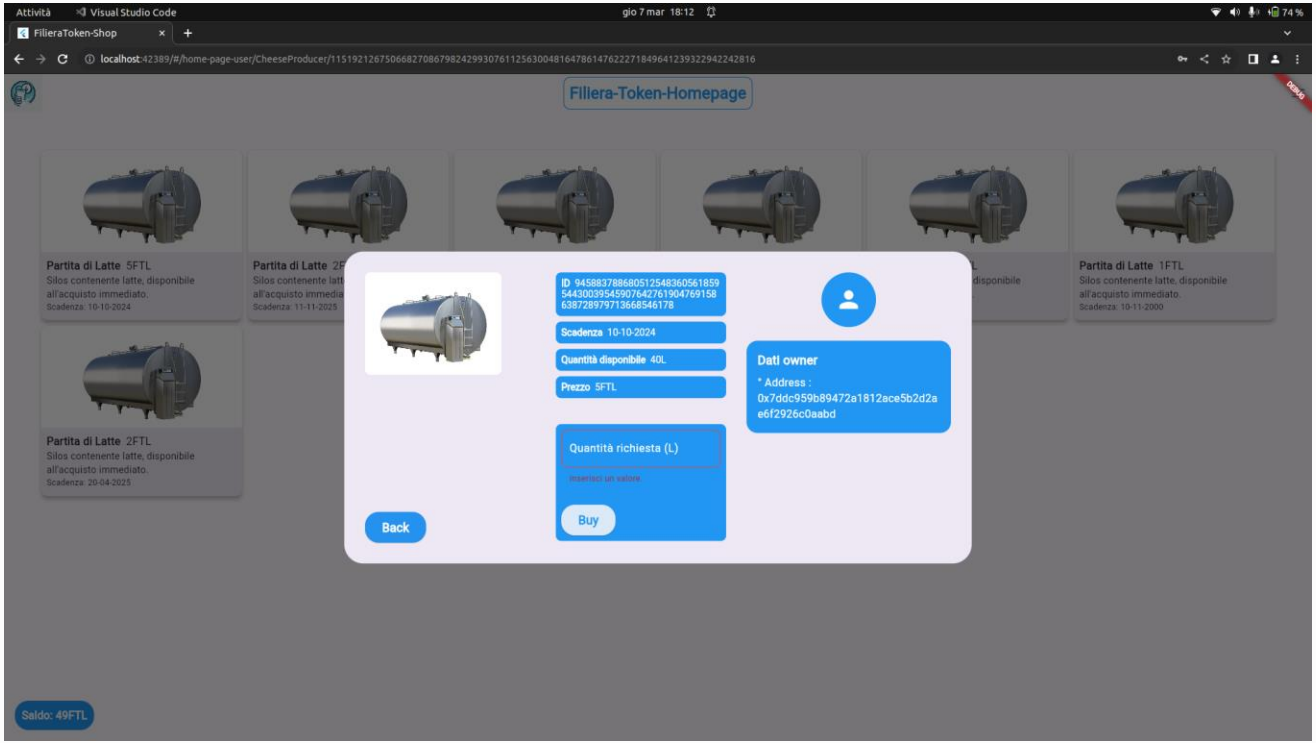
05

FUNZIONALITA' PRINCIPALI

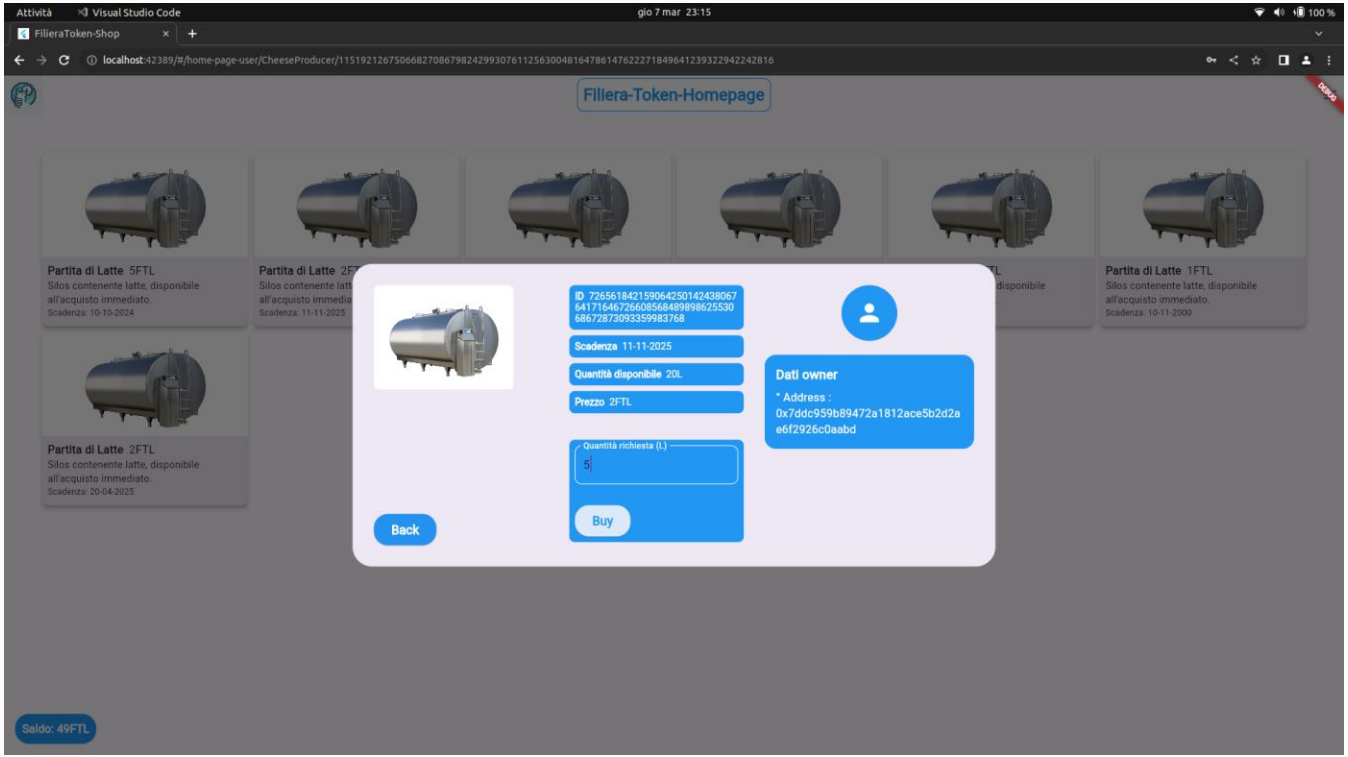
Transazione: Selezione prodotti



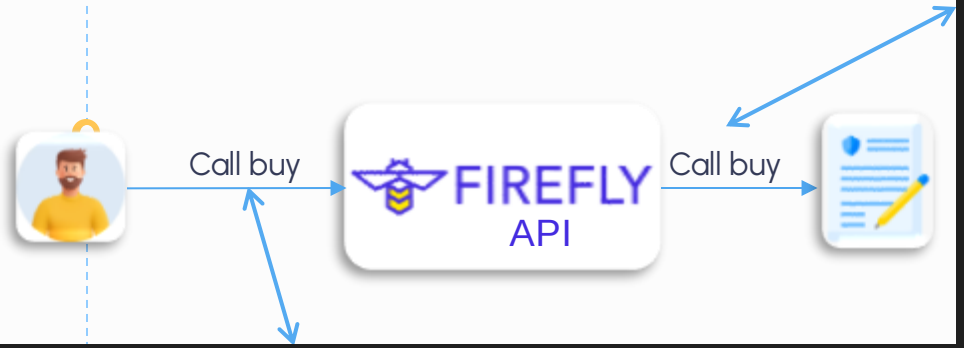
Transazione: Acquisto



Transazione: Acquisto



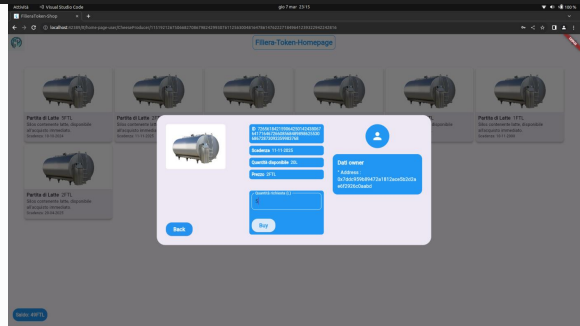
Transazione: Acquisto



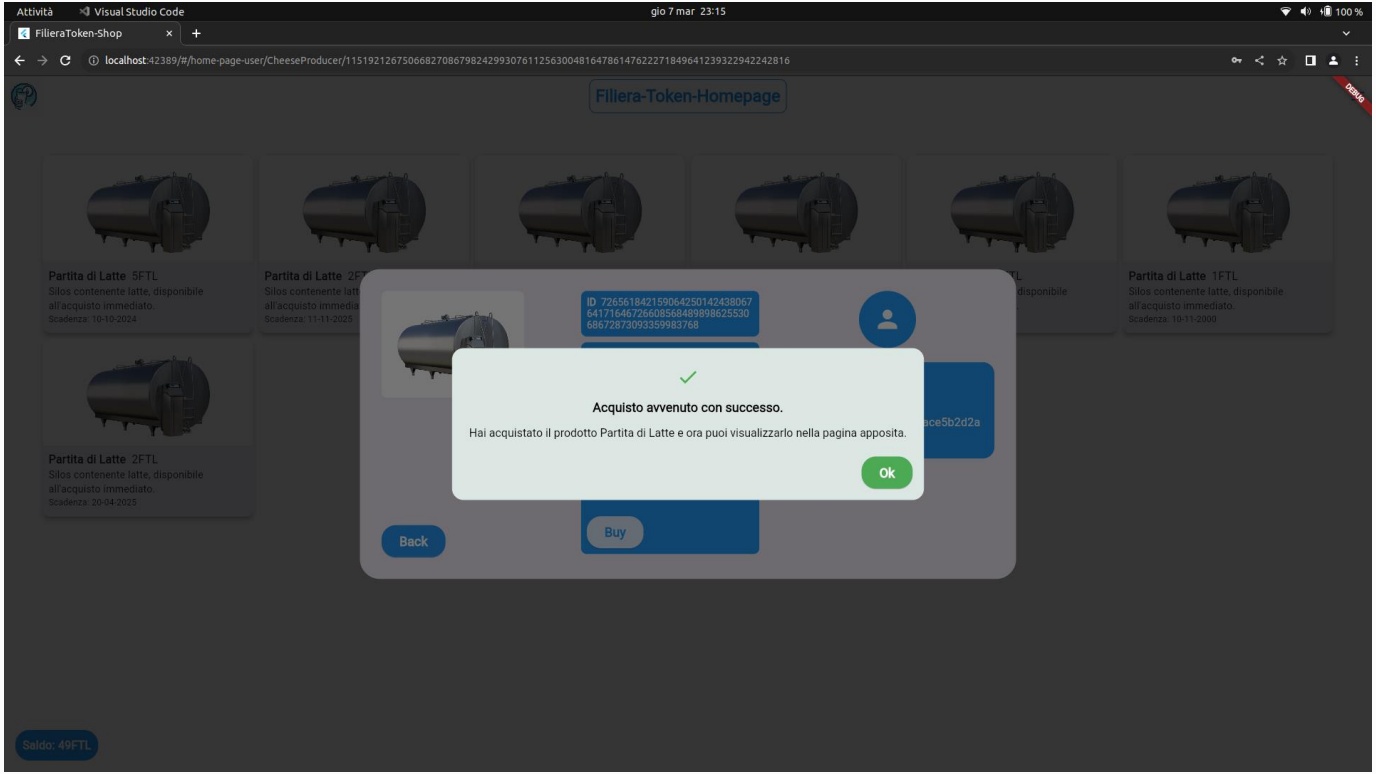
TransactionBuyMilkBatch Product

```
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");
    // Verfica 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);
}
```

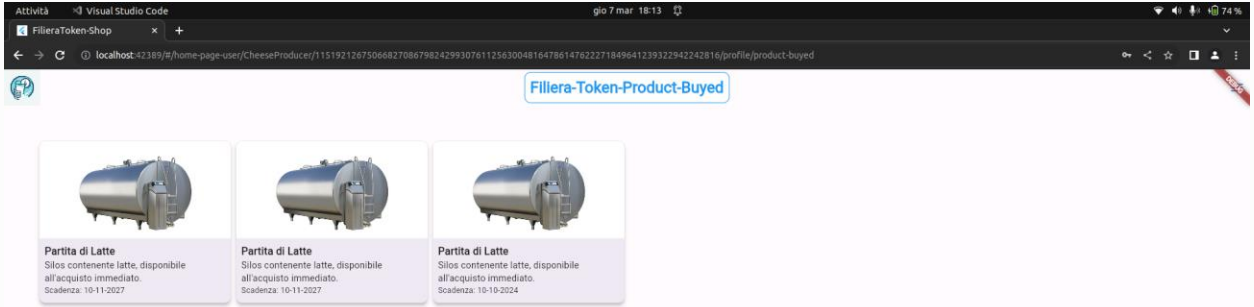
```
Future<bool> buyMilkBatchProduct(
  String milkBatchId,
  String quantityToBuy,
  String buyer,
  String ownerMilkBatch,
  String totalPrice,
) async {

  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;
  }
}
```



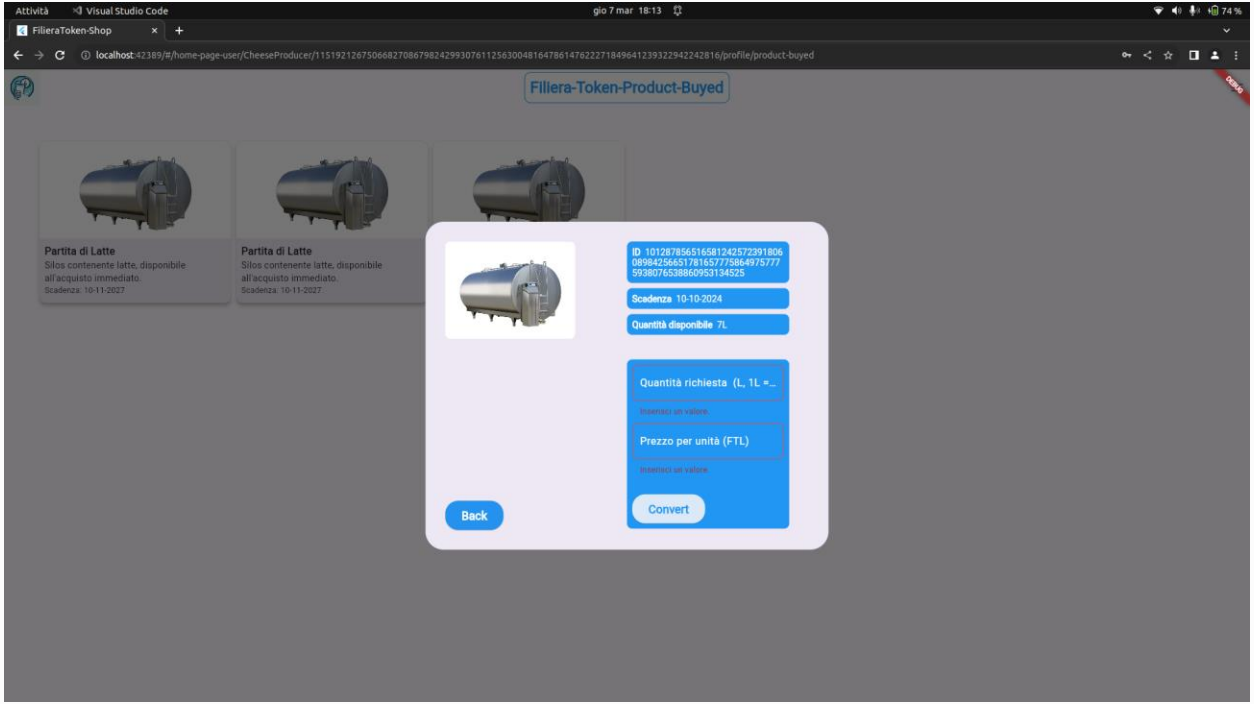
Transazione: Acquisto



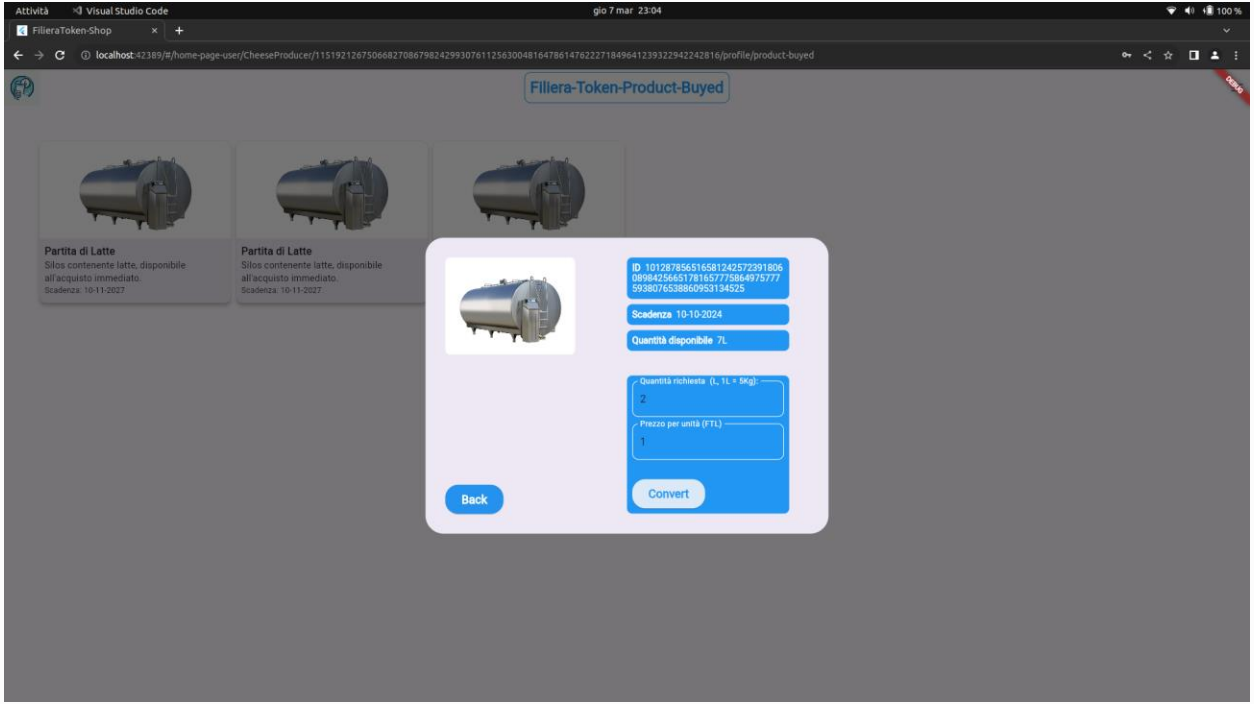
Conversione prodotti: Selezione



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),"Quantita' 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);
}
```

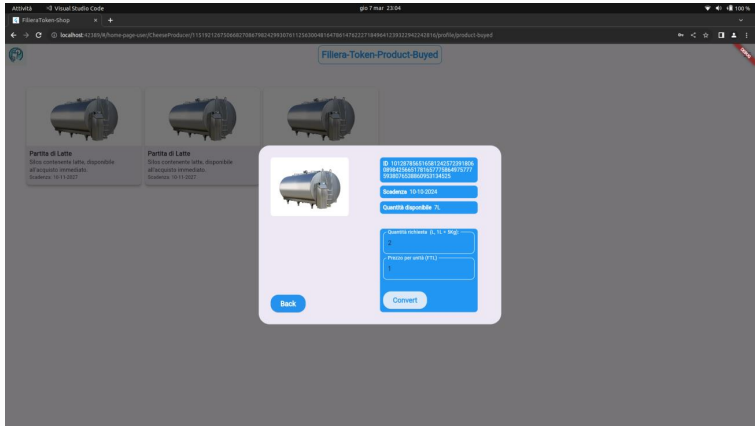
CheeseProducerInventoryService

Call transform

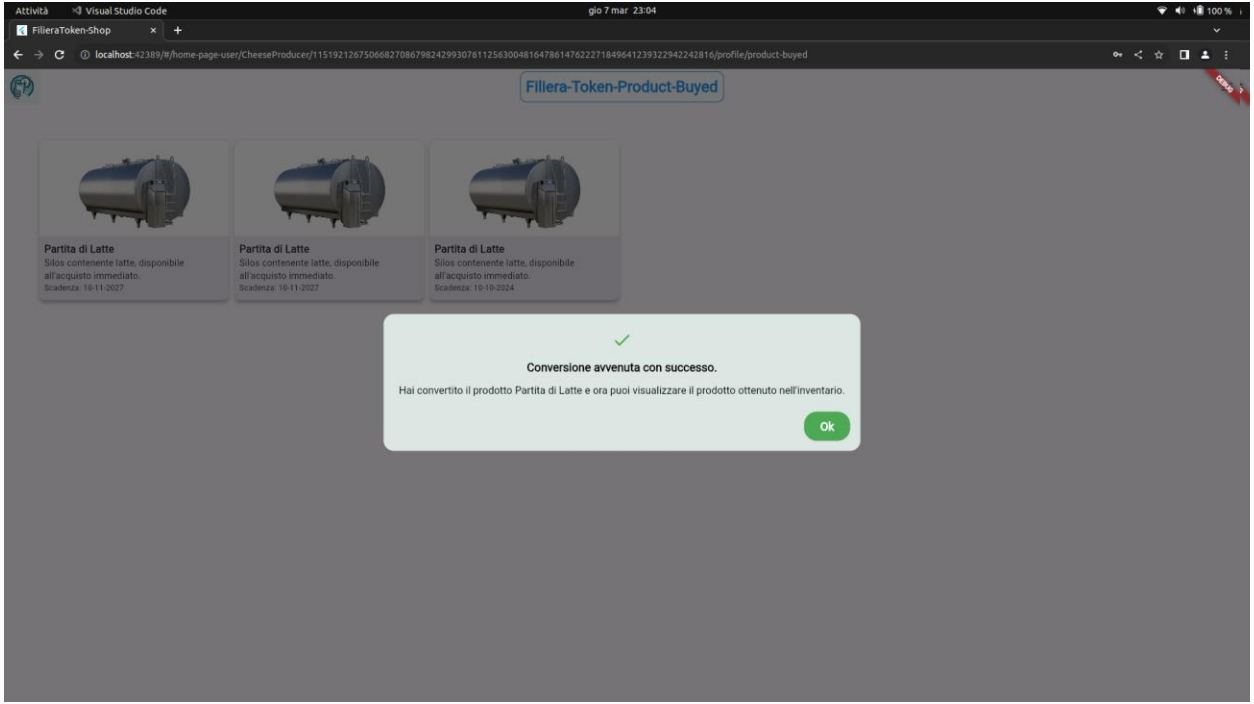
FIREFLY API

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

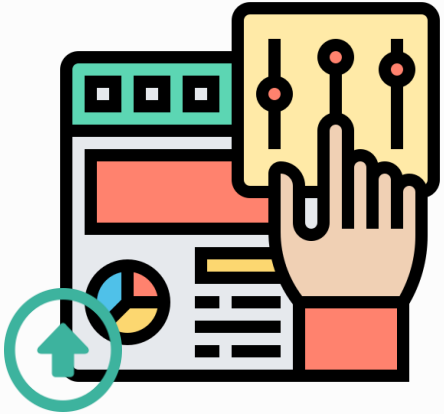


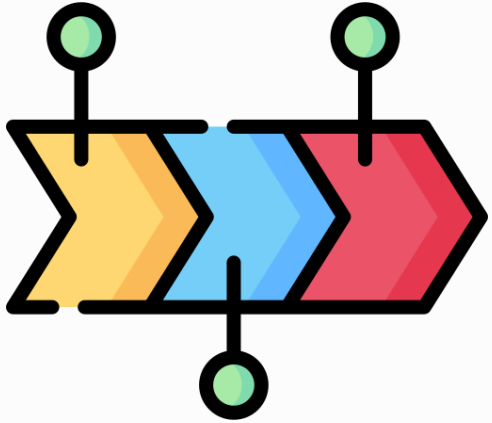
06

CONCLUSIONI

Sviluppi futuri

- Miglioramento della UI/UX
- Cronologia Eventi





GRAZIE PER L'ATTENZIONE!

Umberto Della Monica
Gerardo Leone

LinearIT
A HYA HOLDING COMPANY
UNIVERSITÀ DEGLI STUDI DI SALERNO