

WATCHCHAIN

Mario Lezzi

matr. 0522501840

Sicurezza dei Dati 25/26



<https://github.com/MarioLezzi92/Watchchain>

Problema

Contraffazione

Il mercato secondario degli orologi è invaso da falsi di alta qualità e orologi con parti interne sostituite e non originali.

Scarsa tracciabilità

Chi acquista non ha strumenti per verificare la storia dell'orologio e deve fidarsi ciecamente della parola del venditore.

Limiti del cartaceo

I documenti cartacei di autenticità si deteriorano, si perdono o possono essere facilmente falsificati, rendendo impossibile provare la proprietà.

Idea di soluzione

1. Filiera a ruoli

Producer → Reseller → Consumer, con permessi e responsabilità definite.

2. Identità digitale dell'orologio

Ogni orologio è rappresentato da un NFT che ne attesta identità e proprietà

3. Tracciabilità e trasparenza

Provenienza, certificazione e passaggi di mano registrati in modo immutabile on-chain.

4. Regole automatiche di scambio

Smart contract per vendite con escrow e pagamenti

Attori

Producer

Crea l'NFT e lo vende sul mercato primario ai Reseller.

Reseller

Acquista l'orologio sul primario, lo certifica e lo rivende sul mercato secondario.

Consumer

Acquista l'orologio sul mercato secondario.

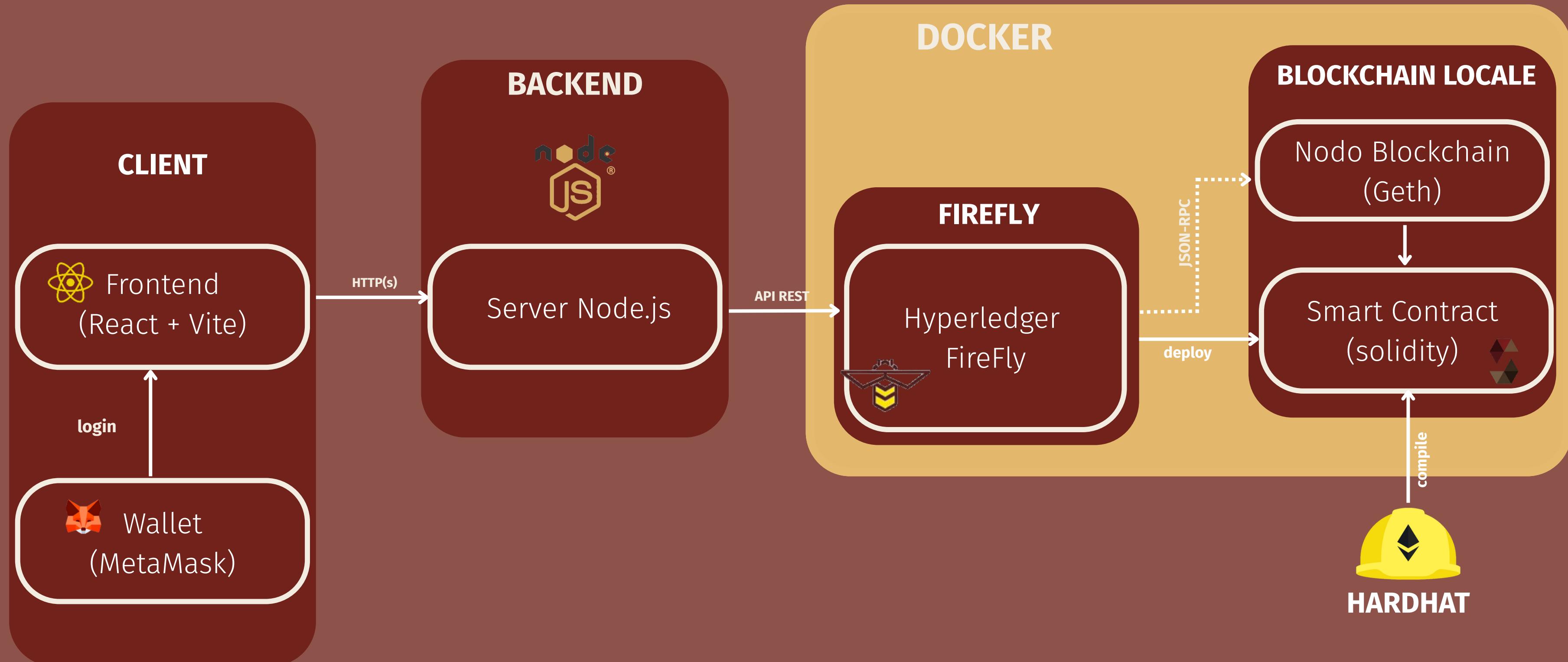
Requisiti Funzionali (RF)

ID	Attore	Requisito
RF_1	Producer	Minting orologio
RF_2	Producer	Abilitazione Reseller
RF_3	Producer	Disabilitazione Reseller
RF_4	Producer	Listing Primario
RF_5	Producer	Blocco Factory
RF_6	Producer	Blocco Marketplace
RF_7	Producer / Reseller	Visualizzazione mercato primario

Requisiti Funzionali (RF)

RF_8	Reseller	Acquisto primario
RF_9	Reseller	Certificazione
RF_10	Reseller	Listing secondario
RF_11	Producer/reseller	Annullamento listing
RF_12	Producer/reseller	Aggiorna prezzo
RF_13	Consumer	Visualizzazione mercato secondario
RF_14	Consumer	Acquisto finale
RF_15	Tutti	Visualizzazione inventario
RF_16	Producer/Reseller	Prelievo fondi

Stack Tecnologico



Infrastruttura blockchain

1. Rete Ethereum locale eseguita su un nodo Geth containerizzato in Docker.
2. FireFly espone API REST e interagisce con Geth tramite JSON-RPC.
3. Node.js valida le operazioni e interagisce con la blockchain tramite API REST di FireFly
4. Smart Contract compilati e testati con Hardhat e deployati sulla rete tramite FireFly.

Hyperledger FireFly

Tre nodi FireFly, ciascuno associato ad un attore.

Network Nodes		
NAME	NODE ID	ORG OWNER
ProducerNode	31f4c...0b29c	did:firefly:node/ProducerNode
ResellerNode	e5a03...09ee1	did:firefly:node/ResellerNode
ConsumerNode	81da3...9557d	did:firefly:node/ConsumerNode



Hyperledger FireFly

- Definizione dei metodi e degli eventi a partire dall'ABI dello Smart Contract.
- Generazione di endpoint REST per l'invocazione delle funzioni on-chain.
- Listeners per intercettare eventi.

Register a Contract API
Generates an interactive HTTP API from your contract

Contract Interface *
WatchMarket_IFACE - 1.1

Name *
WatchMarket_API

Address *
0xcf7ed3acca5a467e9e704c703e8d87f634fb0fc9

The contract address on your blockchain.

Publish
Controls whether to automatically publish the contract API to the network.

Define a Contract Interface
Defines all methods and events of your contract to FireFly

Interface Format *
FFI - FireFly Interface

Either FFI or Solidity ABI format

Schema *
{
 "name": "my-contract",
 "version": "1.0",
 "methods": [
 {
 "name": "method1"
 }
],
 "events": [
 {
 "name": "event1"
 }
]
}

Register a Contract Listener
Instructs FireFly to listen for and index events from your contract

Contract API *
WatchMarket_API

Events *
Paused

Topic *
app-events

A FireFly identifier to group events across multiple listeners.

Name
A custom name for your listener

First Event
newest

The first event for this listener to index. Valid options are 'newest', 'oldest', or a specific block number.



Hyperledger FireFly

Token pool per il mapping degli asset on-chain,
abilitando il monitoraggio in tempo reale dei
wallet e lo storico delle transazioni.

Token Pools / **watchnft**

watchnft

ID	367b34b3-94c4-4dfe-a694-53539d88dea2
Standard	ERC721 (nonfungible)
Connector	erc20_erc721
Locator	address=0x546a240ffbe4f6f0a4de7eca5d13da5591572b00&schema=ERC721NoData&typ...
Transaction ID	868a9ef5-b396-4afe-9bca-8e92334b31ab
Message ID	025370e0-c359-4490-aa5c-48565839936e
State	Active
Created	01/24/2026 10:52:06,533 AM (13 days ago)

Pool Info

```
{
  "address": "0x546a240ffbe4f6f0a4de7eca5d13da5591572b00",
  "name": "WatchNFT",
  "schema": "ERC721NoData"
}
```

Transfers In Pool

ACTIVITY	FROM	TO	AMOUNT	BLOCKCHAIN EVENT	SIGNING KEY	TIMESTAMP
Transfer	0xe5c..bd1cf	0x005..cf921	1	d3a83..de67f	0x005..cf921	2 hours ago

Balances for 0x0eb...602b9

watchnft (Non-Fungible) Total: 6 (#1, #2, #3, #4, #5, #6)
luxurycoin (Fungible) --- Total: 980013

Create a Token Pool

Defines a new set of tokens for FireFly to index

Pool Name *

Pool Symbol

Type * Fungible

Contract Address



Smart Contract



LuxuryCoin (LUX)

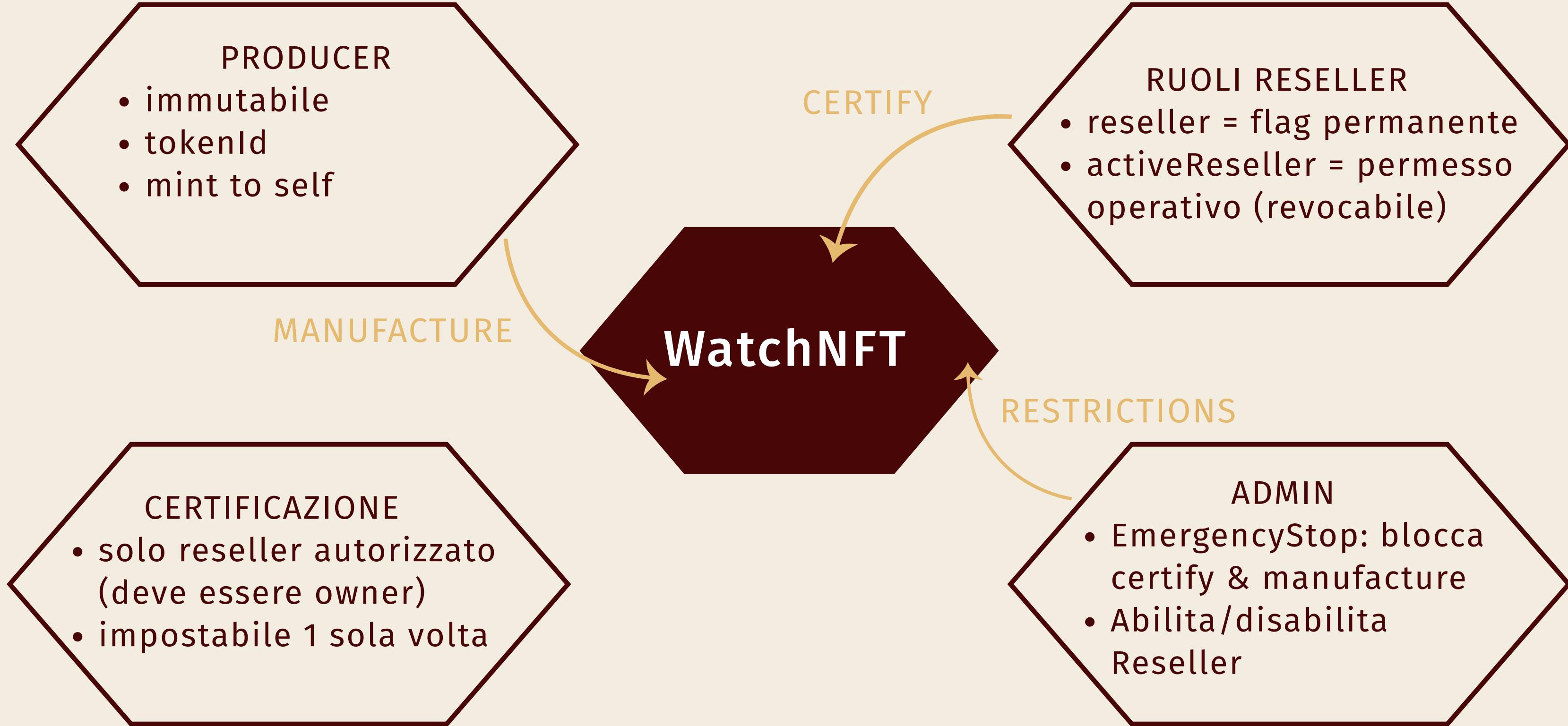
Token ERC-20 utilizzato come valuta interna di Watchain per standardizzare e regolare tutte le transazioni economiche on-chain.

Pagamenti sicuri tramite il meccanismo di *allowance* che autorizza il marketplace a trasferire i fondi.

Supply fissa definita al deploy: 1.000.000 LUX distribuita in modo strategico tra gli attori del sistema.



WatchNFT



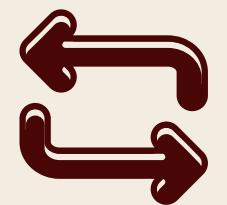
WatchMarket

Gestisce la compravendita dei WatchNFT .

- Modalità:
1. Mercato primario (solo Producer vende, solo Reseller autorizzati acquistano)
 2. Mercato secondario (solo Reseller autorizzato e proprietario vende, NFT deve essere certificato, solo Consumer acquistano).



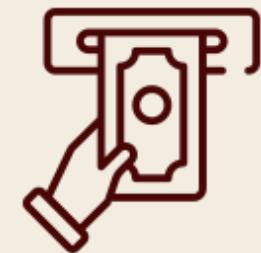
Per pubblicare un annuncio il venditore deve autorizzare il marketplace a trasferire l'NFT



l'NFT viene trasferito in escrow a WatchMarket e resta custodito fino a vendita o annullamento



Pagamento in LUX con allowance, che autorizza il marketplace a trasferire i fondi all'acquirente



Il ricavo non viene inviato subito al venditore ma è registrato come credito interno e riscattato successivamente



PulloverPush

Problema:

I trasferimenti automatici durante la vendita sono rischiosi. Se il destinatario rifiuta i fondi, l'intera transazione di acquisto fallisce.

SOLUZIONE

Separazione netta tra vendita e incasso.
Il contratto non invia token, ma aggiorna solo un saldo virtuale interno associato all'utente.

IMPLEMENTAZIONE

Il venditore deve chiamare withdraw() per incassare

SICUREZZA

Il saldo virtuale viene azzerato prima del trasferimento per evitare attacchi di Re-entrancy.

Il tuo Profilo

RUOLO

Producer

INDIRIZZO

0x0eb57516109f4a7e9e5706a1b45344be1d602b9

SALDO

980013 LUX

VENDITE DA INCASSARE

8 LUX

PRELEVA ORA



EmergencyStop

OBIETTIVO

Bloccare le attività critiche in caso di bug o attacchi



SOLUZIONE
congelamento delle operazioni critiche

IMPLEMENTAZIONE
modificatore whenNotPaused

EXIT STRATEGY
in emergenza gli utenti possono recuperare gli NFT in escrow e prelevare i fondi accumulati



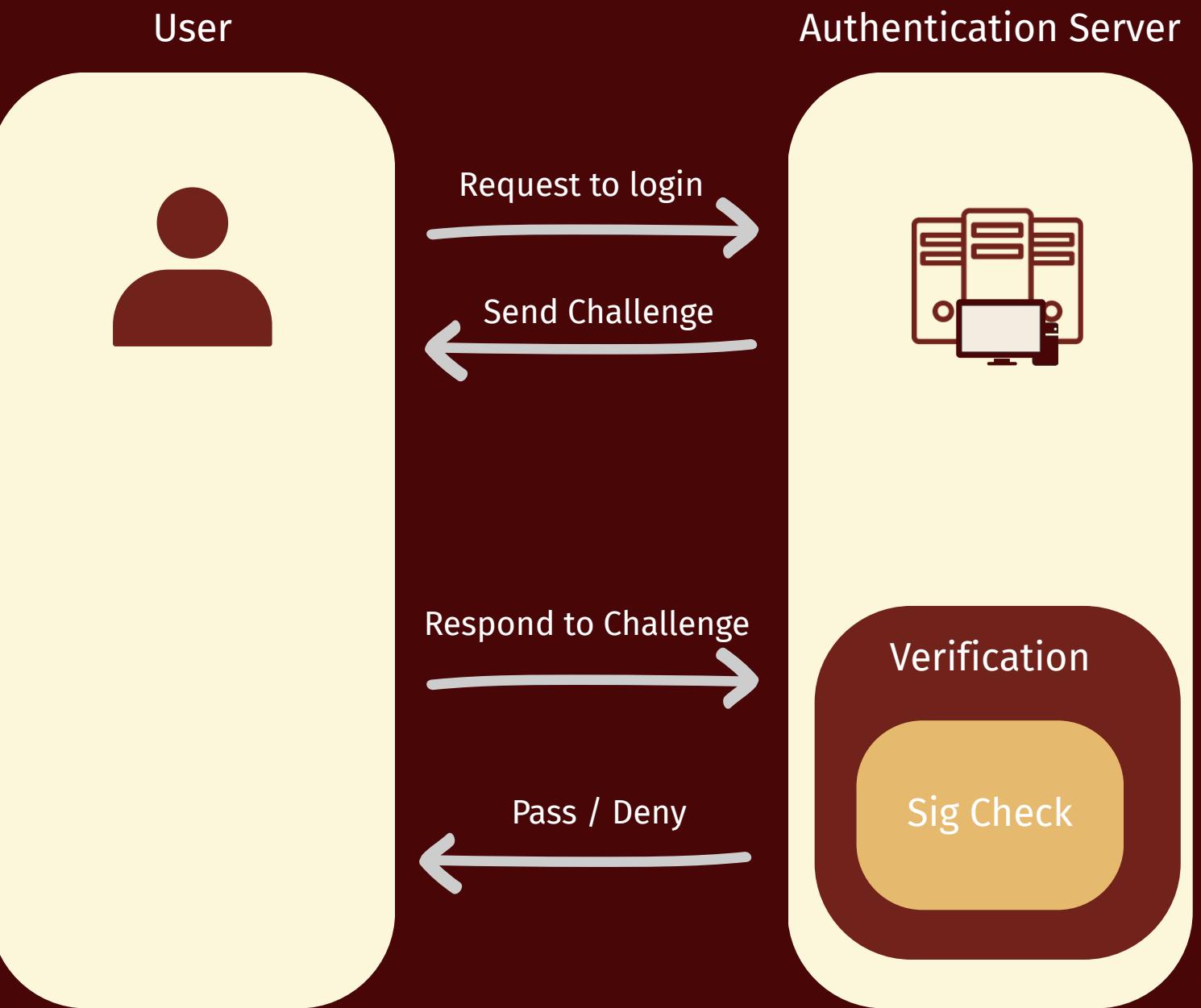
Autenticazione

Login passwordless: No DB per lo storage di credenziali. Identità garantita dal possesso della chiave privata del wallet MetaMask.

Challenge anti-replay: Il server genera un Nonce casuale monouso con TTL.

Firma off-chain: l'utente firma nonce crittograficamente, attraverso il proprio wallet.

Il server **verifica** la validità della firma per autenticare l'accesso. Se valida, rilascia i Cookie HttpOnly.



Gestione sessione

Architettura Stateless

- Tutte le informazioni necessarie sono nel Cookie (JWT).
- Il server verifica la validità del JWT

Sicurezza **HttpOnly**

- Protezione XSS: I token sono encapsulati in *Cookie HttpOnly*, invisibili al motore JavaScript del browser.

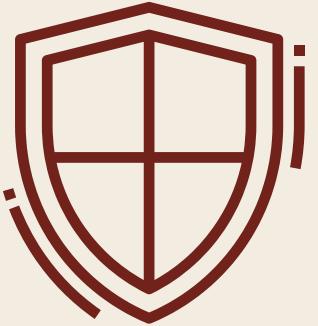
Dual Token

- Access Token (15m): finestra d'attacco ridotta.
- Refresh Token (14gg) + rotation: rinnovo sessione senza richiedere firma all'utente.

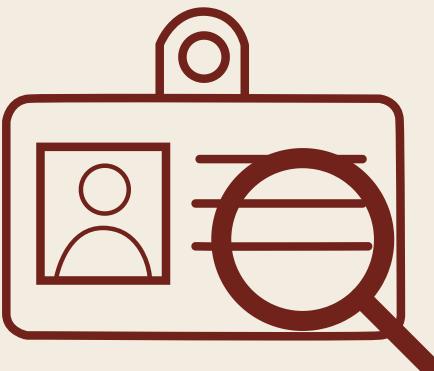
Autorizzazione



RBAC Immutabile: La gestione dei ruoli è demandata agli Smart Contract. Regole di accesso scritte su Blockchain.



Trusted Gateway: Il Server valida le operazioni di scrittura e i permessi dell'utente prima di inoltrarle a FireFly.



Anti-Spoofing: L'identità viene estratta forzatamente dal Token JWT firmato, prevenendo l'impersonificazione.

Sincronizzazione Event-Driven

1. FireFly Listener

Gli eventi emessi dagli Smart Contract vengono intercettati da FireFly che invia una notifica al server.

3. Notifica real-time

Il server notifica i client connessi tramite *WebSocket* quando viene rilevato un nuovo evento.

2. Controllo

Ogni notifica include un messaggio segreto noto solo a FireFly e al server, che ne verifica l'autenticità.

4. Sincronizzazione

Il frontend riceve la notifica e richiede i dati aggiornati a FireFly che li recupera dalla blockchain.

Use Case

UC_01: Autenticazione utente

ID/Nome	UC_01: Autenticazione utente
Attore	Producer, Reseller, Consumer
Entry Condition	<ul style="list-style-type: none">L'utente possiede un wallet associato ad un indirizzo blockchain.
Flusso	<ol style="list-style-type: none">L'utente richiede l'accesso al sistema.Il sistema genera un messaggio univoco temporaneo, associato all'indirizzo dell'utente.L'utente firma il valore utilizzando il proprio wallet.L'utente invia la firma al sistema.Il sistema verifica la validità della firma e l'associazione con l'indirizzo dell'utente.Il sistema autentica l'utente e avvia una sessione valida.
Exit Condition (on success)	L'utente risulta autenticato e può accedere alle funzionalità consentite dal proprio ruolo.
Exit Condition (on failure)	L'utente non ha accesso al sistema.
Flusso alternativo	<ul style="list-style-type: none">Valore scaduto: il sistema richiede una nuova autenticazione.Firma non valida: il sistema rifiuta l'autenticazione.Utente non autorizzato: il sistema limita o nega l'accesso alle funzionalità.

UC_02: Minting Orologio

ID/Nome	UC_02: Minting
Attore	Producer
Entry Condition	<ul style="list-style-type: none">• Il Producer è autenticato.• Il sistema non è in stato di emergenza per le operazioni di creazione.
Flusso	<ol style="list-style-type: none">1. Il Producer richiede la creazione di un nuovo orologio al sistema, attraverso l'apposito pulsante.2. Il sistema, dopo aver richiesto conferma all'utente, valida l'autorizzazione del Producer per l'operazione di creazione.3. Il sistema crea l'orologio assegnandogli un identificativo univoco.4. Il sistema conferma l'avvenuta creazione.
Exit Condition (on success)	Un nuovo orologio digitale viene creato e registrato sulla blockchain. L'orologio risulta di proprietà del Producer ed è disponibile nel suo inventario.
Exit Condition (on failure)	Il sistema rifiuta l'operazione.
Flusso alternativo	Se le operazioni di creazione sono sospese, il sistema non permette di utilizzare il pulsante di mint.

UC_03: Gestione operatività dei Reseller

ID/Nome	UC_03: Gestione Reseller
Attore	Producer
Entry Condition	<ul style="list-style-type: none">• Il Producer è autenticato.
Flusso	<ol style="list-style-type: none">1. Il Producer richiede attraverso l'apposita interfaccia, la gestione di un Reseller, indicando l'indirizzo wallet e l'operazione desiderata (abilitazione o revoca).2. Il sistema richiede conferma.3. Il sistema aggiorna lo stato del Reseller.4. Il sistema conferma l'avvenuto aggiornamento.
Exit Condition (on success)	Il Reseller ottiene/perde i permessi.
Exit Condition (on failure)	Aggiornamento non riuscito. Lo stato del Reseller rimane invariato.
Flusso alternativo	//

UC_04: Vendita e acquisto sul mercato primario

ID/Nome	UC_04: Mercato primario
Attore	Producer, Reseller
Entry Condition	<ul style="list-style-type: none">• Il Producer è autenticato ed è proprietario di uno o più orologi.• Il Reseller è autenticato ed è abilitato come Reseller attivo.• Le operazioni di mercato primario non sono sospese per emergenza del sistema.
Flusso	<ol style="list-style-type: none">1. Il Producer seleziona un orologio dal proprio inventario e richiede di metterlo in vendita sul mercato primario indicando un prezzo.2. Il sistema richiede conferma all'utente.3. Il sistema verifica che l'orologio appartenga al Producer e che la vendita sia consentita.4. Il sistema rende l'orologio disponibile sul mercato primario.5. Il Reseller seleziona un orologio dal mercato primario e richiede l'acquisto.6. Il sistema richiede conferma all'utente.7. Il sistema verifica che il Reseller sia autorizzato ad acquistare sul mercato primario.8. Il sistema registra l'acquisto e trasferisce la proprietà dell'orologio al Reseller.
Exit Condition (on success)	<ul style="list-style-type: none">• L'orologio viene trasferito dal Producer al Reseller.• Il Producer matura un credito corrispondente al prezzo di vendita.• L'orologio non risulta più essere sul mercato primario.
Exit Condition (on failure)	<ul style="list-style-type: none">• Il Producer ha inserito un prezzo non valido. Se il prezzo è ≤ 0, il sistema impedisce il listing.• Errore durante la transazione: il sistema non completa il trasferimento e mantiene lo stato precedente.• Utente non autorizzato: il sistema impedisce l'operazione.
Flusso alternativo	Se il blocco del mercato è attivo, il sistema non permette la compravendita di orologi

UC_05: Certificazione orologio

ID/Nome	UC_05: Certificazione
Attore	Reseller
Entry Condition	<ul style="list-style-type: none">Il Reseller è autenticato.Il Reseller è autorizzato ad operare come tale.Il Reseller è proprietario dell'orologio da certificare.Le operazioni di certificazione non sono sospese per lo stato di emergenza.
Flusso	<ol style="list-style-type: none">Il Reseller seleziona un orologio dal proprio inventario.Il Reseller richiede la certificazione dell'orologio.Il sistema richiede conferma al Reseller.Il sistema verifica che il Reseller sia autorizzato e proprietario dell'orologio.Il sistema registra la certificazione dell'orologio.Il sistema conferma l'avvenuta certificazione.
Exit Condition (on success)	<ul style="list-style-type: none">L'orologio risulta certificato dal Reseller.Lo stato di certificazione dell'orologio è aggiornato e visibile.
Exit Condition (on failure)	Il sistema rifiuta la richiesta di certificazione. Lo stato dell'orologio rimane invariato.
Flusso alternativo	<ul style="list-style-type: none">Operazioni di certificazioni sospese: il sistema segnala che la funzionalità non è disponibile.Il Reseller non è abilitato: il sistema rifiuta l'operazione di certificazione.

ID/Nome	UC_06: Mercato secondario
Attore	Reseller, Consumer
Entry Condition	<ul style="list-style-type: none"> Il Reseller è autenticato ed è autorizzato ad operare come tale. Il Reseller è proprietario di uno o più orologi certificati. Il Consumer è autenticato. Le operazioni di mercato non sono sospese per lo stato di emergenza del sistema.
Flusso	<ol style="list-style-type: none"> Il Reseller seleziona un orologio certificato dal proprio inventario e richiede di metterlo in vendita sul mercato secondario indicando un prezzo. Il sistema richiede la conferma al Reseller. Il sistema verifica che l'orologio sia certificato e di proprietà del Reseller. Il sistema rende l'orologio disponibile sul mercato secondario. Il Consumer seleziona un orologio dal mercato secondario e richiede l'acquisto. Il sistema richiede la conferma al Consumer. Il sistema verifica che l'operazione rispetti i requisiti di vendita. Il sistema registra l'acquisto e trasferisce la proprietà dell'orologio al Consumer.
Exit Condition (on success)	<ul style="list-style-type: none"> Un orologio viene trasferito dal Reseller al Consumer. Il Reseller matura un credito corrispondente al prezzo di vendita. L'orologio non risulta più essere sul mercato secondario.
Exit Condition (on failure)	<ul style="list-style-type: none"> Il Reseller ha inserito un prezzo non valido. Se il prezzo è ≤ 0, il sistema impedisce il listing. Errore durante la transazione: il sistema non completa il trasferimento e mantiene lo stato precedente. Utente non autorizzato: il sistema impedisce l'operazione.
Flusso alternativo	Se il blocco del mercato è attivo, il sistema non permette la compravendita di orologi.

UC_06: Vendita e acquisto sul mercato secondario

UC_07: Pagamenti e prelievo dei fondi

ID/Nome	UC_07: Pagamenti e prelievo fondi
Attore	Producer, Reseller
Entry Condition	<ul style="list-style-type: none">L'utente è autenticato.L'utente ha maturato crediti a seguito di una o più vendite.
Flusso	<ol style="list-style-type: none">L'utente richiede il prelievo dei fondi maturati.Il sistema richiede la conferma all'utente.Il sistema verifica che l'utente abbia un saldo disponibile.Il sistema trasferisce i fondi all'utente.Il sistema aggiorna il saldo dei crediti.
Exit Condition (on success)	<ul style="list-style-type: none">I fondi maturati vengono trasferiti all'utente.Il saldo dei crediti dell'utente viene aggiornato.
Exit Condition (on failure)	<ul style="list-style-type: none">Nessun credito disponibile: il sistema non mostra il tasto per il prelievo.Errore durante il trasferimento: il sistema non aggiorna il saldo e segnala il problema.
Flusso alternativo	//

UC_08: Aggiornamento in tempo reale tramite eventi Blockchain

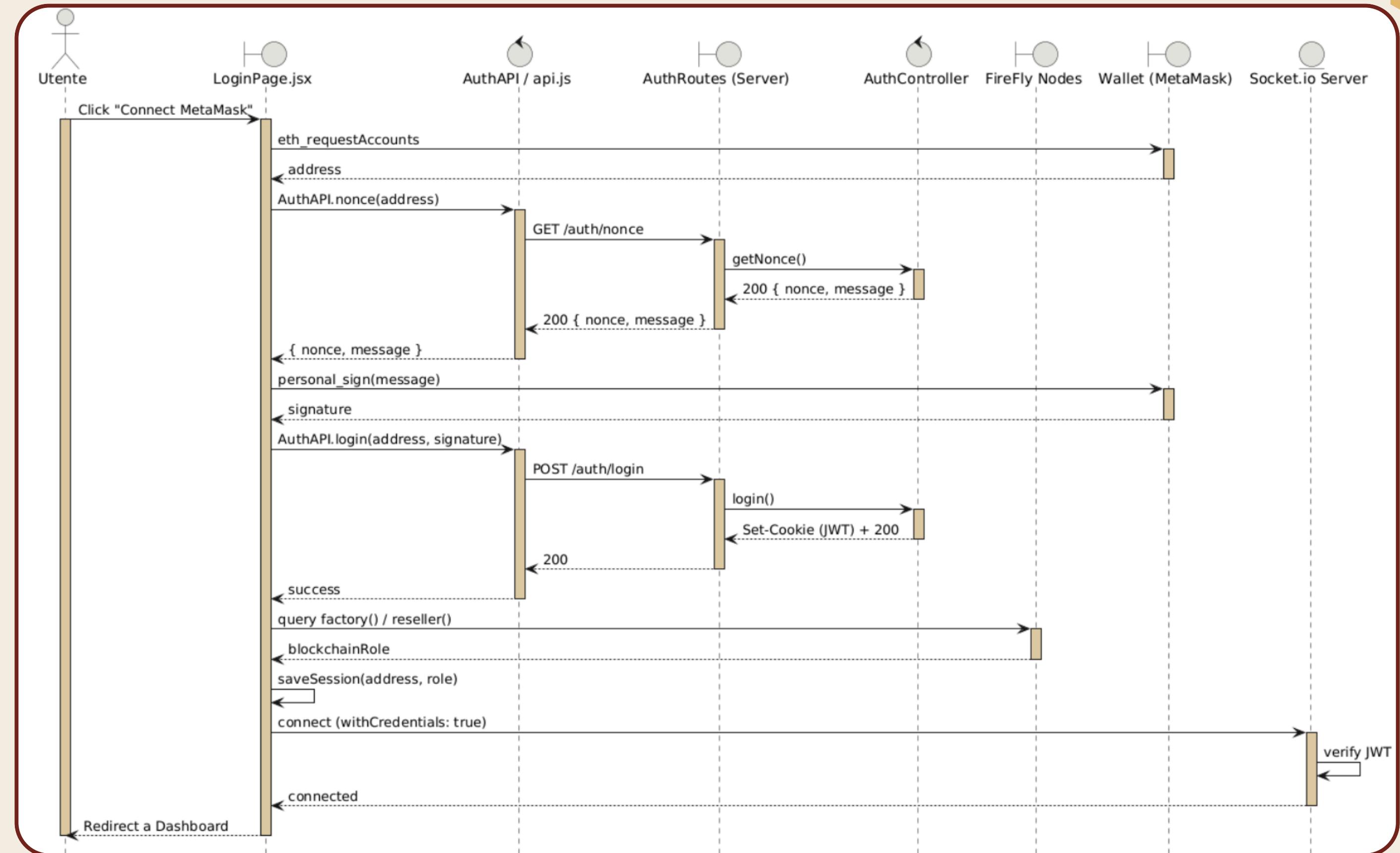
ID/Nome	UC_08: Aggiornamenti in tempo reale
Attore	Producer, Reseller, Consumer
Entry Condition	<ul style="list-style-type: none">• Il sistema è in esecuzione e in ascolto degli eventi.• L'utente è autenticato e sta visualizzando una schermata che mostra dati aggiornabili (market, profilo, ecc.).
Flusso	<ol style="list-style-type: none">1. L'utente compie un'operazione che può modificare lo stato (es. acquisto, vendita, certificazione).2. Il sistema rileva il cambiamento di stato.3. Il sistema notifica agli utenti connessi che è disponibile un aggiornamento.4. Il sistema aggiorna automaticamente i dati sul dispositivo dell'utente recuperando lo stato più recente.5. L'utente visualizza lo stato aggiornato.
Exit Condition (on success)	<ul style="list-style-type: none">• Lo stato visualizzato dall'Utente è aggiornato.
Exit Condition (on failure)	Il sistema ignora l'evento e non invia notifiche. L'utente non visualizza gli aggiornamenti in tempo reale.
Flusso alternativo	<ul style="list-style-type: none">• Evento non valido o incompleto: il sistema ignora l'evento e non invia notifiche.• Sorgente non autorizzata: il sistema rifiuta l'evento e non invia notifiche.• Errore durante l'elaborazione: il sistema non notifica e registra l'errore per diagnosi.• Utente non connesso: l'utente non riceve l'aggiornamento in tempo reale, ma visualizzerà lo stato aggiornato alla successiva consultazione.

UC_09: Gestione Emergenza e Sospensione operazioni

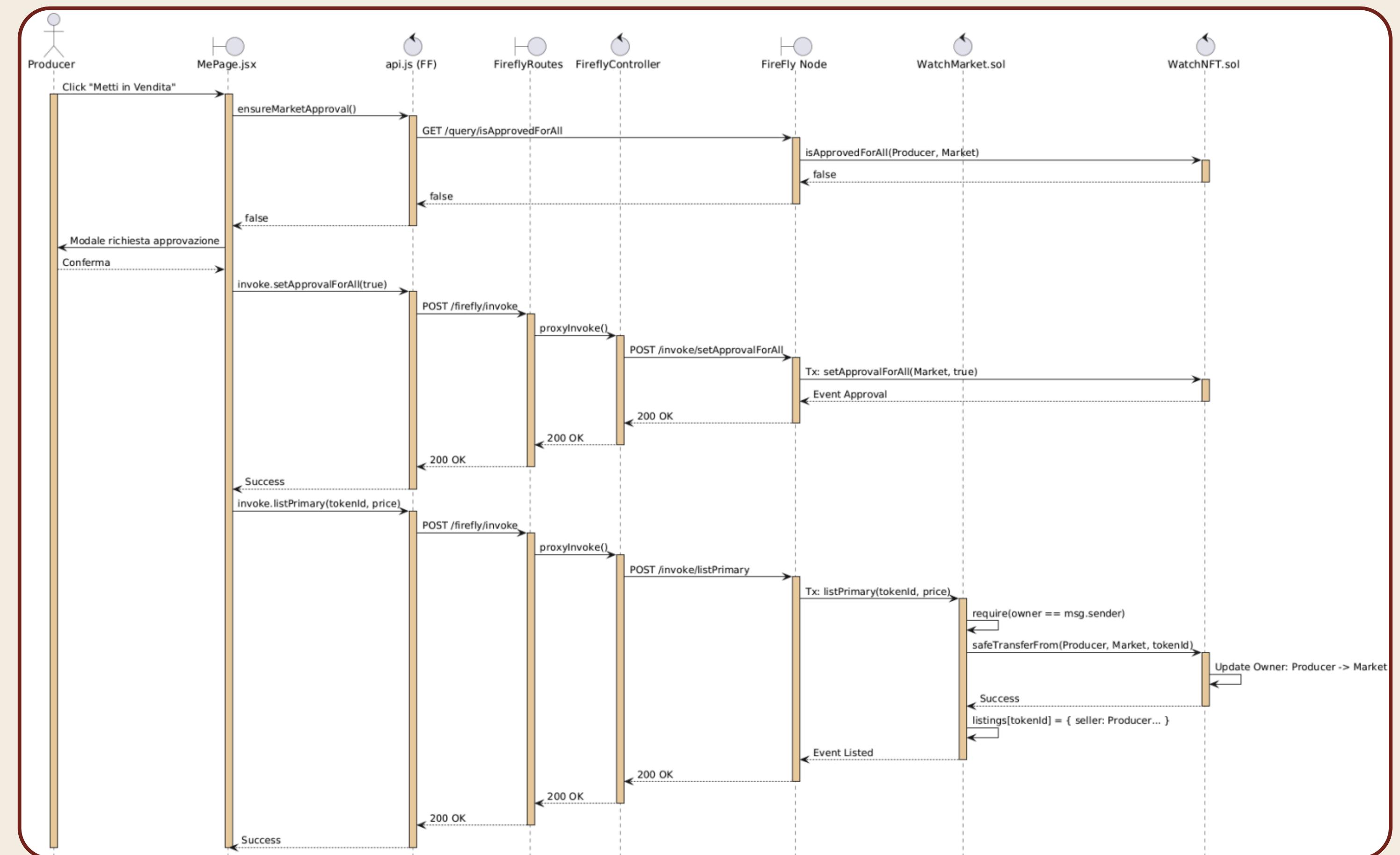
ID/Nome	UC_09: Sospensione operazioni
Attore	Producer
Entry Condition	<ul style="list-style-type: none"> • Il Producer è autenticato.
Flusso	<ol style="list-style-type: none"> 1. Il Producer accede alla funzionalità di gestione emergenza. 2. Il Producer seleziona l'azione desiderata. 3. Il sistema richiede conferma dell'operazione. 4. Il sistema aggiorna lo stato operativo del sistema, abilitando o bloccando le operazioni interessate. 5. Il sistema conferma l'avvenuto aggiornamento.
Exit Condition (on success)	<ul style="list-style-type: none"> • Se la sospensione è attivata, le operazioni di creazione e/o compravendita risultano temporaneamente non disponibili. • Se la sospensione è disattivata, le operazioni tornano disponibili secondo le regole di autorizzazione previste.
Exit Condition (on failure)	Lo stato del sistema rimane invariato e l'operazione non viene applicata.
Flusso alternativo	//

Sequence Diagrams

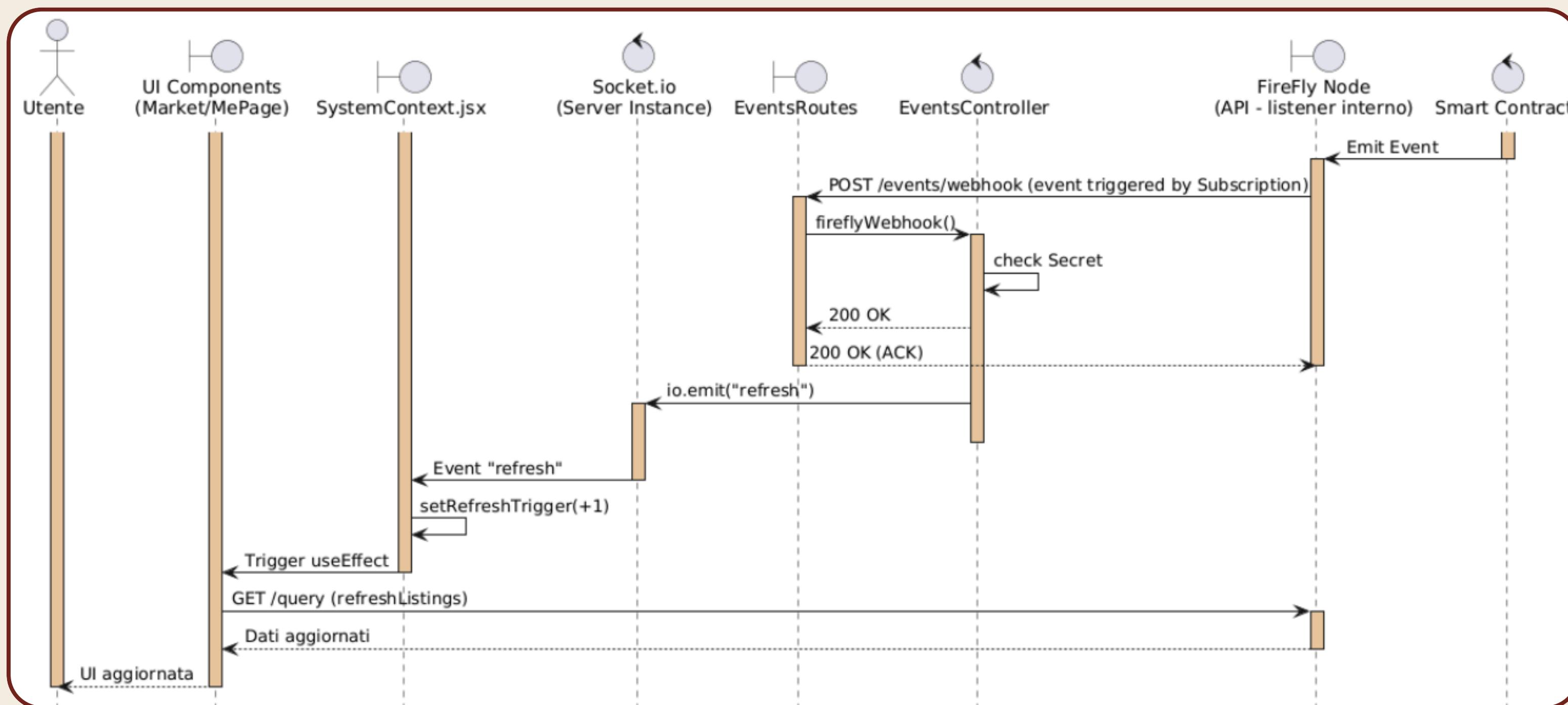
SD_01: Autenticazione utente



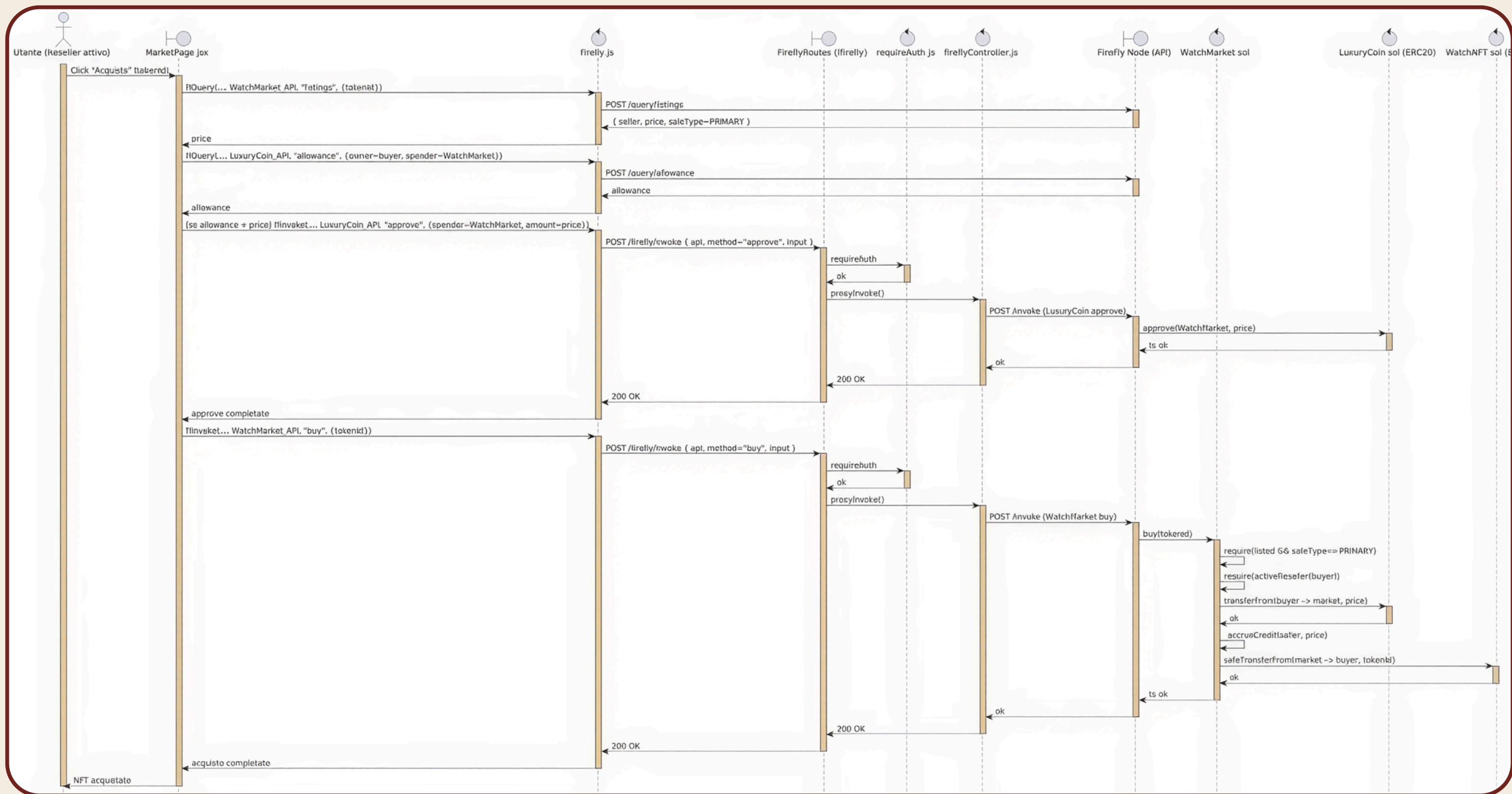
SD_02: Listing primario



SD_03: Aggiornamento in tempo reale tramite eventi



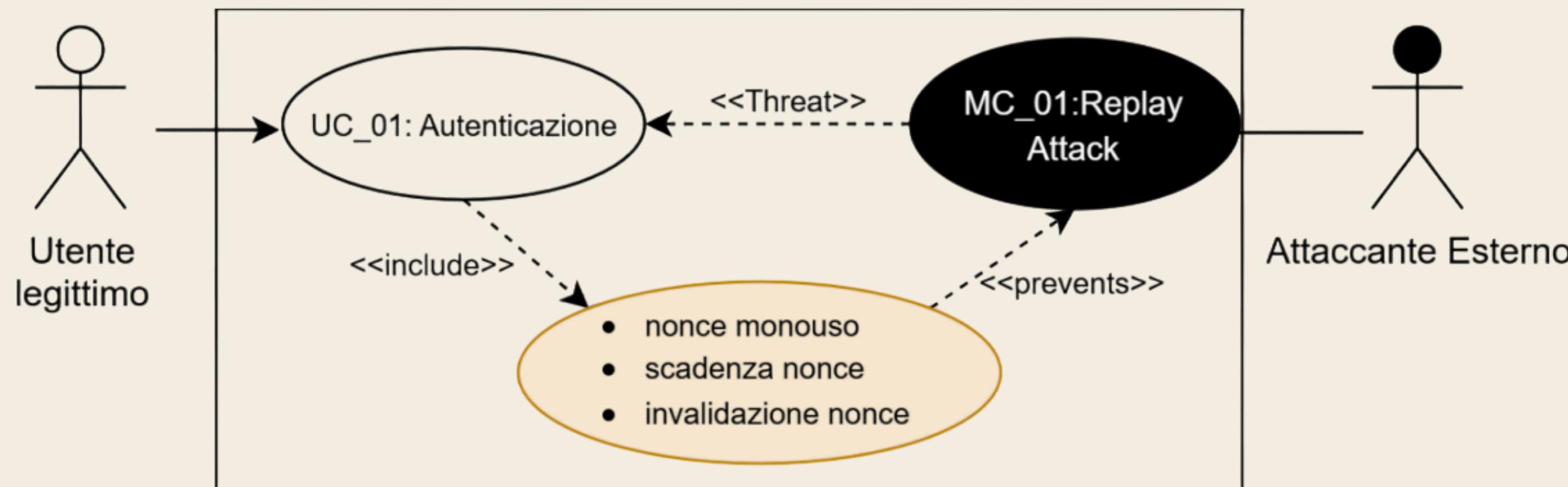
SD_04: Acquisto primario



Misuse Case

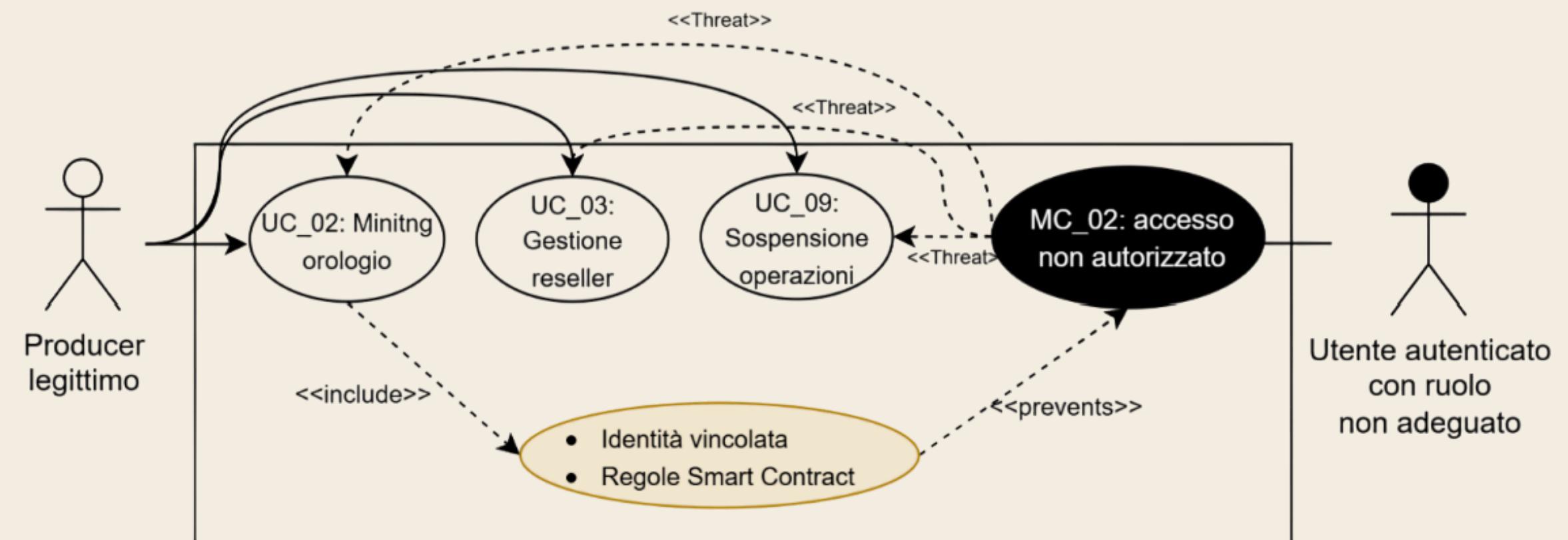
MC_01: Riutilizzo della firma di autenticazione (Replay Attack)

ID/Nome	MC_01: Replay Attack
Scenario	Un attaccante tenta di autenticarsi riutilizzando una firma crittografica precedentemente intercettata.
Misactor	Attaccante esterno
Obiettivo	Accedere al sistema impersonando un utente legittimo.
Funzionalità minacciata	UC_01: Autenticazione utente
Contromisure	<ul style="list-style-type: none">Utilizzo di nonce monouso associati all'indirizzo walletScadenza temporale del nonceInvalidazione del nonce dopo il primo utilizzo



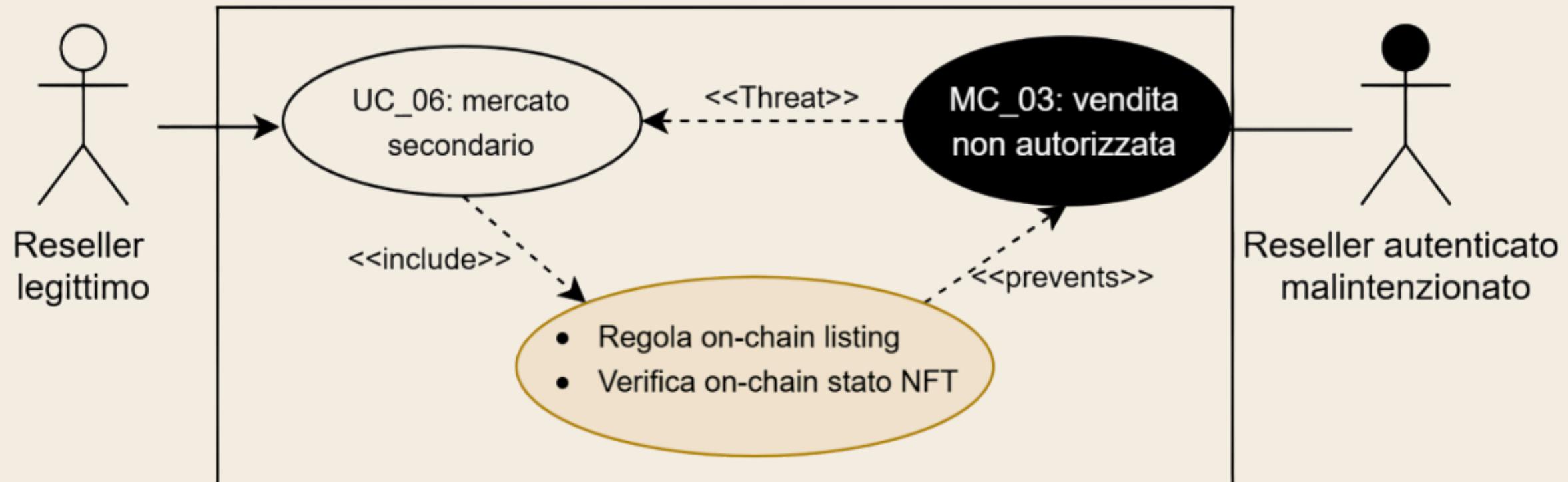
MC_02: Accesso a funzionalità non autorizzate per ruolo

ID/Nome	MC_02: Accesso non autorizzato
Scenario	Un utente autenticato tenta di accedere direttamente a operazioni riservate ad altri ruoli, ad esempio invocando manualmente endpoint o funzioni di smart contract non autorizzate.
Misactor	Utente autenticato con ruolo non adeguato
Obiettivo	Ottenere accesso a operazioni non consentite dal modello di dominio.
Funzionalità minacciate	<ul style="list-style-type: none"> • UC_02: Minting Orologio • UC_03: Gestione Reseller • UC_09: Sospensione operazioni
Contromisure	<ul style="list-style-type: none"> • Identità vincolata: firma sempre associata all'indirizzo autenticato. • Autorizzazione finale on-chain: operazioni non ammesse falliscono.



MC_03: Vendita di orologi non certificati

ID/Nome	MC_03: Vendita non autorizzata
Scenario	Un utente tenta di listare sul mercato secondario un orologio non certificato, aggirando i vincoli del sistema.
Misactor	Reseller autenticato malintenzionato
Obiettivo	Vendere un asset non conforme alle regole del sistema.
Funzionalità minacciata	UC_06: Mercato secondario
Contromisure	<ul style="list-style-type: none"> • Verifica on-chain dello stato di certificazione dell'NFT • Regola on-chain: listing consentito solo per NFT certificati.



Limitazioni

Firma Custodial

L'utente firma solo il login, le transazioni Blockchain sono firmate dai nodi FireFly.

Trade-off: Si sacrifica la decentralizzazione totale per abbattere i costi (Gas Fee) e semplificare l'esperienza.

Infrastruttura Statica

Attualmente la rete FireFly è preconfigurata per supportare solo tre attori (1 Nodo = 1 Attore).

L'architettura non scala dinamicamente.

Sviluppi futuri

- Superare la rigidità dei nodi statici.
- Permettere l'ingresso di nuovi partecipanti senza dover riconfigurare l'infrastruttura dei nodi FireFly.
- Evoluzione UI con pagine dedicate agli storici acquisti/vendite



Grazie per
l'attenzione!



<https://github.com/MarioLezzi92/Watchchain>