# Università degli studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata


## Project Work Blockchains



| **Studente** | **Email** | **Matricola** | **WP** |
|---|---|---|---|
| Luca Donnarumma | l.donnarumma16@studenti.unisa.it | 0622702172 | WP2 |
| Giuseppe Biscardi | g.biscardi@studenti.unisa.it | 0622702315 | WP3 |
| Elena Falcone | e.falcone9@studenti.unisa.it | 0622702314 | WP1 |

Prof. Carlo Mazzocca


a.a. 2024/25

# WP1 - MODELLO

## 1.1. DESCRIZIONE DEL PROBLEMA

Negli ultimi anni, la maggior parte dei consumatori ha sviluppato l'abitudine di leggere le recensioni online prima di prendere qualsiasi decisione d'acquisto. Questo comportamento è diventato un passo fondamentale nel processo decisionale, tanto che le recensioni sono spesso considerate una fonte di informazioni più autorevole rispetto persino alla pubblicità tradizionale. I clienti monitorano costantemente i feedback pubblicati sulle piattaforme digitali, sia che si tratti dell'acquisto di un prodotto o di un servizio, sia che debbano scegliere un ristorante o un hotel. Di conseguenza, le aziende sono sempre più consapevoli dell'importanza di mantenere una buona reputazione online, impegnandosi attivamente a monitorare e gestire i propri profili su vari portali di recensioni, al fine di tutelare la loro immagine.

Tuttavia, se da un lato le recensioni online rappresentano un riferimento essenziale per i consumatori, dall'altro la loro affidabilità viene sempre più messa in discussione. Uno studio, infatti, rivela che il 75% dei consumatori è preoccupato per la diffusione di recensioni false e che il 39% degli utenti statunitensi si fida meno dei sistemi di recensione rispetto a cinque anni fa (Clark, 2025), segnalando un progressivo indebolimento della credibilità di questi strumenti.

Le cause principali di questa tendenza sono:

- la facilità con cui si possono creare account falsi, utilizzati per lasciare recensioni fittizie, sia positive che negative, dando luogo a situazioni paradossali, come la presenza di ristoranti inesistenti nelle classifiche di ristorazione più alte. (*The Shed at Dulwich*, n.d.)

- la mancanza di trasparenza nei criteri di ordinamento e visibilità delle recensioni, con alcune piattaforme che penalizzano o oscurano recensioni negative senza una logica chiara (Handy, 2012);

- la scarsa incentivazione degli utenti reali a lasciare recensioni autentiche, che spesso non sono premiate né considerate rilevanti;

- la presenza di dinamiche poco virtuose tra piattaforme e operatori commerciali, dove i gestori dei servizi o prodotti possono "acquistare" recensioni o esercitare pressioni per far rimuovere quelle negative.

Nel contesto della **ristorazione**, queste problematiche risultano particolarmente evidenti e impattanti. Le recensioni online influenzano in modo determinante la scelta di un ristorante da parte dei clienti, soprattutto nei grandi centri urbani o nelle località turistiche, dove l'offerta è ampia e spesso sconosciuta al consumatore.
La possibilità di lasciare recensioni anonime o attraverso account non verificati rende banale, per esempio, colpire intenzionalmente un concorrente con recensioni negative false, oppure innalzare artificialmente il proprio punteggio con commenti positivi acquistati o scritti da amici, familiari o bot. In entrambi i casi, si genera una distorsione dell'esperienza reale, minando la fiducia degli utenti nel sistema.
A peggiorare la situazione contribuisce la **scarsa trasparenza** nei criteri con cui le piattaforme decidono quali recensioni mettere in evidenza. Non è raro che giudizi brevi e superficiali — magari

lasciati da clienti occasionali — vengano messi in evidenza, mentre recensioni più articolate e autentiche, scritte da clienti abituali che conoscono davvero la cucina e il servizio del ristorante, rimangano in secondo piano. Questo meccanismo penalizza l'impegno costante dei ristoratori e contribuisce a restituire un'immagine parziale o distorta della loro attività.
Inoltre, alcuni ristoratori lamentano di subire pressioni da parte di piattaforme che offrono "servizi premium" in cambio di maggiore visibilità o della possibilità di intervenire su recensioni negative.

In questo scenario, molti clienti abituali scelgono di non recensire, scoraggiati da un sistema percepito come poco meritocratico. Allo stesso tempo, i ristoratori onesti faticano a emergere se non accettano di aderire a dinamiche opache o potenzialmente scorrette. Il risultato è un sistema in cui l'utente meno esperto tende a considerare le valutazioni online come un criterio assoluto, senza riuscire a cogliere le distorsioni che lo influenzano. Così, nel migliore dei casi, si finisce per privilegiare sempre gli stessi ristoranti, spesso più bravi nel marketing che in cucina; nel peggiore, si rischia un'esperienza deludente che penalizza sia il cliente che l'intero settore.

### 1.1.1 Obiettivi del progetto

Per questo motivo, si rende necessario **progettare un sistema decentralizzato per la gestione di recensioni**, in grado di:

- garantire la **trasparenza e tracciabilità** delle recensioni senza compromettere la privacy degli utenti;
- **incentivare la partecipazione**, sia da parte dei clienti che dei ristoratori;
- offrire una **rappresentazione autentica e meritocratica** della qualità dei servizi ristorativi.

Un sistema basato su tecnologie decentralizzate, come la **blockchain**, potrebbe fornire una soluzione innovativa, riducendo l'intermediazione delle piattaforme centralizzate e restituendo fiducia e trasparenza a un meccanismo oggi spesso percepito come inaffidabile.

## 1.2. ATTORI ONESTI

In un sistema decentralizzato di gestione delle recensioni, i vari attori interagiscono per garantire che il processo di valutazione e feedback avvenga in modo sicuro, trasparente e affidabile. Ogni attore ha un ruolo specifico che contribuisce al funzionamento dell'intero sistema. La decentralizzazione consente di ridurre il controllo centralizzato, aumentando così la fiducia e la libertà degli utenti. In questo capitolo esploreremo i principali attori coinvolti nel sistema, analizzando i loro compiti e come interagiscono tra loro.

### 1.2.1 Panoramica degli attori onesti coinvolti

Il sistema decentralizzato di gestione delle recensioni si basa sull'interazione di quattro attori principali, ciascuno con un ruolo fondamentale:

- **Reviewers (Customers):** individui che interagiscono con il sistema per lasciare recensioni dei prodotti acquistati.
- **Likers (Customers):** individui che interagiscono con il sistema per supportare le recensioni dei prodotti acquistati.

- **Viewers (Customers):** utenti passivi che si limitano a leggere le recensioni col fine di scegliere un seller senza interagire attivamente con il sistema.
- **Sellers:** fornitore di servizi che offrono prodotti o risorse all'interno del sistema.
- **Review Entity:** componente che conserva e gestisce le recensioni lasciate dai clienti.
- **Certified Authority:** ente che garantisce l'identità degli attori

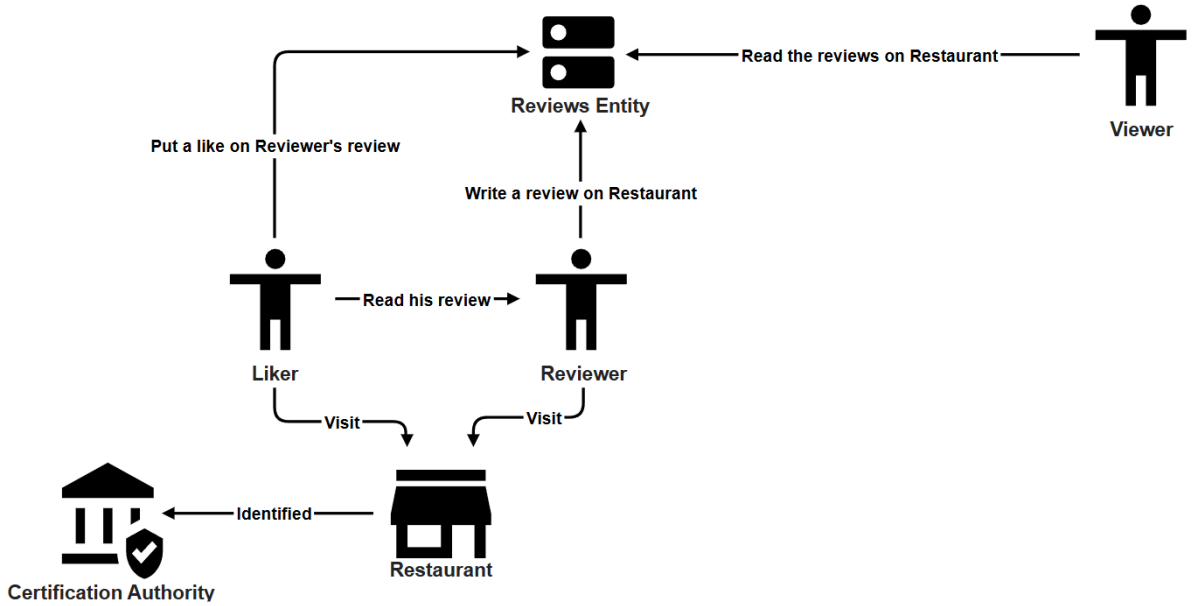



*Figure 1. Schema riassuntivo delle interazioni tra gli attori onesti*

## Customers

I clienti sono gli utenti finali che interagiscono con la piattaforma per acquistare prodotti o servizi, lasciare recensioni e valutare quelle altrui. Sono il motore del sistema, poiché forniscono i contenuti (le recensioni) su cui si basa il valore informativo della piattaforma. Si possono distinguere tre ruoli complementari:

- **Reviewers:** lasciano recensioni sui prodotti/servizi acquistati.
- **Likers:** supportano le recensioni pubblicate da altri utenti.
- **Viewers:** leggono le recensioni presenti sul sistema.

**Obiettivi:**

- Condividere la propria esperienza per aiutare altri utenti.
- Supportano le recensioni altrui per contribuire alla qualità del sistema.
- Accedere a servizi e prodotti affidabili attraverso una piattaforma trasparente.

## Sellers

I sellers sono le entità (i ristoranti) che offrono prodotti o risorse all'interno del sistema. Sono i destinatari diretti delle recensioni, da cui dipende la loro reputazione sulla piattaforma.

**Obiettivi:**

- Aumentare il proprio profitto attraverso una reputazione positiva e meritata.

- Costruire un’immagine trasparente e affidabile del proprio brand.
- Comprendere il feedback dei clienti per ottimizzare prodotti e servizi.
- Fidelizzare la clientela tramite una comunicazione aperta e responsabile.

### Review entity

È il componente tecnico che gestisce, conserva e rende accessibili le recensioni lasciate dai clienti. Opera in maniera trasparente e adotta misure di sicurezza per prevenire manipolazioni o falsificazioni.

**Obiettivi:**

- Offrire un servizio di recensioni affidabile e accessibile.
- Favorire la partecipazione degli attori al sistema.

### Certified Authority (Autorità Certificatrice)

È l’ente responsabile della verifica delle identità di clienti e venditori. In un contesto decentralizzato, questa funzione è cruciale per prevenire frodi e abusi, garantendo l'autenticità degli attori e la sicurezza delle interazioni.

**Obiettivi:**

- Garantire l’autenticità delle identità nel sistema.

| Attore | Sinonimi | Ruolo nel sistema | Obiettivo principale |
|---|---|---|---|
| Reviewer | Utente, cliente, recensore, customer | Acquista e lascia recensioni sui prodotti o servizi. | Condividere esperienze utili con altri utenti. |
| Liker | Utente, cliente, customer | Acquista e valuta l’utilità delle recensioni scritte da altri utenti. | Supportare le recensioni più rilevanti per migliorare la qualità complessiva del sistema. |
| Seller | Ristorante, ristoratore | Offre prodotti/servizi e ricevono recensioni. | Migliorare reputazione e vendite attraverso recensioni positive. |
| Review Entity | - | Gestisce e conserva le recensioni in modo sicuro. | Garantire integrità, accessibilità e affidabilità delle recensioni. |
| Certified Authority | Ente certificatore, certification Entity | Verifica l’identità degli utenti nel sistema. | Assicurare autenticità e fiducia nelle interazioni. |

*Table 1. Tabella riassuntiva degli attori onesti*

## 1.3. THREAT MODEL

In un sistema di recensioni per ristoranti, è fondamentale considerare le minacce che potrebbero compromettere l'autenticità e l'affidabilità dei feedback pubblicati. Pertanto, in questa sezione ci

concentreremo sull'identificazione dei principali attaccanti del sistema, definendo in dettaglio i loro obiettivi e le risorse a loro disposizione.

### Terminologia

In questo capitolo, oltre a descrivere gli avversari, presenteremo anche le loro capacità e risorse. Al fine di evitare ripetizioni e garantire chiarezza, è necessario stabilire una terminologia precisa. Per evitare ripetizioni superflue nella sezione successiva, è utile definire con precisione le principali categorie di risorse e competenze che un attaccante può possedere:

- **Capacità Tecniche:** Rappresentano le conoscenze e le abilità specialistiche di un attaccante, comprendenti competenze in programmazione, l'interazione con sistemi complessi (ad esempio, blockchain, smart contract).

- **Capacità Computazionali:** Indicano le risorse hardware e software a disposizione di un attaccante per eseguire attacchi, che possono variare da dispositivi singoli a reti organizzate di risorse (come cloud, botnet, o intelligenza artificiale), capaci di lanciare attacchi su vasta scala e ad alta intensità.

- **Capacità Economiche:** Si riferiscono alla disponibilità di risorse finanziarie che un attaccante può impiegare per acquistare strumenti, identità, servizi o per influenzare il sistema attraverso transazioni costose, supportando attacchi di tipo economico come spam, attacchi a pagamento o operazioni di manipolazione dei mercati.

- **Capacità sociali:** le capacità sociali si riferiscono all'abilità di un attaccante di sfruttare la psicologia umana e interagire con gli individui per manipolare comportamenti, raccogliere informazioni o ottenere accesso a risorse. Queste competenze sono particolarmente rilevanti per attacchi come il phishing, il social engineering e la manipolazione emotiva, e possono coinvolgere l'inganno, l'intimidazione, la persuasione o la creazione di fiducia falsa.

- **Accesso al Sistema:** Descrive il livello di accesso che un attaccante ha a un sistema o rete. Questo può variare da un accesso limitato (come quello di un utente normale) a privilegi elevati (ad esempio, amministratori, operatori di oracolo), fino al completo controllo di nodi e comunicazioni off-chain.

Al fine di mantenere concisa l’analisi, per ciascun attaccante verranno discusse solo le risorse considerate più rilevanti ai fini della minaccia specifica. Si assumerà quindi che le capacità dell’avversario negli altri ambiti (tecniche, computazionali, economiche, ecc.), laddove non sia specificato altrimenti, siano irrilevanti.

## 1.3.1. Attaccanti

Di seguito vengono descritti i principali attori malevoli identificati nel threat model del sistema:

**Fake Reviewer:** utente che scrive una recensione senza aver mai acquistato o usufruito del servizio del seller recensito.

- **Scopo:** agisce per danneggiare o favorire indebitamente un’attività, anche in minima parte.

- **Risorse:** le capacità sociali rivestono un ruolo cruciale, poiché un attaccante esperto, anche con accesso legittimo al sistema, può rendere la recensione credibile, ingannando gli utenti e influenzando significativamente la percezione dell'attività recensita. Queste capacità sociali possono arrivare ad essere molto buone, ma comunque rimangono strettamente correlate al talento individuale della persona nell'ingannare gli altri.

**Sybil attacker:** attaccante che crea numerose identità falsificate per manipolare o influenzare il sistema.

- **Scopo:** compromettere l'integrità del processo di valutazione, distorcendo l'esito delle recensioni e influenzando la percezione generale di un prodotto o servizio.
- **Risorse:** capacità computazionali e tecniche avanzate, necessarie per creare un gran numero di identità falsificate e alterare significativamente il processo di valutazione, influenzando così i risultati complessivi.

**Phantom Seller:** ristoratore fittizio che si presenta come attività legittima pur non offrendo alcun servizio reale.

- **Scopo:** ottenere visibilità e guadagni tramite recensioni fraudolente e promozioni ingannevoli, spesso con l'intento di truffare i clienti.
- **Risorse:** le risorse computazionali ed economiche sono limitate, ma sufficienti, poiché la partecipazione al sistema richiede solo strumenti di base a costo accettabile. La visibilità ottenuta con recensioni false e promozioni ingannevoli può poi amplificare l'efficacia del raggiro. Le risorse sociali restano contenute, dipendendo dalla capacità dell'attaccante di costruire una reputazione online credibile.

**Review Repeater:** cliente che, dopo aver pubblicato una recensione valida, tenta di inviarne ulteriori per la stessa visita o spesa, sfruttando ripetutamente la medesima prova d'acquisto.

- **Scopo:** amplificare l'impatto della propria opinione, cercando di influenzare la reputazione dell'attività in modo sproporzionato rispetto all'effettiva esperienza.
- **Risorse:** dispone di capacità tecniche elevate, necessarie per eludere i controlli sulla duplicazione delle recensioni.

**Proxy Reviewer:** cliente che, dopo aver acquistato un servizio, per pigrizia o per motivi strategici, cede ad altri il diritto o la possibilità di lasciare una recensione, delegando così l'opinione a soggetti terzi.

- **Scopo:** alterare la reputazione dell'attività tramite recensioni non autentiche, che non riflettono l'esperienza diretta dell'acquirente.
- **Risorse:** risorse sociali moderate, in quanto l'efficacia dell'attacco dipende dalla disponibilità e dalla fiducia di terzi nel lasciare recensioni false; risorse tecniche limitate.

**Dishonest Reviewer:** cliente che ha realmente usufruito del servizio ma pubblica una recensione ingannevole o non corrispondente alla realtà, per motivi personali o esterni.

- **Scopo:** influenzare negativamente o positivamente la reputazione dell'attività, motivato da rancori personali, vantaggi esterni o altri fattori emotivi.
- **Risorse:** risorse tecniche, economiche, computazionali e sociali minime; l'attacco si basa principalmente su intenzionalità e motivazione personale, non su capacità particolari.

**Proof Broker:** ristoratore che condivide la propria prova d'acquisto con un altro esercente, che la distribuisce a clienti ignari per ottenere recensioni ingannevoli sul primo seller.

- **Scopo:** costruire una reputazione artificiale, alterando la percezione pubblica tramite recensioni non veritiere, basate su prove d'acquisto falsificate.
- **Risorse:** risorse sociali moderate (per trovare e coinvolgere terzi), risorse economiche contenute, ma sufficiente coordinazione per orchestrare lo scambio delle prove d'acquisto.

**Privacy Violator:** attaccante esterno che mira a raccogliere o esporre dati sensibili degli utenti, sfruttando vulnerabilità nel sistema per compromettere la loro privacy.

- **Scopo:** violare la privacy degli utenti raccogliendo dati sensibili tramite la piattaforma, sfruttando l'anonimato o estraendo informazioni private da recensioni pubbliche.
- **Risorse:** accesso limitato al sistema con capacità tecniche e computazionali sufficienti per tracciare il comportamento degli utenti e analizzare le loro recensioni.

**Reputation Manipulator:** seller che corrompe gli utenti per ottenere recensioni positive sulla propria attività e/o per aumentare il numero di recensioni positive.

- **Scopo:** influenzare il sistema di recensioni attraverso pratiche fraudolente con l'obiettivo di ottenere vantaggi competitivi ingiusti.
- **Risorse:** risorse economiche consistenti per corrompere un numero sufficiente di utenti tale da alterare il processo di valutazione e compromettere l'affidabilità delle recensioni.

**Review Alterer:** utente che modifica o manipola il contenuto di una recensione altrui.

- **Scopo:** alterare l'opinione pubblica riguardo un'attività o servizio, cambiando la valutazione o il contenuto di recensioni già pubblicate, con l'obiettivo di favorire o danneggiare un'attività.
- **Risorse:** capacità tecniche moderate, in quanto l'attaccante ha bisogno di accesso al sistema per modificare il testo o la valutazione di una recensione. Le risorse sociali e economiche non sono sempre necessarie, ma l'attaccante potrebbe utilizzare account falsi o sfruttare vulnerabilità nel sistema per compiere questo tipo di manipolazione.

**Replay Attacker:** utente che intercetta un'operazione fatta da un altro utente e cerca di utilizzarla per compromettere la credibilità del sistema.

- **Scopo:** Arrecare un danno agli altri utenti, ad esempio far spendere nuovamente degli ethereum all'utente vittima.
- **Risorse:** capacità tecniche tali da riuscire ad intercettare le operazioni sulla blockchain per ri-inviarle.

# 1.4. FUNZIONALITÀ

In questa sezione si definiscono le funzionalità fondamentali che il sistema deve implementare per garantire la sicurezza, l'affidabilità e la privacy degli utenti, in conformità alle minacce individuate nel modello di threat analysis.

## F1 - Operazione di Pagamento

Prima che la prova di acquisto possa essere rilasciata, è necessario che l'utente effettui un pagamento utilizzando la criptovaluta **Ethereum (ETH)** direttamente sulla blockchain. Questa modalità consente di associare in modo trasparente e verificabile l'effettiva transazione economica all'azione di acquisto, garantendo l'integrità e la tracciabilità del processo.

Il sistema prevede che il pagamento venga inviato a un indirizzo specifico controllato dallo smart contract. Solo una volta che il contratto rileva l'avvenuto trasferimento di fondi e ne verifica la validità on-chain, viene autorizzata la generazione e l'emissione della **prova di acquisto**.

Questo meccanismo assicura che ogni recensione, successivamente abilitata dalla prova di acquisto, sia supportata da un'effettiva transazione economica registrata pubblicamente. In tal modo, si previene l'abuso del sistema da parte di soggetti malevoli e si rafforza la fiducia nell'autenticità delle recensioni.

## F2 - Rilascio prova d'acquisto

**Descrizione:**
Il sistema deve essere in grado di dimostrare che una persona ha effettivamente **usufruito del servizio**, pur senza svelarne l'identità.

**Dettagli:**

1. Il sistema deve implementare un meccanismo che garantisca la **prova dell'effettivo utilizzo (prova d'acquisto)** del servizio da parte dell'utente.

2. La possibilità di pubblicare una recensione deve essere legata a un **evento verificato e univoco**, garantendo che ogni recensione si riferisca a una singola esperienza di servizio.

## F3 - Scrittura delle recensioni

**Descrizione:**
Il sistema deve permettere al cliente di **pubblicare una recensione** relativa all'esperienza presso il seller, garantendo integrità, trasparenza e pseudoanonimato del contenuto.

**Dettagli:**

1. Il sistema **non deve permettere ad un seller di recensire sé stesso** né manipolare il sistema per generare recensioni positive.

## F4 - Verifica dell'Autenticità della Prova d'Acquisto e Gestione del Trasferimento

**Descrizione:**
Il sistema deve garantire che, una volta che una prova d'acquisto è stata associata a una recensione, non deve poter essere riutilizzata né trasferita per un altro scopo. Questo impedisce l'uso della stessa prova d'acquisto da parte di più persone o per più recensioni.

**Dettagli:**

1. La prova d'acquisto deve essere **associata in modo univoco** al profilo dell'utente che ha effettuato l'acquisto, e solo tale utente può utilizzarla per pubblicare una recensione.

2. La prova d'acquisto **non deve essere trasferibile**. Non può essere ceduta o trasferita a un altro utente, neppure se la persona che ha effettuato l'acquisto lo desidera.

3. Ogni prova d'acquisto **deve essere associata al seller** da cui l'acquisto è stato effettuato. Una volta rilasciata, non deve poter essere utilizzata per recensire nessun altro seller, ma solo quello specifico dove l'acquisto è avvenuto.

## F5 - Modifica o Revoca delle Recensioni

**Descrizione:**
Gli utenti devono poter **modificare o revocare le proprie recensioni**, ma secondo regole ben definite per prevenire abusi e garantire la correttezza del sistema.

**Dettagli:**

1. La modifica non deve essere permessa **oltre le 24 ore dalla pubblicazione** della recensione.

## F6 - Sistema di valutazione delle recensioni

**Descrizione:**
Il sistema deve consentire agli utenti di esprimere un voto alle recensioni degli altri, al fine di promuovere la visibilità dei contenuti più rilevanti e utili per la comunità.

**Dettagli:**

1. Le recensioni devono poter ricevere **voti positivi** ("utile") da parte degli utenti.
2. Solo gli utenti che hanno realmente visitato il ristorante oggetto della recensione devono poter mettere un like alla stessa.
3. Gli utenti **non devono poter votare le proprie recensioni**, in modo da prevenire manipolazioni del sistema (ad esempio, favorire artificiosamente la visibilità di recensioni scritte da sé).

## F7 - Incentivazione per la partecipazione al sistema

**Descrizione:**
Il sistema deve premiare gli utenti che scrivono recensioni per incentivare la partecipazione attiva e promuovere un impatto positivo sulla reputazione del seller.

**Dettagli:**

1. Gli utenti devono poter guadagnare **incentivi digitali** come ricompensa per aver recensito un certo numero di sellers. Questi premi sono finalizzati a motivare una partecipazione attiva e costruttiva.

## F8 - Tracciabilità

**Descrizione:**
Le recensioni devono essere memorizzate in modo **immutabile e tracciabile**, garantendo la trasparenza e la protezione della privacy degli utenti.

**Dettagli:**

1. Le recensioni devono venire salvate in un sistema sicuro che garantisce l'**immutabilità** del loro contenuto. Una volta pubblicata, la recensione non deve poter essere modificata senza registrare una nuova versione, mantenendo l'integrità del contenuto originale.

## F9 – Gestione della Privacy e Accesso ai Dati

**Descrizione:**
Il sistema deve garantire la protezione dei dati personali degli utenti e dei ristoranti, assicurando che le azioni compiute dagli utenti (come scrivere una recensione) non siano tracciabili o collegabili tra loro o alla loro identità reale.

**Dettagli:**

1. Le recensioni sono pubbliche ma non ricollegabili direttamente all'identità reale dell'utente.

2. Il sistema deve impedire che un osservatore, anche interno, possa collegare tra loro recensioni o altre azioni riconducibili allo stesso utente. Ogni interazione deve apparire come isolata.

3. Non deve essere possibile per terzi (inclusi gli operatori del sistema) risalire all'identità dell'utente a partire dai dati pubblici o dai metadati delle recensioni.

# 1.5. PROPRIETÁ DA PRESERVARE

Il sistema deve garantire un insieme di proprietà fondamentali per proteggere l'integrità del processo di recensione, contrastare comportamenti malevoli e tutelare la privacy e la fiducia degli utenti. Di seguito si descrivono tali proprietà associate ai principali vettori di attacco identificati nel modello di minaccia e alle principali proprietà delineate in precedenza.

1. **Confidenzialità:** garantire la protezione delle informazioni personali e delle recensioni degli utenti, impedendo l'accesso non autorizzato.
2. **Integrità:** garantire che le recensioni non vengano alterate senza traccia, mantenendo la veridicità del contenuto originale.
3. **Autenticità**: garantire che solo attori autentici possano partecipare al sistema, evitando la manipolazione da parte di attori malevoli.

4. **AntiFakeAccount property:** il sistema deve limitare la creazione di account fake.
5. **Non ripudio**: ogni azione (come la pubblicazione o modifica di una recensione) deve essere tracciata, e l'utente non può negare di aver compiuto una determinata azione.
6. **Trasparenza:** ogni modifica o interazione deve essere tracciata in modo chiaro, mantenendo la visibilità sullo stato delle recensioni e delle azioni degli utenti.
7. **Privacy:** garantire lo pseudo-anonimato degli utenti, al fine di tutelare l'identità degli stessi e impedire che le recensioni o le interazioni nel sistema possano essere ricollegate a persone specifiche, anche in presenza di dati pubblici o pseudonimi.
8. **Buy proof property:** garantire che l'utente possa lasciare una recensione solo e soltanto se ha effettivamente usufruito del prodotto.
9. **Usage property:** garantire che ciascuna prova d'acquisto sia strettamente associata all'acquirente e sia valida esclusivamente presso il seller in cui è stata originata.

## Attaccanti e proprietà

In questa sottosezione verrà analizzato nel dettaglio il comportamento di ciascun attaccante, evidenziando le modalità attraverso cui potrebbe compromettere le proprietà fondamentali del sistema, quali integrità, autenticità, non ripudio, disponibilità e trasparenza. L'obiettivo è comprendere l'impatto specifico di ogni attacco sul funzionamento complessivo del sistema e sulle sue garanzie di sicurezza, al fine di individuare le potenziali vulnerabilità e delineare le contromisure più efficaci.

## Fake Reviewer

La presenza dell'avversario violerebbe:

- **Buy proof property:** in quanto, se riuscisse nel proprio intento, inserirebbe nel sistema una recensione pur non essendo in possesso di una prova d'acquisto valida per il venditore (seller) oggetto della recensione.

## Sybil attacker

La presenza dell'avversario violerebbe:

- **AntiFakeAccount property:** in quanto, se riuscisse nel proprio intento, potrebbe creare un numero indefinito o illimitato di account falsi, con l'obiettivo di autoattribuirsi una quantità arbitraria di recensioni positive, alterando artificialmente la reputazione della propria attività.

## Phantom Seller

La presenza dell'avversario violerebbe:

- **Autenticità:** in quanto, se riuscisse nel proprio intento, potrebbe creare una o più attività fittizie, sia con finalità fraudolente sia per scopi goliardici, compromettendo l'affidabilità del sistema.

## Review Repeater

La presenza dell'avversario violerebbe:

- **Buy proof property:** in quanto, se riuscisse nel proprio intento, potrebbe inserire nel sistema più di una recensione associata alla medesima prova d'acquisto, influenzando in modo artificiale la reputazione di una specifica attività, in senso positivo o negativo.

## Proxy Reviewer

La presenza dell'avversario violerebbe:

- **Buy proof property:** in quanto, se l'avversario riuscisse nel proprio intento, consentirebbe a un utente privo di una valida prova d'acquisto di pubblicare comunque una recensione, compromettendo l'integrità del sistema di valutazione.
- **Non ripudio:** in quanto, se riuscisse nel proprio intento, il soggetto che ha ceduto la prova d'acquisto potrebbe negare la paternità della recensione pubblicata da chi ne ha fatto uso, compromettendo la tracciabilità e la responsabilità dell'azione.

## Dishonest Reviewer

La presenza dell'avversario non comporterebbe la violazione di alcuna proprietà del sistema, in quanto l'inserimento di una recensione falsa — sebbene motivata da ragioni personali tra il venditore e il cliente — risulta comunque tecnicamente lecito. Il sistema, infatti, privilegia la partecipazione e l'apertura, lasciando spazio a soggettività e opinioni, anche quando non pienamente oggettive.

## Proof Broker

La presenza dell'avversario violerebbe:

- **Usage property:** in quanto, se l'avversario riuscisse nel proprio intento, effettuerebbe un trasferimento della sua prova d'acquisto, valida per la propria attività, a favore di un altro venditore, con l'obiettivo di ottenere recensioni positive per quest'ultimo.

## Privacy Violator

La presenza dell'avversario violerebbe:

- **Privacy:** in quanto, se l'avversario riuscisse nel proprio intento, riuscirebbe a identificare le identità dei reviewer o almeno a tracciarli in base al loro comportamento, ottenendo una sorta di profilazione degli stessi.

## Reputation Manipulator

La presenza dell'avversario non costituisce una vera e propria violazione delle proprietà del sistema, poiché la manipolazione da lui attuata è formalmente esterna e non produce effetti illegittimi direttamente sul sistema.

## Review Alterer

La presenza dell'avversario violerebbe:

- **Integrità**, in quanto, se riuscisse nel proprio intento, altererebbe a proprio vantaggio recensioni già esistenti, compromettendo l'autenticità delle informazioni.

- **Non ripudio**, in quanto la modifica delle recensioni verrebbe attribuita all'autore originario, il quale non potrebbe negarne la paternità, nonostante non ne sia il responsabile.
- **Trasparenza**, in quanto l'intervento dell'avversario renderebbe opaca la provenienza e l'autenticità delle recensioni, inducendo in errore altri utenti circa la reale esperienza dei clienti.

## Replay Attacker

La presenza dell'avversario violerebbe:

- **Integrità**, in quanto, se riuscisse nel proprio intento, potrebbe far sembrare che una recensione sia stata inviata più volte o modificata, senza che l'utente reale lo abbia fatto. Questo altera il significato e la validità delle recensioni, compromettendo l'integrità del contenuto.
- **Non ripudio**, in quanto, se riuscisse nel proprio intento, potrebbe far trovare ad un utente un'azione a lui associata che non ha effettivamente compiuto.

| Attaccante | Descrizione | Funzionalità vulnerabili | Proprietà da preservare |
|---|---|---|---|
| Fake Reviewer | Cliente che scrive una recensione senza aver mai acquistato o usufruito del servizio del seller recensito | F3 - Scrittura delle recensioni,<br>F4 - Verifica della prova d'acquisto | Buy proof property |
| Sybil Attacker | Seller che crea identità false per manipolare o influenzare un sistema. | F3 - Scrittura delle recensioni,<br>F4 - Verifica della prova d'acquisto | AntiFakeAccount |
| Phantom Seller | Seller fittizio che si presenta come attività legittima pur non offrendo alcun servizio reale. | F3 - Scrittura delle recensioni,<br>F4 - Verifica della prova d'acquisto | Autenticità |
| Review Repeater | Cliente che, dopo aver pubblicato una recensione valida, tenta di inviarne ulteriori per la stessa visita o spesa. | F3 - Scrittura delle recensioni,<br>F4 - Verifica della prova d'acquisto | Buy proof property |
| Proxy Reviewer | Cliente che, dopo aver acquistato un servizio, per pigrizia o per motivi strategici, cede ad altri il diritto o la possibilità di lasciare una recensione. | F3 - Scrittura delle recensioni,<br>F4 - Verifica della prova d'acquisto | Buy proof property, Non ripudio. |
| Dishonest Reviewer | Cliente che ha realmente usufruito del servizio ma | F3 - Scrittura delle recensioni | - |

| | | | |
|---|---|---|---|
| | pubblica una recensione ingannevole o non corrispondente alla realtà, per motivi personali o esterni. | | |
| Proof Broker | Seller che condivide la propria prova d'acquisto con un altro esercente, che la distribuisce a clienti ignari per reindirizzare recensioni ingannevoli sul primo seller. | F3 - Scrittura delle recensioni,<br>F4 - Verifica della prova d'acquisto | Usage property |
| Privacy Violator | Attaccante esterno che mira a raccogliere o esporre dati sensibili degli utenti, sfruttando vulnerabilità nel sistema per compromettere la loro privacy. | F9 - Gestione della privacy | Privacy, Confidenzialità |
| Reputation Manipulator | Seller che corrompe gli utenti per ottenere recensioni positive sulla propria attività e/o per aumentare il numero di recensioni positive. | F3 - Scrittura delle recensioni<br>F8 - Tracciabilità | - |
| Review Alterer | Utente che modifica o manipola il contenuto di una recensione altrui. | F5 - Modifica/Revoca recensioni<br>F1 - Operazione di Pagamento<br>F3 - Scrittura delle recensioni<br>F6 - Sistema di valutazione delle recensioni | Integrità, Non ripudio, Trasparenza |
| Replay Attacker | utente che intercetta un operazione fatta da un altro utente e cerca di utilizzarla per compromettere la credibilità del sistema. | F1 - Operazioni di pagamento<br>F2 - Rilascio prova d'acquisto<br>F5 - Modifica/Revoca recensioni | Integrità, Non ripudio |

*Table 2. Tabella riassuntiva delle proprietà da preservare e le funzionalità da implementare rispetto agli attaccanti*

# WP2 - SOLUZIONE

## 2.1 INTRODUZIONE

In questo capitolo verrà presentata **una possibile soluzione progettuale** al problema illustrato nel capitolo precedente, con l'obiettivo di offrire una visione chiara e completa del funzionamento del sistema proposto.
In particolare, dopo una prima panoramica, verrà descritta in modo più dettagliato e a basso livello l'architettura del sistema, evidenziando le componenti principali, le interazioni tra gli attori coinvolti e le azioni chiave che vengono eseguite.

## 2.2 STRUMENTI E TECNOLOGIE UTILIZZATE

Il sistema si basa su una blockchain **ibrida**, costruita su Ethereum e simulata tramite Ganache per scopi di sviluppo. La rete è concepita come **permissioned** per i ristoratori, che devono identificarsi e autenticarsi per poter interagire con il sistema in qualità di attori ufficiali. Questo meccanismo garantisce maggiore sicurezza, coerenza dei dati e prevenzione di comportamenti malevoli da parte di soggetti che potrebbero fingersi esercizi commerciali legittimi.

Al contrario, l'accesso per gli utenti finali (clienti) è **permissionless**: chiunque può scrivere recensioni in modo pseudoanonimo, senza dover fornire un'identità verificata. Questa apertura è fondamentale in un contesto come quello della ristorazione, dove è importante consentire la libera espressione delle opinioni. L'approccio ibrido consente dunque di bilanciare **controllo** e **apertura**, mantenendo i principali vantaggi della tecnologia blockchain — come trasparenza, tracciabilità e immutabilità — pur limitando l'accesso privilegiato alle funzioni critiche del sistema.

Il sistema integra inoltre diversi strumenti della tecnologia Web3, tra cui:

- **Smart Contract:** automatizzano l'esecuzione delle logiche di business, come la generazione di NFT o l'assegnazione dei bonus, rendendo le interazioni tra utenti e ristoranti trasparenti, sicure e programmabili.
- **Token e NFT:** i token vengono utilizzati per rappresentare rispettivamente prove di acquisto reali, mentre gli NFT (Non-Fungible Token) servono come voucher unici e non trasferibili, collegati a specifici utenti e ristoranti.
- **Wallet:** un wallet è un software o applicazione che consente all'utente di gestire i propri asset digitali e interagire con la blockchain. Attraverso il wallet, l'utente può firmare transazioni, autenticarsi con la propria chiave privata e partecipare alle attività on-chain come la riscossione dei bonus o la pubblicazione di recensioni.

## 2.3 ASSUNZIONI

- Si parte dal presupposto che tutti gli attori coinvolti nel sistema possiedano già un wallet contenente una coppia di chiavi crittografiche (pubblica e privata), necessari per l'accesso al sistema.
- Si assume che l'ente certificatore – ad esempio un ente governativo – sia considerato trusted. Per semplificare l'implementazione, questo attore verrà rappresentato mediante uno smart contract.

- Si presume che gli utenti abbiano familiarità con le criptovalute e siano in grado di effettuare pagamenti in Ethereum.

## 2.4 PANORAMICA AD ALTO LIVELLO DEL SISTEMA

Il sistema proposto è una piattaforma decentralizzata basata su tecnologia blockchain, progettata per gestire in modo trasparente, verificabile e sicuro il processo di scrittura, pubblicazione e valutazione delle recensioni nel settore della ristorazione. Gli attori principali coinvolti sono gli utenti e i ristoranti. I ristoranti, per poter partecipare attivamente, devono registrarsi attraverso un ente certificatore, che verifica la loro identità e autenticità (es. partita IVA) in modalità off-chain. Una volta verificati, i ristoranti vengono ufficialmente riconosciuti come affiliati e registrati in una struttura dati pubblica e immutabile sulla blockchain.

Gli utenti, per poter scrivere una recensione, devono prima effettuare un pagamento reale presso un ristorante affiliato. Questo pagamento, verificato da uno smart contract, consente il rilascio di un **token di recensione**, prova crittografica che attesta l'effettiva consumazione. Ogni token è strettamente legato all'identità dell'utente e al ristorante specifico, e viene registrato all'interno di una struttura dati che tiene traccia dei rapporti utente-ristorante.

La pubblicazione della recensione è possibile solo se l'utente possiede un token valido. Ogni recensione è memorizzata in strutture dati efficienti che ne permettono l'indicizzazione per ristorante, per utente e per combinazioni specifiche. Inoltre, il sistema permette agli utenti di esprimere un voto positivo (like) sulle recensioni, con meccanismi di verifica per evitare voti duplicati o non autorizzati.

Le recensioni possono essere modificate (entro un tempo definito) o revocate esclusivamente dall'autore originale, garantendo controllo e integrità del contenuto. Le azioni di modifica o revoca sono tracciate, e l'aggiornamento delle strutture dati avviene in modo coerente, compresa la rimozione di eventuali like in caso di modifica.

Infine, il sistema prevede un meccanismo di **incentivazione** per le recensioni ritenute utili o per la partecipazione attiva dell'utente. Al raggiungimento di una certa soglia di recensioni, l'utente può ricevere un **NFT sotto forma di voucher di sconto**, valido esclusivamente presso il ristorante recensito. Questo NFT può essere utilizzato per ottenere sconti sui futuri pagamenti, sempre tramite smart contract, che verifica la validità e l'associazione del voucher all'utente e al ristorante.

## 2.5 FUNZIONALITÁ

Questa sezione fornisce una descrizione più dettagliata e a basso livello delle funzionalità del sistema, ampliando e approfondendo quanto già illustrato nella panoramica generale.

### 2.5.1 - Identificazione attori

### Registrazione dei ristoranti

I ristoranti, per poter partecipare attivamente al sistema, devono identificarsi presso un **ente certificatore**. Per registrarsi nel sistema, i ristoranti devono compiere le seguenti operazioni preliminari:

- Accedere alla blockchain tramite **un wallet**;

A questo punto forniscono le seguenti informazioni all'ente certificatore:

- La propria partita IVA.

Successivamente l'ente certificatore **valida off-chain l'autenticità dei documenti ricevuti** (partita IVA) consultando fonti ufficiali o database pubblici.
Se tutte le condizioni sono soddisfatte, il sistema procede con le seguenti operazioni:

- Registra l'address **verificato** all'interno di una struttura dedicata;
- Rende il registro **consultabile in modo pubblico** e **immutabile** per qualsiasi verifica futura;
- Riconosce ufficialmente il **ristorante come affiliato**.

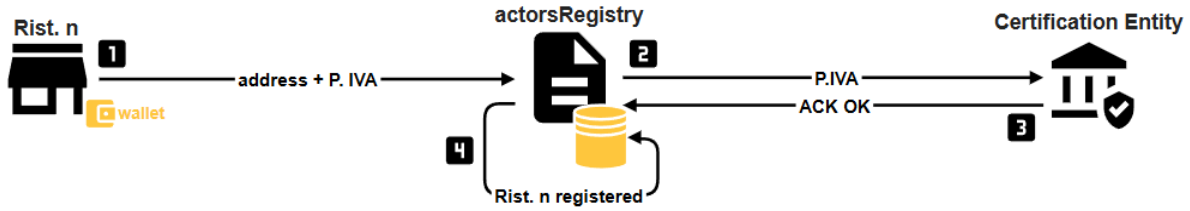



*Figure 2. Schema riassuntivo della registrazione dei ristoranti*

## Registrazione dei customers

I clienti, per registrarsi nel sistema, devono compiere le seguenti operazioni preliminari:

- Accedere alla blockchain tramite **un wallet**;

Una volta che ne hanno uno a disposizione possono attivamente partecipare al sistema.

### Nota sulla verifica degli address

Da questo punto in poi, ogni volta che si farà riferimento alla verifica di un address, si intenderà implicitamente l'utilizzo di una funzione dello smart contract incaricato di gestire l'autenticazione. Essa sarà incaricata di controllare che l'address in questione sia registrato all'interno della struttura dati che raccoglie i ristoranti affiliati.

Allo stesso modo, ogni volta che si parlerà della verifica del possesso di un address da parte di un soggetto, si intenderà che tale smart contract effettua una verifica crittografica in cui il soggetto dimostra di possedere la chiave privata corrispondente alla chiave pubblica associata al wallet.

## 2.5.2 - Pagamento del servizio

Per ottenere un token che consenta la scrittura di una recensione, l'utente deve prima effettuare un **pagamento presso il ristorante** che intende recensire. L'utente fornisce allo smart contract le seguenti informazioni:

- L'**ammontare** del pagamento in Ethereum;
- L'address del **mittente del pagamento**;
- L'address del **destinatario del pagamento**;
- L'**eventuale sconto**, sotto forma di NFT.

Lo smart contract esegue le seguenti operazioni:

- **Verifica del possesso dell'address dell'utente**.
- **Verifica l'identità del ristorante**, controllando che il suo address corrisponda a uno dei ristoranti affiliati e attivi nella rete;
- **Verifica l'NFT**, assicurandosi che:
  - sia valido (non scaduto né già utilizzato);
  - sia correttamente associato all'utente, tramite la prova crittografica del possesso dell'address legato all'NFT;
  - sia riferito al ristorante presso cui si sta effettuando il pagamento;
- **Verifica il saldo**, controllando che l'utente disponga di fondi sufficienti per completare il pagamento verso il ristorante.

Se tutte le operazioni sono andate a buon fine, il sistema procede con il **rilascio del token**.

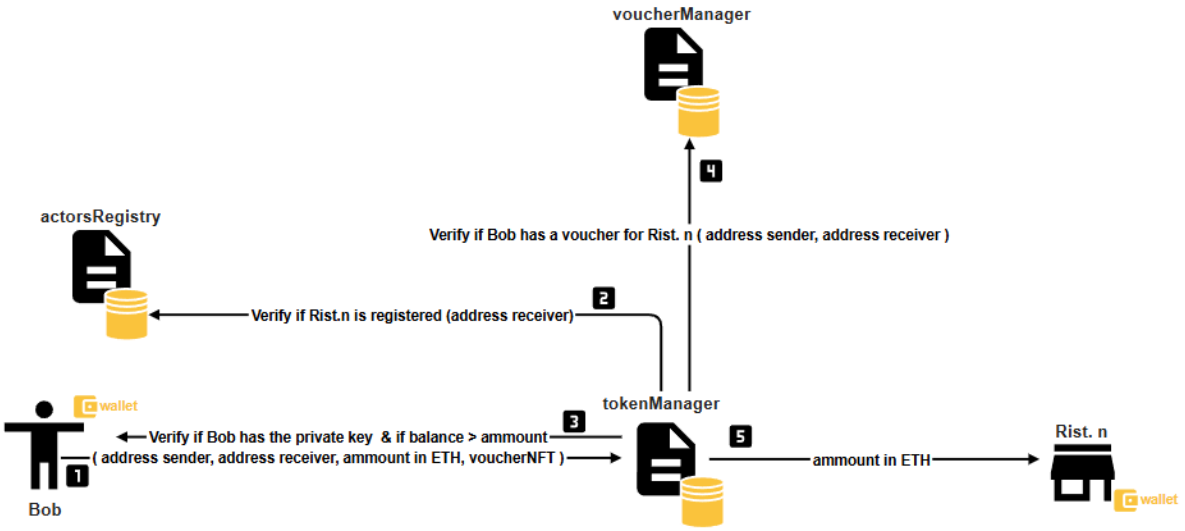



*Figure 3. Schema riassuntivo del pagamento del prodotto*

## 2.5.3 - Rilascio, verifica e gestione della prova d'acquisto

Dopo aver effettuato il pagamento, l'utente ha diritto a **ricevere un token che attesti l'avvenuta consumazione**, abilitandolo a scrivere una recensione. L'intero processo è gestito da uno smart contract. I token sono gestiti all'interno dello smart contract tramite una **struttura (mapping) principale** che associa ciascun token a una specifica coppia di address: utente e ristorante. Per ogni coppia, il sistema mantiene un contatore che registra il numero di token rilasciati a quell'utente per quel ristorante.

Lo smart contract esegue le seguenti verifiche riportate precedentemente. Se tutte le verifiche risultano superate, il sistema procede con l'operazione:

- **Genera un token di recensione**, associando l'address dell'utente a quello del ristorante. In particolare, per ogni coppia di addresses utente-ristorante, viene mantenuto un contatore che tiene traccia del numero di token rilasciati per quell'utente in relazione a quel ristorante specifico.

Questo meccanismo garantisce che ogni token sia emesso solo a fronte di un'esperienza di consumo reale e che sia **strettamente legato all'identità dell'utente**.

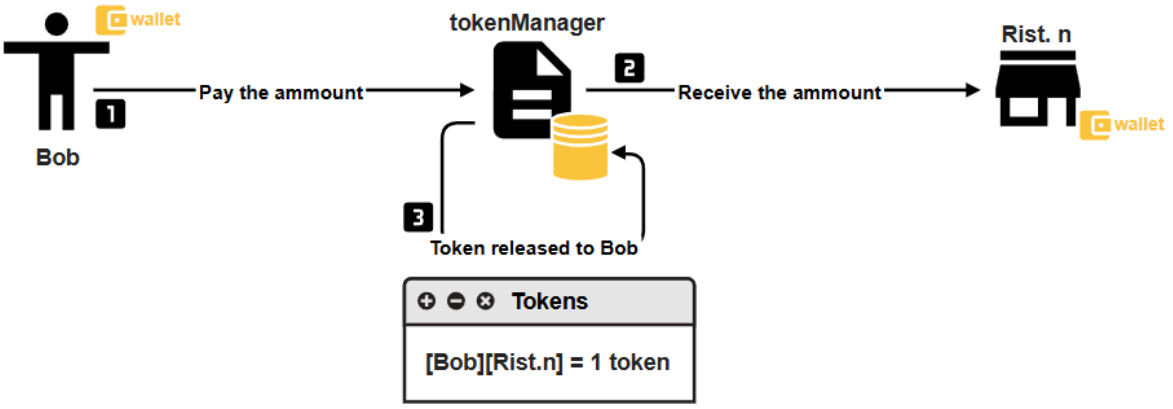



*Figure 4. Schema riassuntivo del rilascio, verifica e gestione della prova d'acquisto*

## 2.5.4 - Scrittura delle recensioni

Per poter scrivere una recensione, l'utente deve essere in possesso di un **token di recensione**, emesso a seguito di un **acquisto valido**. L'intero processo è gestito da uno smart contract.

### Struttura di una recensione

Ogni recensione è composta da:

- L' **address dell'utente** (autore della recensione);
- L' **address del ristorante** recensito;
- Il **contenuto testuale** della recensione;
- Il **numero di like** associati;
- Il **timestamp** in cui è stata pubblicata

Le recensioni vengono organizzate in diverse strutture dati (mapping), per permettere una gestione efficiente e facilmente consultabile:

- **Struttura primaria**: memorizza tutte le recensioni inserite nel sistema. Ogni recensione viene archiviata e identificata con un identificativo univoco;
- **Struttura secondaria per ristorante**: raggruppa le recensioni per ciascun ristorante, permettendo di consultare tutte le recensioni relative a un determinato ristorante;
- **Struttura secondaria per utente e ristorante**: mappa tutte le recensioni scritte da un singolo utente verso uno specifico ristorante, per facilitare la consultazione delle recensioni lasciate da un utente;
- **Struttura secondaria per i likers associati ad una recensione**: tiene traccia di quali utenti hanno già votato ogni recensione.

### Processo di pubblicazione

Nel momento in cui si desidera lasciare una recensione, l'utente fornisce allo smart contract dedicato le seguenti informazioni:

- Il **proprio address**;
- L'address **del ristorante** che intende recensire;
- Il **contenuto testuale** della recensione;

Lo smart contract preposto alla gestione delle recensioni esegue una serie di verifiche prima di accettare la pubblicazione:

- **Verifica del possesso dell'address dell'utente**.
- **Verifica l'identità del ristorante**, assicurandosi che l'address del ristorante sia valido, attivo e ancora correttamente registrato all'interno del sistema;
- **Verifica che esista almeno un token valido associato a quello specifico utente e ristorante**, confermando così il diritto a pubblicare una recensione basata su un'esperienza reale;

Se tutte le verifiche sopra sono superate con successo, lo smart contract procede come segue:

- **Decrementa il contatore di token** associato alla combinazione specifica utente–ristorante, presente nello smart contract di gestione token;
- **Assegna un identificativo univoco (ID)** alla nuova recensione;
- **Memorizza la recensione** all'interno di una struttura dati principale che indicizza le recensioni in base al loro ID;
- **Aggiorna le strutture secondarie**:
    - La recensione viene aggiunta alla struttura che mappa tutte le recensioni riferite a un determinato ristorante;
    - La recensione viene anche aggiunta alla struttura che raccoglie tutte le recensioni lasciate da un singolo utente verso un determinato ristorante;
- **Restituisce l'identificativo della recensione** all'utente.

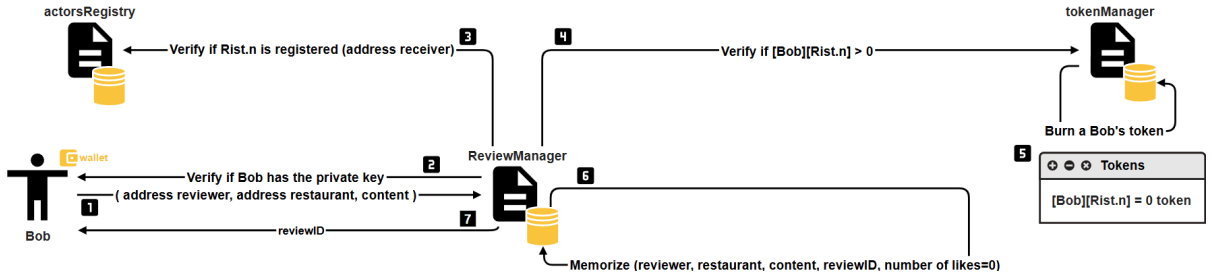



*Figure 5. Schema riassuntivo del processo di pubblicazione*

## 2.5.5 - Modifica o Revoca delle recensioni

Le operazioni di **modifica** o **revoca** delle recensioni sono regolate da un meccanismo che garantisce la **trasparenza** e la **tracciabilità** di ogni azione.

### Revoca della recensione

La **revoca di una recensione** già pubblicata è un'operazione che consente all'utente di rimuovere una recensione dal sistema, garantendo che ogni azione sia tracciabile e giustificata. La revoca può essere effettuata solo dall'utente che ha originariamente pubblicato la recensione, per evitare manipolazioni o

modifiche non autorizzate. Per avviare il processo di revoca, l'utente deve fornire allo smart contract le seguenti informazioni:

- L'**identificativo univoco della recensione** da revocare;

Lo smart contract esegue quindi le seguenti verifiche:

- Verifica che **esista una recensione associata all'identificativo fornito**. Se la recensione non esiste, il sistema non permette la revoca;
- Controlla che l'address contenuto nella recensione corrisponda effettivamente all'identità dell'utente che sta richiedendo la revoca.
- **Verifica del possesso dell'address dell'utente**.

Se tutte le condizioni sono soddisfatte, il sistema procede con le operazioni di :

- **Recupero della recensione da revocare**: Lo smart contract localizza la recensione che l'utente desidera revocare, utilizzando l'identificativo univoco.
- **Rimozione dalla struttura principale delle recensioni**: la recensione viene eliminata dalla struttura dati centrale che conserva tutte le recensioni nel sistema, in modo che non risulti più accessibile;
- **Rimozione dalle strutture secondarie**: la recensione viene anche rimossa dalle strutture secondarie che gestiscono l'indicizzazione delle recensioni per **ristorante** e **utente-ristorante**. Questo assicura che tutte le aggregazioni e mappature delle recensioni siano aggiornate correttamente.

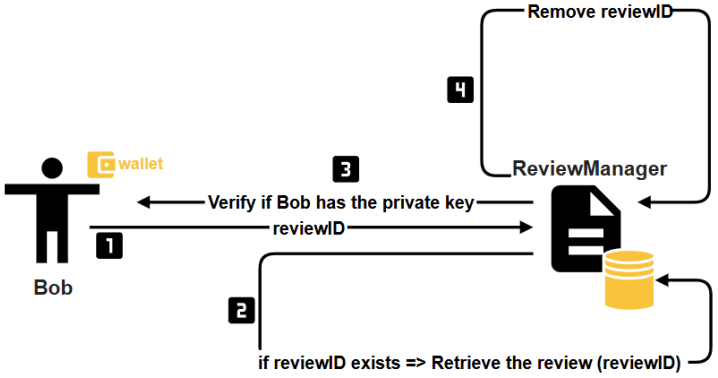



*Figure 6. Schema riassuntivo della revoca della recensione*

## Modifica della recensione

Le modifiche alle recensioni possono essere eseguite solo in determinate circostanze, per prevenire abusi e preservare l'integrità del sistema.

- Ogni recensione può essere modificata solo entro un periodo di tempo definito al momento della sua pubblicazione. Una volta che questo periodo scade, la recensione non sarà più modificabile;

Per effettuare una modifica su una recensione esistente, l'utente deve fornire allo smart contract le seguenti informazioni:

- **L’identificativo univoco della recensione**, per individuare e identificare correttamente la recensione che si desidera modificare;
- **Il nuovo testo della recensione**;

Lo smart contract verifica i seguenti requisiti prima di consentire la modifica:

- **Verifica l’esistenza della recensione,** confermando che la recensione associata all’identificativo fornito esista effettivamente nel sistema.
- **Controlla l’identità dell’utente**, confrontando l’address presente nella recensione con l’address dell’utente che sta richiedendo la modifica.
- **Verifica del possesso dell’address dell’utente**.
- Verifica che **non sia terminato l’intervallo di tempo** ammissibile per modificare la recensione.

Se tutte le verifiche vanno a buon fine, lo smart contract esegue le seguenti operazioni:

- **Recupero della recensione da modificare**: Lo smart contract localizza la recensione che l'utente desidera modificare, utilizzando l’identificativo univoco.
- **Modifica della recensione nella struttura centrale**: Il nuovo testo fornito dall'utente viene registrato nella struttura dati centrale che conserva tutte le recensioni nel sistema, sostituendo il contenuto precedente.
- **Rimozione dei like**: Se la recensione aveva ricevuto dei like, questi vengono azzerati.

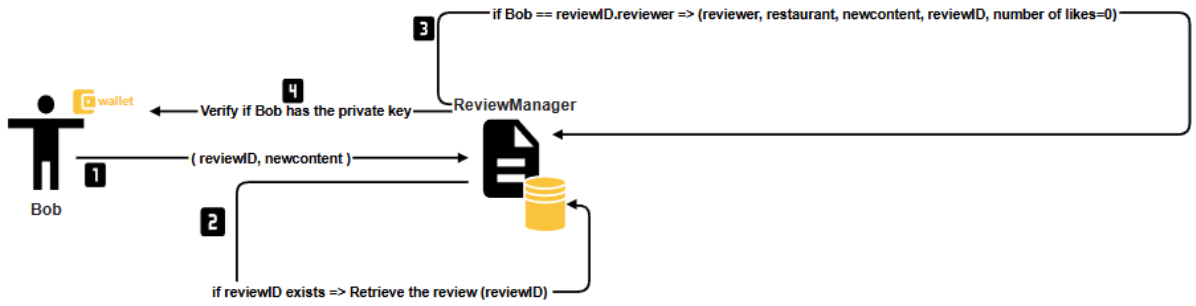



*Figure 7. Schema riassuntivo della modifica della recensione*

## 2.5.6 - Sistema di valutazione delle recensioni

Attraverso il sistema di votazione, si mira a riconoscere e valorizzare gli utenti che contribuiscono in modo sincero e costruttivo, favorendo così la creazione di una community coesa e affidabile.

### Voto positivo (like) su una recensione

Per esprimere un voto positivo (like) su una recensione, l’utente deve fornire allo smart contract le seguenti informazioni:

- Il **proprio address**;
- L’**identificativo della recensione** che si intende votare.

Lo smart contract esegue le seguenti verifiche e operazioni:

- **Recupero della recensione da votare**: Lo smart contract localizza la recensione che l'utente desidera votare, utilizzando l'identificativo univoco;
- **Estrae l'address del ristorante** associato alla recensione che si intende votare;
- **Verifica che l'utente abbia un token per il ristorante a cui riferisce la recensione.**
- **Controlla che l'utente sia effettivamente il titolare dell'address**, tramite una prova crittografia basata sul possesso della chiave privata corrispondente;
- Verifica, attraverso la struttura secondaria alla recensione, che l'utente **non abbia già espresso un voto per quest'ultima**;

Se tutte le condizioni sono soddisfatte, il contratto:

- **Aggiorna la struttura secondaria alla recensione** per indicare che l'utente ha già votato quella recensione.
- **Aggiorna il numero di like** associato alla recensione.

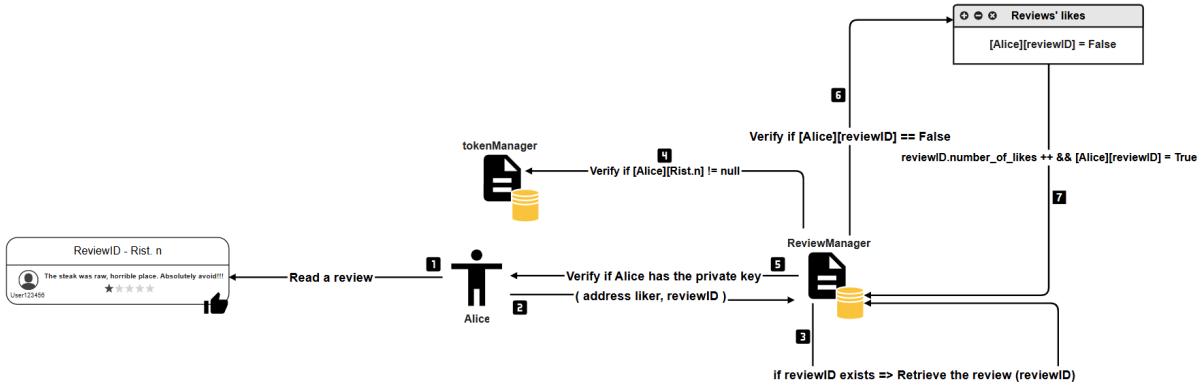



*Figure 8. Schema riassuntivo dell'apposizione di un voto su di una recensione*

## 2.5.7 - Incentivazione per la partecipazione al sistema

Il sistema premia gli utenti che hanno avuto una partecipazione attiva in quest'ultimo. Ogni N recensioni di un utente ad un ristorante rendono il reviewer abilitato a ricevere il voucher di sconto rappresentato da un NFT, valido esclusivamente presso il ristorante oggetto delle recensioni. L'intero processo è gestito da uno smart contract.

Gli NFT sono gestiti all'interno dello smart contract tramite una **struttura (mapping) principale** che associa ciascun voucher all'address dell'utente che lo possiede.

### Emissione dell'NFT per l'autore della recensione

Per generare l'NFT, lo smart contract prende in input dalle operazioni precedenti:

- L'**address dell'utente autore della recensione (reviewer)**;
- L'**address del ristorante** associato alla recensione.

Se tutte le condizioni espresse nella sezione precedente sono soddisfatte, il sistema procede con la seguente operazione:

- **Emette un NFT (voucher di sconto)** contenente una determinata percentuale di sconto. Il bonus viene associato **esclusivamente all'utente che ha scritto la recensione** e al ristorante corretto;

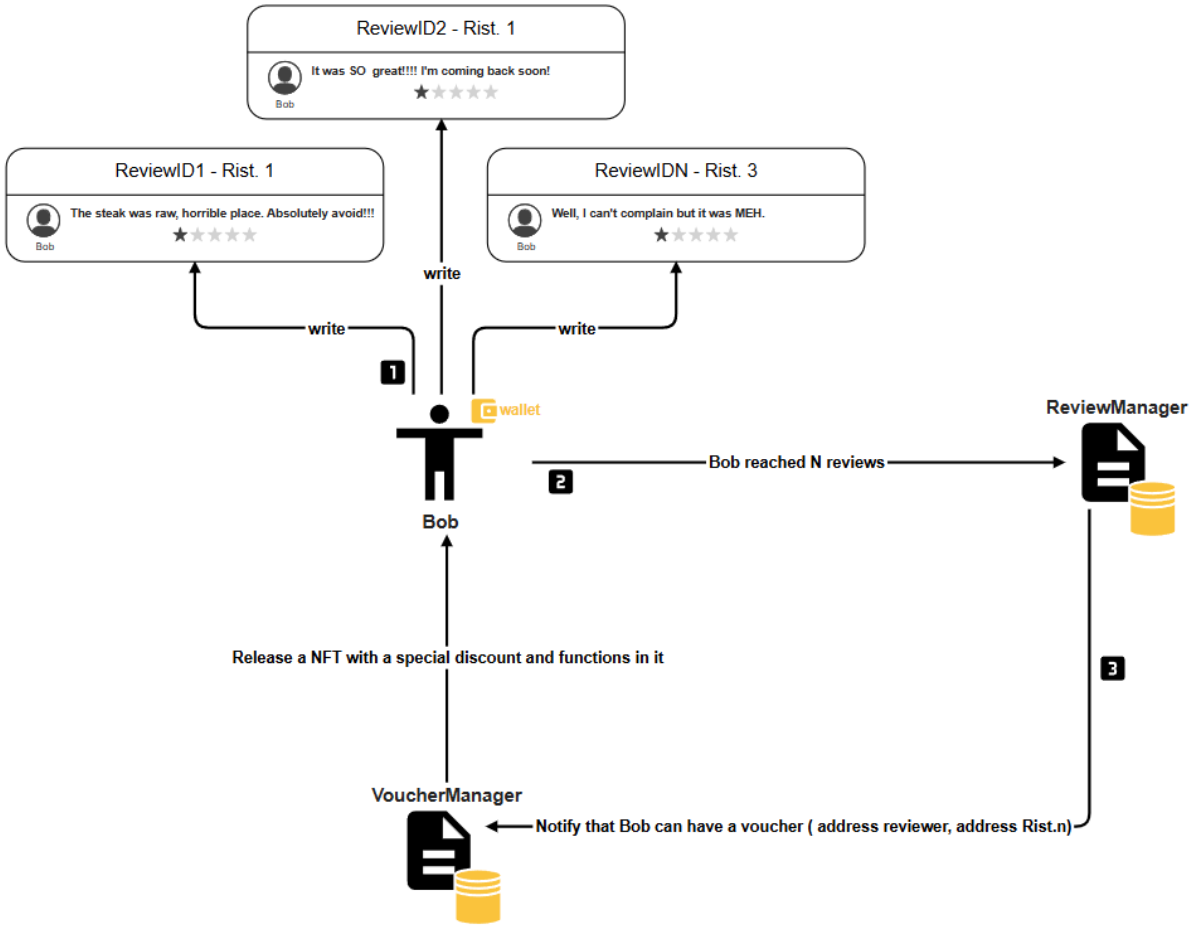



*Figure 9. Schema riassuntivo dell'emissione del VoucherNFT*

## Utilizzo dell'NFT

Per utilizzare l'NFT come **voucher di sconto**, l'utente deve fornire allo smart contract il suddetto voucher (NFT) ricevuto.
Lo smart contract esegue le seguenti verifiche in aggiunta a quelle già precedentemente definite per i pagamenti:

- **Controlla che l'address del ristorante nel voucher corrisponda** al ristorante presso il quale si sta effettuando il pagamento;
- **Controlla che l'address dell'utente nel voucher corrisponda** all'utente che sta effettuando il pagamento

Se tutte le condizioni sono soddisfatte, il sistema considera valido il voucher e **applica lo sconto previsto al pagamento**.

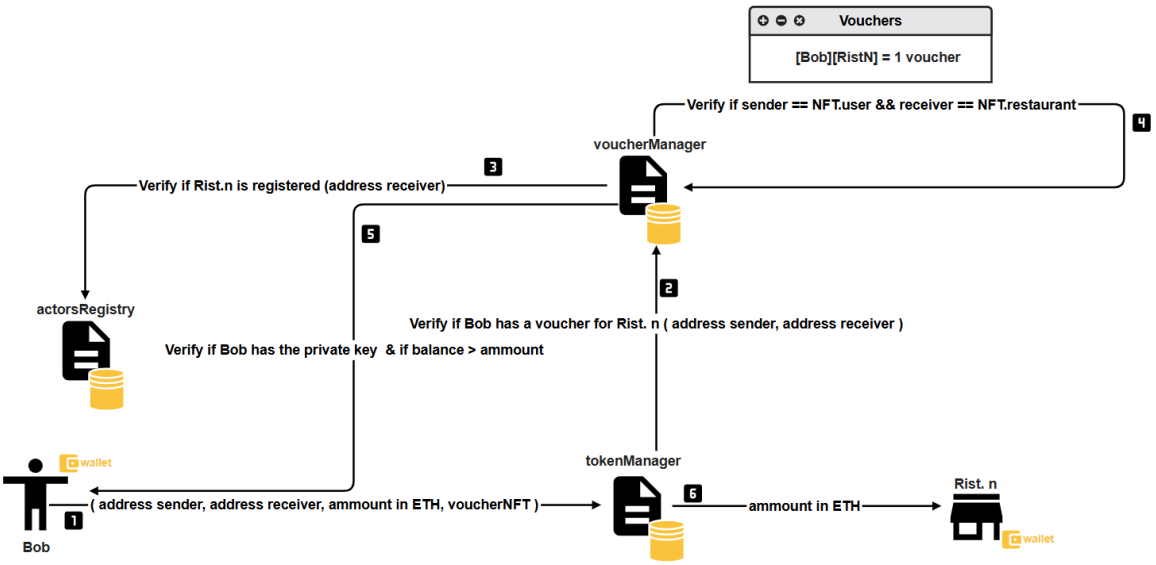



*Figure 10. Schema riassuntivo dell'utilizzo di un VoucherNFT da parte di un customer*

# WP3 - ANALISI DELLA SICUREZZA

In questo capitolo andremo ad analizzare, attaccante per attaccante, l'efficacia della soluzione descritta nel WP2 rispetto alle principali minacce identificate. Concluderemo con una riflessione complessiva in cui esamineremo le criticità residue e i limiti intrinseci di un approccio che deve convivere con componenti off-chain non controllabili.

## 3.1 VALUTAZIONE DEGLI ATTACCANTI

In questa sezione analizziamo ciascun scenario di attacco descrivendo il comportamento dell'avversario, le funzionalità e le proprietà del sistema coinvolte, e le contromisure adottate. Concludiamo valutando se la soluzione sia in grado di bloccare formalmente l'attacco o se si limiti a mitigarne gli effetti.

### 3.1.1 Fake reviewer

**descrizione:**

Utente che scrive una recensione senza aver mai acquistato o usufruito del servizio del seller recensito.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni,
- F4 - Verifica della prova d'acquisto

**proprietà attaccate:**

- Buy proof property

**soluzione:**

L'attaccante non riesce a violare la proprietà di Buy Proof perché, durante la fase di acquisto, viene rilasciato un token come prova dell'avvenuto acquisto. Non possedendo questo token, l'attaccante non è in grado di scrivere recensioni, in quanto privo della necessaria prova di acquisto.

### 3.1.2 Sybil Attack

**descrizione:**

Attaccante che crea numerose identità falsificate per manipolare o influenzare il sistema, scrivendo numerose recensioni favorevoli o sfavorevoli ai ristoranti.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni,
- F4 - Verifica della prova d'acquisto

**proprietà attaccate:**

- AntiFakeAccount property

**soluzione:**

Questo tipo di attacco è mitigato dal fatto che, anche se un utente o un ristorante possono creare più wallet per simulare identità diverse, è comunque necessario effettuare una transazione reale presso un ristorante per ottenere i token. Ciò comporta una spesa concreta in Ethereum. Anche nel caso in cui sia il ristorante stesso a generare account falsi e simulare transazioni al fine di ottenere recensioni positive, dovrà comunque sostenere dei costi reali, come le commissioni di rete (gas fee) e le eventuali imposte derivanti dalla conversione degli Ethereum in valuta fiat. In questo modo, ogni tentativo di manipolazione del sistema comporta una perdita economica concreta, scoraggiando comportamenti fraudolenti.

### 3.1.3 Phantom Seller

**descrizione:**

Ristoratore fittizio che si presenta come attività legittima pur non offrendo alcun servizio reale.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni,
- F4 - Verifica della prova d'acquisto

**proprietà attaccate:**

- Autenticità

**soluzione:**

Questo tipo di attaccante non rappresenta una minaccia per il nostro sistema, poiché ogni ristorante deve necessariamente identificarsi per poter ricevere recensioni. L'identificazione richiede la presentazione della Partita IVA all'ente certificatore, il quale è in grado di verificarne la validità. Di conseguenza, eventuali Partite IVA false o inesistenti possono essere facilmente rilevate e bloccate, impedendo l'accesso al sistema a soggetti non autorizzati.

### 3.1.4 Review Repeater

**descrizione:**

Cliente che, dopo aver pubblicato una recensione valida, tenta di inviarne ulteriori per la stessa visita o spesa, sfruttando ripetutamente la medesima prova d'acquisto.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni,
- F4 - Verifica della prova d'acquisto

**proprietà attaccate:**

- Buy proof property

**soluzione:**

Tale avversario non ha alcun margine di attacco nel sistema, poiché la soluzione adottata prevede l'utilizzo di una prova d'acquisto sotto forma di token, che viene automaticamente bruciato una volta utilizzato per pubblicare una recensione. Questa strategia impedisce il riutilizzo della prova d'acquisto, evitando che un utente — intenzionalmente o per errore — possa rilasciare più di una recensione associata allo stesso acquisto.

## 3.1.5 Proxy Reviewer

**descrizione:**

Cliente che, dopo aver acquistato un servizio, per pigrizia o per motivi strategici, cede ad altri il diritto o la possibilità di lasciare una recensione, delegando così l'opinione a soggetti terzi.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni,
- F4 - Verifica della prova d'acquisto

**proprietà attaccate:**

- Buy proof property
- Non ripudio

**soluzione:**

L'attaccante non riesce ad avere successo perché, durante la fase di acquisto, viene rilasciato un token che attesta l'effettiva spesa effettuata presso uno specifico ristorante. Questo token è immutabile, non trasferibile ed è strettamente associato alla coppia utente–ristorante. Di conseguenza, la proprietà di Buy Proof risulta garantita e non può essere violata.

Per quanto riguarda la proprietà del Non Ripudio, un potenziale problema potrebbe sorgere solo nel caso in cui l'utente cedesse volontariamente il proprio token e fornisse anche l'accesso alla chiave privata del proprio wallet. Tuttavia, a meno che non si verifichi questa condizione molto specifica, il sistema è in grado di preservare anche questa proprietà, rendendo inefficaci gli attacchi di questo tipo.

## 3.1.6 Dishonest Reviewer

**descrizione:**

Cliente che ha realmente usufruito del servizio ma pubblica una recensione ingannevole o non corrispondente alla realtà, per motivi personali o esterni.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni

**proprietà attaccate:**

-

**soluzione:**

Il sistema non è in grado di difendersi da questo tipo di avversario, poiché non dispone di strumenti per verificare la veridicità del giudizio espresso dall'utente. Allo stesso modo, non può intervenire nei casi in cui altri utenti, leggendo una recensione potenzialmente falsa, decidano comunque di assegnare un voto positivo. L'utilità percepita di una recensione, infatti, è un criterio intrinsecamente soggettivo e non oggettivamente valutabile dal sistema.

## 3.1.7 Proof Broker

**descrizione:**

Ristoratore che condivide la propria prova d'acquisto con un altro esercente, che la distribuisce a clienti ignari per ottenere recensioni ingannevoli sul primo seller.

**funzionalità attaccate:**

- F3 - Scrittura delle recensioni,
- F4 - Verifica della prova d'acquisto

**proprietà attaccate:**

- Usage property

**soluzione:**

Il sistema impedisce all'attaccante di portare a termine l'attacco, in quanto il token non viene effettivamente rilasciato all'utente, ma viene invece mantenuto all'interno di uno smart contract dedicato, attraverso un'associazione univoca tra utente e ristorante. In questo modo, il controllo del token resta vincolato al contratto intelligente, prevenendo tentativi di manipolazione o trasferimento illecito e rafforzando la sicurezza del sistema.

## 3.1.8 Privacy Violator

**descrizione:**

Attaccante esterno che mira a raccogliere dati sensibili degli utenti, sfruttando vulnerabilità e informazioni di dominio pubblico nel sistema per compromettere la loro privacy.

**funzionalità attaccate:**

- F9 - Gestione della privacy

**proprietà attaccate:**

- Privacy
- Confidenzialità

**soluzione:**

Questo tipo di attaccante non è in grado di violare le proprietà di privacy e confidenzialità, poiché il sistema garantisce agli utenti un livello di pseudo-anonimato.

All’interno della blockchain, infatti, le identità reali non vengono registrate direttamente: ogni utente è rappresentato da un wallet address, che non contiene informazioni personali identificabili. Di conseguenza, anche se l’attaccante ha accesso a tutte le informazioni pubbliche sulla blockchain, non può associare con certezza tali indirizzi a persone fisiche, salvo errori o negligenze da parte dell’utente stesso.

Tuttavia, è importante sottolineare che lo pseudo-anonimato non equivale a non tracciabilità: tutte le transazioni sono pubbliche, immutabili e permanentemente registrate. Questo significa che è possibile ricostruire l’intera attività di un indirizzo specifico, e in alcuni casi, dedurne l’identità reale attraverso analisi comportamentali o l’incrocio con dati esterni. Pertanto, pur garantendo una buona protezione della privacy, il sistema non è completamente immune da attacchi avanzati di profilazione e dunque non riesce totalmente a garantire la *unlinkability*. Nonostante ciò, nel caso in cui l’utente sia sufficientemente accorto da generare un wallet diverso per ogni pagamento, la *unlinkability* può essere effettivamente raggiunta, rendendo complicato collegare tra loro le varie transazioni.

## 3.1.9 Reputation Manipulator

### descrizione:

Seller che corrompe gli utenti per ottenere recensioni positive sulla propria attività e/o per aumentare il numero di recensioni positive.

### funzionalità attaccate:

- F3 - Scrittura delle recensioni,

### proprietà attaccate:

-

### soluzione:

L’attacco in questione non può essere evitato, poiché avviene off-chain, ovvero al di fuori dell’infrastruttura tecnica del sistema, rendendolo non rilevabile né gestibile dal meccanismo on-chain.

L’unica forma di mitigazione possibile consiste nella presenza di un numero sufficiente di recensioni autentiche e imparziali, capaci di controbilanciare l’effetto delle recensioni fraudolente. In questo senso, il sistema fa affidamento sul comportamento virtuoso della maggioranza degli utenti per mantenere l’integrità complessiva delle valutazioni.

Tuttavia, dal punto di vista formale, la soluzione non riesce a coprire questo tipo di attacco, poiché non esistono garanzie tecniche o meccanismi automatici che possano prevenirlo o neutralizzarlo in modo diretto.

## 3.1.10 Review Alterer

### descrizione:

Utente che modifica o manipola il contenuto di una recensione altrui.

### funzionalità attaccate:

- F5 - Modifica/Revoca recensioni

### proprietà attaccate:

- Integrità
- Non ripudio
- Trasparenza

### soluzione:

L'attaccante non riesce a portare a termine il suo attacco, in quanto la modifica di una recensione è consentita esclusivamente al suo autore. Ogni recensione è infatti associata all'address dell'utente che l'ha pubblicata e, per poter apportare modifiche, è necessario superare una verifica crittografica che dimostri il possesso della corrispondente chiave privata.

Non disponendo di tale chiave, l'attaccante non è in grado di autenticarsi e quindi non può effettuare alcuna modifica.

Anche nell'eventualità remota in cui un attaccante riuscisse a superare i meccanismi di sicurezza, il sistema prevede un'ulteriore misura di mitigazione: la possibilità di modificare una recensione è limitata a un intervallo di tempo definito. Ciò riduce sensibilmente la finestra temporale durante la quale è possibile compiere azioni malevole, limitando l'impatto potenziale dell'attacco.

## 3.1.11 Replay Attacker

### descrizione:

Utente che intercetta un operazione fatta da un altro utente e cerca di utilizzarla per compromettere la credibilità del sistema.

### funzionalità attaccate:

- F5 - Modifica/Revoca recensioni
- F1 - Operazione di Pagamento
- F3 - Scrittura delle recensioni
- F6 - Sistema di valutazione dell'utilità delle recensioni

### proprietà attaccate:

- Integrità
- Non ripudio

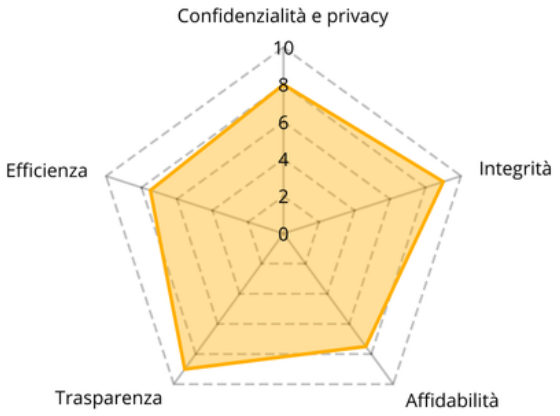
**soluzione:**

Questo tipo di attacco non va a buon fine, poiché qualsiasi operazione associata a un utente richiede una prova crittografica, atta a verificare che sia effettivamente quell'utente a compierla.

Ogni azione è infatti vincolata alla firma digitale generata tramite la chiave privata dell'utente, che rappresenta univocamente la sua identità. In assenza di tale chiave, un attaccante non è in grado di autenticarsi e, di conseguenza, non può eseguire operazioni né impersonare l'utente.

In questo modo le operazioni non possono essere alterate da terzi, poiché ogni modifica richiede una firma valida, e l'autore di un'azione non può negarne la paternità, essendo la firma crittografica prova inequivocabile del suo coinvolgimento.

## 3.2 RIEPILOGO E OSSERVAZIONI FINALI




Riepilogando, i principali punti di forza della nostra soluzione risiedono nell'elevato livello di **integrità** e **trasparenza** garantito dall'uso della blockchain e della crittografia. Questi strumenti forniscono solide garanzie di sicurezza, ma comportano un inevitabile compromesso in termini di efficienza, leggermente sacrificata rispetto a soluzioni centralizzate.

Sul fronte della **privacy** e della **confidenzialità**, il sistema offre buone tutele grazie all'impiego di indirizzi pseudonimi e all'assenza di dati personali registrati on-chain. Tuttavia, la proprietà di **unlinkability** non è pienamente garantita: la natura pubblica e permanente della blockchain consente, in alcuni casi, di collegare le attività di uno stesso wallet tramite analisi comportamentali o incrocio con dati esterni. Nonostante ciò, il sistema fornisce un livello di anonimato adeguato per l'utente medio, che ha comunque la possibilità di utilizzare più wallet per aumentare la propria riservatezza.

Per quanto riguarda l'**affidabilità** del sistema di recensioni, il quadro è più sfumato. Il sistema non può garantire che ogni recensione sia onesta, utile o libera da influenze esterne, come ad esempio seller che incentivano o producono recensioni tramite account falsi. Inoltre l'utilità di una recensione è soggettiva e difficilmente valutabile in modo oggettivo. Questo può incidere sulla percezione di affidabilità da parte degli utenti. Tuttavia, il sistema punta a incentivare la partecipazione e la creazione di una community attiva e collaborativa: in uno scenario in cui la maggioranza degli utenti agisce in buona fede, le recensioni fraudolente tendono a perdere peso e visibilità nel tempo.

# WP4 - IMPLEMENTAZIONE

In questo capitolo vengono descritte nel dettaglio le principali scelte progettuali adottate durante l'implementazione della soluzione delineata nel WP2. Verranno analizzate la struttura, le componenti principali e le funzionalità dei contratti smart sviluppati, evidenziandone le logiche operative e le motivazioni alla base delle decisioni architetturali. Nella parte conclusiva del capitolo, sarà presentata un'analisi delle prestazioni del sistema.

**Link Github**

## 4.1 Premesse tecniche

Per lo sviluppo e il test della soluzione è stato utilizzato **Ganache**, un ambiente di sviluppo personale per la blockchain Ethereum che consente di simulare una rete locale. Questo ha permesso di eseguire transazioni e testare i contratti in modo rapido e controllato, senza costi reali in termini di gas.

I **contratti intelligenti (smart contracts)** sono stati sviluppati utilizzando il linguaggio **Solidity**, lo standard de facto per la programmazione su Ethereum. Solidity consente di definire la logica di business e le condizioni necessarie all'esecuzione delle varie operazioni sulla blockchain.

### Assenza di un'interfaccia grafica

Per ragioni di semplicità si è scelto di non sviluppare un'interfaccia grafica a supporto della soluzione. Sebbene una UI rappresenti un elemento essenziale per rendere la DApp accessibile e usabile da parte dell'utente finale, in questa fase il focus è stato volutamente limitato alla definizione dell'architettura dei contratti smart, rinviando l'integrazione front-end a una possibile fase successiva del progetto.

### Divisione dei contratti

A causa di alcune limitazioni di Ganache, i contratti non possono superare la dimensione massima di **24 KB**. Per questo motivo, si è reso necessario suddividere il contratto principale, che risultava troppo grande, in più contratti di dimensioni minori. Contestualmente, è stata ottimizzata la logica complessiva, alleggerendo i controlli e distribuendo alcune funzionalità tra i contratti secondari. Ad esempio, la funzione relativa all'applicazione dei voucher è stata spostata all'interno del contratto VoucherManager.sol.

## 4.2 Principali componenti della soluzione

La soluzione implementata si compone di diversi moduli, ciascuno dei quali ha un ruolo specifico all'interno del sistema e contribuisce al corretto funzionamento del flusso operativo. Di seguito vengono descritte le principali.

## 4.2.1 CertifiedAuthority.sol

La **Certified Authority** è il modulo responsabile della validazione delle identità nel sistema. Il suo compito principale è garantire che solo soggetti autorizzati — ovvero aziende o enti con Partita IVA valida — possano interagire con i contratti intelligenti.

### Implementazione attuale

La **Certified Authority (CA)** rappresenta l'entità responsabile della validazione delle identità e dell'autenticità delle Partite IVA coinvolte nel sistema. In un'implementazione reale e distribuita, questo ruolo dovrebbe essere svolto da un **oracolo**, ovvero un meccanismo che permette alla blockchain di interagire in modo affidabile con dati esterni.

Tuttavia, dato che l'implementazione e la gestione sicura di un oracolo comportano una complessità non banale — sia in termini tecnici sia di sicurezza —, in questa fase prototipale si è scelto di **simulare il comportamento dell'oracolo** direttamente all'interno del contratto che funge da Certified Authority.

Per farlo, è stato implementato un **mapping** in Solidity che associa ciascun indirizzo a una partita IVA valida. Questo mapping funge da registro delle entità autorizzate e viene inizializzato manualmente con i dati noti.

## 4.2.2 ActorRegistry.sol

Il modulo **Actor Registry** ha il compito di gestire l'elenco dei ristoranti affiliati al sistema, ovvero tutte le entità che possono essere recensite dagli utenti all'interno della piattaforma. La presenza di un registro centralizzato e verificabile è fondamentale per garantire la correttezza e la trasparenza del sistema di recensioni.

### Dettagli di implementazione

Il registro è implementato tramite un **mapping** in Solidity che associa ogni ristorante affiliato al suo **indirizzo wallet**. Questo consente ai contratti intelligenti di verificare in modo immediato se un determinato indirizzo è effettivamente parte del sistema.
L'iscrizione di un ristorante al sistema avviene attraverso una funzione di registrazione che può essere invocata esclusivamente da un amministratore autorizzato (tipicamente identificato tramite msg.sender). Durante la fase di iscrizione viene effettuato un controllo incrociato con il contratto della Certified Authority. In particolare, prima di aggiungere un ristorante al sistema, si verifica che la Partita IVA associata all'indirizzo sia presente e valida all'interno del mapping gestito dalla CA. Solo nel caso in cui la verifica abbia esito positivo, il ristorante viene effettivamente registrato nel mapping del modulo Actor Registry.

### Considerazioni sulla privacy

È importante osservare che la **pubblicazione degli indirizzi wallet dei ristoranti** può sollevare alcune preoccupazioni in termini di privacy. Tuttavia, va evidenziato che:

- L'elenco dei ristoranti affiliati è un **requisito funzionale** necessario: senza questo elenco, gli utenti non potrebbero sapere quali strutture è possibile recensire o consultare.
- Gli indirizzi sulla blockchain sono pseudonimi e non direttamente riconducibili a dati personali, a meno che non vengano collegati a informazioni off-chain.
- Ogni ristorante ha la **possibilità di aggiornare regolarmente il proprio indirizzo wallet**, offrendo così un livello minimo di anonimato dinamico, qualora desiderato.

### 4.2.3 TokenManager.sol

Il contratto **TokenManger** rappresenta il modulo incaricato di **gestire i pagamenti** tra utenti e ristoranti, integrando in modo trasparente il **sistema di sconti** basato su voucher (NFT) e il **rilascio di un token** che funge da prova digitale della transazione. Alla base del contratto vi è una struttura mapping nidificata che tiene traccia, per ogni utente, del **numero di token ricevuti da ciascun ristorante**.

#### Token non standard

Una delle scelte progettuali più significative è stata quella di non ricorrere a standard token come ERC20 o ERC721 per rappresentare i token transazionali. In alternativa, si è deciso di utilizzare un semplice mapping interno. Sebbene questa soluzione limiti l'interoperabilità con altri marketplace esterni, garantisce una maggiore efficienza in termini di gas e una minore complessità nel controllo dello stato, risultando perfettamente adeguata per il nostro caso d'uso specifico in cui i token hanno valore solo all'interno del sistema.

#### Emissione dei token

All'interno del contratto TokenManager.sol, l'emissione dei token è strettamente subordinata al buon esito della transazione economica tra utente e ristorante. In altre parole, nessun token viene generato se il trasferimento di fondi non va a buon fine. Questo tipo di scelta è stata intrapresa perché:

- Previene situazioni anomale in cui un utente riceva un token senza aver effettuato alcun pagamento.
- Rende la logica del sistema robusta e verificabile a posteriori.
- Permette l'eventuale implementazione di controlli aggiuntivi.

#### Implementazione della logica di rimborso

Nel contesto dei pagamenti su blockchain, può verificarsi la situazione in cui un utente invii un **importo superiore a quello effettivamente richiesto** per una determinata transazione. Questo comportamento può essere accidentale — ad esempio a causa di un errore dell'utente — oppure intenzionale, per garantire che la transazione venga elaborata correttamente anche in presenza di fluttuazioni nei costi accessori. In entrambi i casi, è fondamentale garantire la restituzione automatica dell'importo in eccesso, al fine di mantenere l'equità del sistema.
All'interno della funzione di pagamento è stata introdotta esplicitamente una l**ogica di calcolo e rimborso** del resto, nel caso in cui l'importo trasferito (**msg.value**) sia superiore all'importo dovuto (**amount**). La differenza tra le due cifre viene identificata come "resto" e immediatamente restituita al mittente della transazione (**msg.sender**).

#### Suddivisione per limiti tecnici

Come già spiegato, durante la fase di sviluppo e test, è emerso che **Ganache impone una dimensione massima per ciascun contratto**. Per rispettare questo vincolo, si è deciso di **estrarre alcune funzionalità secondarie** dal contratto TokenManager.sol e di distribuirle su moduli separati. In particolare, la funzione **applyVoucher**, inizialmente pensata come parte integrante del flusso di pagamento, è stata trasferita nel contratto VoucherManager.sol.

## 4.2.4 VoucherManager.sol

Il contratto **VoucherManager** è stato progettato per gestire l'emissione e il tracciamento dei voucher utilizzabili nel sistema come sconti durante i pagamenti.

### Struttura dei voucher

Ogni voucher è rappresentato da una struttura **Voucher** che contiene informazioni essenziali come un identificativo progressivo, il proprietario (cliente), il ristorante a cui è associato, l'ammontare dello sconto e un campo metadataURI pensato per contenere dati aggiuntivi, ad esempio sotto forma di metadati JSON accessibili da IPFS o da un server esterno. Questa struttura è memorizzata in un mapping che consente l'accesso diretto a ogni voucher tramite il suo **ID**.

### Usabilità del voucher

Ogni voucher emesso all'interno del sistema è dotato di un **attributo interno** denominato **available**, il cui valore booleano riflette la disponibilità effettiva del voucher. Questo campo svolge un ruolo fondamentale nel determinare se il voucher sia ancora utilizzabile o se, al contrario, sia già stato consumato in una transazione. All'atto dell'emissione, il campo available viene impostato a true; nel momento in cui il voucher viene utilizzato per ottenere uno sconto, tale valore viene immediatamente aggiornato a false, rendendolo inutilizzabile per eventuali pagamenti futuri. Per evitare riutilizzi fraudolenti o errori di esecuzione, la funzione applyVoucher include un controllo esplicito sullo stato del campo available. Qualora il voucher risultasse già utilizzato (cioè se **available == false**), l'invocazione della funzione fallisce automaticamente, impedendo qualsiasi applicazione dello sconto o prosecuzione della transazione.
A completamento del ciclo di vita del voucher, nel momento in cui questo viene applicato con successo e marcato come non più disponibile, il contratto **emette un evento specifico sulla blockchain**.

## 4.2.5 ReviewManager.sol

Il contratto **ReviewManager** è il cuore del sottosistema dedicato alle recensioni e nasce con l'obiettivo di gestire in modo efficiente la pubblicazione, la modifica, la cancellazione e l'interazione (like) sui contenuti generati dagli utenti.

**Autorizzazioni e controllo degli accessi**
Nel contratto è stato introdotto un **modifier** specifico per garantire che **solo l'autore della recensione possa modificarla o cancellarla**. Questo accorgimento progettuale rafforza l'integrità del sistema, impedendo ad attori esterni di alterare contenuti altrui e prevenendo comportamenti malevoli. L'utilizzo di tale controllo consente di legare in modo diretto ogni azione sensibile all'identità del wallet che ha creato originariamente la recensione, sfruttando il meccanismo intrinseco di autenticazione delle transazioni Ethereum. La verifica viene applicata direttamente all'interno delle funzioni modifyReview e deleteReview, garantendo che ogni operazione di aggiornamento sia coerente con l'autore effettivo del contenuto.
In aggiunta, è stato definito un **secondo modifier** che impone un **limite temporale alla modifica**: ogni recensione può essere modificata solo entro una finestra temporale specifica dalla sua creazione (24 ore). Trascorso tale intervallo, la funzione modifyReview non può più essere invocata con successo.

### Divisione dei contratti

A causa delle limitazioni imposte dalla rete di sviluppo Ganache, in particolare il vincolo di 24 KB sulla dimensione massima dei contratti, si è reso necessario **suddividere le funzionalità** inizialmente previste nel contratto ReviewManager.sol in due componenti distinti.
Il contratto **ReviewManager.sol** conserva le funzionalità principali relative alla gestione diretta delle recensioni, come la creazione, la modifica, la cancellazione e l'interazione tramite like. Le funzionalità secondarie e di supporto sono state invece estratte e implementate nel contratto ausiliario **SupportReviewManager.sol.**

Inoltre, per ragioni legate sempre alla limitazione di spazio, si è deciso di non includere nell'implementazione le due strutture secondarie inizialmente prevista per mantenere rispettivamente la **lista degli account che hanno espresso un like su una determinata recensione** e **le liste delle recensioni relative ai singoli ristoranti**. Queste funzionalità non erano strettamente necessarie per il corretto funzionamento delle operazioni di base e, di conseguenza, sono state escluse insieme alle funzionalità ad essa collegate, come la possibilità di recuperare l'elenco completo degli utenti che hanno messo like a una recensione. Nel caso in cui si volesse implementare questa funzionalità in futuro, sarebbe sufficiente introdurre una semplice struttura dati del tipo *mapping(IDreview => address[]),* che permetterebbe di associare a ogni recensione l'elenco dei wallet che hanno interagito con essa tramite like.

### Semplificazione condizione per l'emissione del voucher

A causa di vincoli di spazio nel deploy dei contratti, abbiamo adottato una logica semplificata per l'emissione dei voucher rispetto a quanto previsto nel WP2. In particolare, anziché emettere il voucher al raggiungimento della *N*-esima recensione effettuata da un utente su un determinato ristorante, **il voucher viene ora generato al raggiungimento di *N* like su una singola recensione** dell'utente.

L'implementazione dell'approccio originario non è stata possibile, in quanto avrebbe richiesto una **struttura ausiliaria** — precedentemente esclusa — necessaria per tracciare il numero di recensioni lasciate da ciascun utente per ogni ristorante.

## 4.2.6 Scelte generali valide per tutti i contratti

1. **Controlli fail-fast:** tutte le istruzioni require sono state posizionate all'inizio delle funzioni, per verificare immediatamente la validità degli input. Questo approccio riduce lo spreco di gas evitando l'esecuzione di operazioni costose in caso di errore.
2. **Minimizzazione dello stato:** i contratti evitano di salvare dati ridondanti. Informazioni già reperibili da altri moduli (es. dettagli del voucher) vengono ottenute on-demand tramite chiamate, riducendo lo spazio occupato e la complessità della logica interna.
3. **Architettura modulare:** la logica dell'applicazione è stata progettata per essere suddivisa il più possibile in diversi contratti indipendenti, ciascuno responsabile di una singola funzionalità. Questa scelta, in condizioni in cui la strumentazione permette, incrementa la leggibilità, favorisce la manutenibilità e semplifica futuri aggiornamenti o estensioni.

### 4.2.7 Possibili miglioramenti futuri

In questa sezione vengono elencati i possibili miglioramenti da considerare per un'eventuale evoluzione e ottimizzazione dell'implementazione, non realizzati finora a causa di limitazioni tecniche o di tempo.

- **Miglioramento della leggibilità, coerenza e modularità del codice:** Per motivi di spazio e di tempo non è stato possibile strutturare il codice in maniera più modulare. Una maggiore modularità faciliterebbe non solo la manutenzione, ma anche la scalabilità del sistema, rendendo più semplice l'aggiunta di nuovi contratti o funzionalità in futuro.
- **Personalizzazione degli sconti da parte dei singoli ristoranti:** Attualmente lo sconto applicato è uniforme, ma sarebbe utile permettere ai singoli ristoranti di definire autonomamente il proprio livello di sconto, aumentando così la flessibilità del sistema.
  **Delegabilità del voucher:** Implementare la possibilità di delegare l'utilizzo del voucher ad un'altra persona, migliorando l'esperienza utente e ampliando i casi d'uso.
- **Integrazione di un oracolo reale per la verifica delle partite IVA:** Attualmente, lo sviluppo è stato effettuato su Ganache, ambiente di test locale, che non permetteva l'utilizzo di un vero oracolo software.
- **Introduzione di un'interfaccia utente grafica:** Al momento la DApp non dispone di una UI. Lo sviluppo di un'interfaccia grafica intuitiva è essenziale per rendere la piattaforma accessibile anche agli utenti meno esperti, migliorandone significativamente l'usabilità.

## 4.3 Analisi delle prestazioni

In questa sezione viene presentata una bozza di analisi quantitativa delle prestazioni del sistema, con l'obiettivo di valutare la correttezza funzionale e l'efficienza dell'implementazione.

### 4.3.1 Casi di test analizzati

Piuttosto che analizzare l'intero insieme di casi d'uso possibili, per motivi di tempo ci si è concentrati sulle funzionalità principali, coprendo sia i flussi di successo che le principali casistiche di errore.

1. **Pagamento da parte di un utente verso un ristorante**
2. **Scrittura di una recensione**
3. **Like su di una recensione**
4. **Modifica della recensione**
5. **Cancellazione della recensione**

### 4.3.2 Metriche considerate

Per valutare le prestazioni complessive del sistema sviluppato, sono state considerate le seguenti metriche fondamentali, comunemente adottate nell'analisi di sistemi basati su smart contract:

- **Tempo di esecuzione (latenza):** misura il tempo necessario affinché una transazione venga confermata sulla blockchain. È stato osservato il tempo medio tra l'invio della transazione e la sua inclusione in un blocco, parametro utile a valutare la reattività percepita dal punto di vista dell'utente.

- **Costo computazionale (gas consumption):** ogni funzione contrattuale comporta un consumo di gas, che si traduce in un costo economico in fase di esecuzione,

### 4.3.3 Strumenti utilizzati

Per l'esecuzione dei test di performance è stato adottato **Truffle**, uno dei framework di sviluppo più consolidati per la realizzazione di smart contract su Ethereum. Sebbene non sia stato trattato direttamente nel corso, è parte integrante della **Truffle Suite**, la stessa raccolta di strumenti che include **Ganache**, con cui si integra nativamente. La documentazione ufficiale di Ganache fa riferimento a Truffle come uno degli strumenti consigliati per l'interazione (*Truffle Suite Ganache Overview.*, n.d.) e la gestione dei contratti smart in ambienti locali di test. Truffle fornisce un ambiente completo per la scrittura, compilazione, migrazione e testing dei contratti Solidity, e la sua stretta integrazione con Ganache consente una simulazione affidabile della blockchain locale per misurare tempi di esecuzione e consumo di gas in condizioni controllate.

### 4.3.4 Risultati

Al termine dell'esecuzione dei test funzionali, è stato generato un report contenente i dati relativi al **consumo di gas** e al **tempo di esecuzione** (in millisecondi) per ciascuna operazione effettuata. Queste metriche forniscono indicazioni utili **sull'efficienza** degli smart contract e aiutano a individuare eventuali aree di miglioramento, sia dal punto di vista **computazionale** che dell'**esperienza utente**.

| **Azione** | **Gas Usato** | **Tempo (ms)** |
|---|---|---|
| Pagamento da parte di un utente verso un ristorante | 82,750 | 102 |
| Scrittura di una recensione | 216,051 | 131 |
| Like su di una recensione | 165,505 | 155 |
| Modifica della recensione | 67,126 | 52 |
| Cancellazione della recensione | 46,935 | 54 |

*Table 3. Tabella riassuntiva performance del sistema implementato*

Come si può osservare, l'operazione più costosa in termini di gas è l'**aggiunta di una recensione**, seguita dalle interazioni combinate di pagamento e like. Le operazioni di modifica e cancellazione risultano invece molto più leggere, come previsto. Questo conferma che il carico computazionale è proporzionato alla complessità delle operazioni e che il sistema si comporta in modo coerente con quanto abbiamo progettato.

# APPENDICE: VULNERABILITY ASSESSMENT DEL CODICE

In questa sezione procederemo con una valutazione approfondita del codice sviluppato nel capitolo precedente, effettuando un'analisi di sicurezza focalizzata principalmente sugli aspetti legati alla programmazione e all'implementazione. L'obiettivo è identificare potenziali vulnerabilità, criticità o errori che potrebbero compromettere la robustezza e la sicurezza dell'applicazione, al fine di suggerire eventuali miglioramenti e best practice da adottare.

## Strumenti utilizzati e metodologie adottate

Per condurre l'analisi di sicurezza del codice, dopo aver raccolto tutta la documentazione tecnica necessaria, abbiamo impiegato lo strumento **Slither** per eseguire test automatici mirati all'individuazione dei principali punti critici.
Questi test sono stati integrati da una revisione manuale approfondita, che ha permesso di valutare con maggiore dettaglio aspetti specifici e contestuali non sempre rilevabili tramite l'analisi automatizzata. La combinazione di approcci automatici e manuali ha garantito un'analisi più completa e accurata delle potenziali vulnerabilità del codice.

## Risultati di Slither

Di seguito sono riportati i risultati ottenuti dall'analisi automatica effettuata con lo strumento **Slither**. Le diverse categorie evidenziano le tipologie di criticità rilevate, con il relativo numero di occorrenze e il livello di severità associato.

```
**THIS CHECKLIST IS NOT COMPLETE**. Use `--show-ignored-findings` to show all the results.
Summary
 - [solc-version](#solc-version) (6 results) (Informational)
 - [immutable-states](#immutable-states) (31 results) (Optimization)
 - [reentrancy-no-eth](#reentrancy-no-eth) (1 results) (Medium)
 - [shadowing-local](#shadowing-local) (2 results) (Low)
 - [events-access](#events-access) (6 results) (Low)
 - [missing-zero-check](#missing-zero-check) (13 results) (Low)
 - [reentrancy-benign](#reentrancy-benign) (1 results) (Low)
 - [reentrancy-events](#reentrancy-events) (3 results) (Low)
 - [timestamp](#timestamp) (2 results) (Low)
 - [naming-convention](#naming-convention) (66 results) (Informational)
 - [reentrancy-unlimited-gas](#reentrancy-unlimited-gas) (3 results) (Informational)
 - [too-many-digits](#too-many-digits) (2 results) (Informational)
 - [constable-states](#constable-states) (9 results) (Optimization)
```

| Categoria | N° | Livello di Severità | Descrizione |
|---|---|---|---|
| **solc-version** | 6 | Informational | Avvisi relativi alla versione del compilatore Solidity utilizzata, che potrebbero influire sulla sicurezza o compatibilità. |
| **immutable-states** | 31 | Optimization | Variabili dichiarate come non modificabili dopo l'inizializzazione, con suggerimenti per migliorare l'efficienza. |
| **reentrancy-no-eth** | 1 | Medium | Possibile vulnerabilità di reentrancy non legata a trasferimenti di Ether, da valutare con attenzione. |
| **shadowing-local** | 2 | Low | Variabili locali che sovrascrivono altre variabili con lo stesso nome, potenzialmente fonte di confusione o errori. |
| **events-access** | 6 | Low | Controlli su accesso o emissione di eventi non ottimali, con possibili implicazioni sulla tracciabilità delle operazioni. |
| **missing-zero-check** | 13 | Low | Mancanza di controlli per valori zero in alcune condizioni, che potrebbe causare comportamenti inattesi. |
| **reentrancy-benign** | 1 | Low | Situazioni di reentrancy considerate non pericolose ma comunque da monitorare. |
| **reentrancy-events** | 3 | Low | Potenziali problemi di reentrancy legati all'emissione di eventi, con basso livello di rischio. |
| **timestamp** | 2 | Low | Uso di timestamp per condizioni logiche, che può portare a vulnerabilità o comportamenti imprevisti a causa della manipolabilità. |
| **naming-convention** | 66 | Informational | Suggerimenti e avvisi riguardanti la coerenza e la correttezza delle convenzioni di denominazione adottate nel codice. |
| **reentrancy-unlimited-gas** | 3 | Informational | Avvisi su potenziali problemi di reentrancy dovuti a chiamate con gas illimitato. |
| **too-many-digits** | 2 | Informational | Segnalazioni relative all'uso di numeri con troppi decimali o cifre, che possono causare confusione o errori di calcolo. |

| | | | |
|---|---|---|---|
| **constable-states** | 9 | Optimization | Variabili che potrebbero essere dichiarate come costanti per migliorare l'efficienza e la chiarezza del codice. |

*Table 4. Tabella vulnerabilità trovate da Slither*

## Analisi e remediations

Dopo aver individuato le potenziali vulnerabilità nel codice, le abbiamo valutate singolarmente in base al loro impatto sul sistema. Per ciascuna vulnerabilità abbiamo individuato possibili remediation, applicandole solo nei casi in cui la vulnerabilità risultava effettivamente critica e l'intervento non comprometteva in modo significativo la struttura progettuale esistente.

| Categoria | Livello di Severità | Remediation |
|---|---|---|
| **solc-version** | Informational | Per quanto in questa demo non sia possibile, per risolvere questo tipo di vulnerabilità basta aggiornare la versione di solc utilizzata. |
| **immutable-states** | Optimization | Dichiarare immutabili le istanze dei contratti importati e le eventuali variabili owner. |
| **reentrancy-no-eth** | Medium | Effettuare il cambio di stato prima delle chiamate esterne è una buona pratica di sicurezza per prevenire vulnerabilità di tipo reentrancy.<br>Tuttavia, in questo caso specifico, non è stato possibile rispettare tale ordine: a causa di limitazioni di spazio nel contratto, la logica del ReviewManager è stata suddivisa in due contratti distinti.<br>La funzione coinvolta nella potenziale reentrancy effettua una verifica che, per coerenza funzionale, deve necessariamente essere eseguita prima del cambio di stato.<br>Di conseguenza, la struttura attuale non consente di anticipare l'aggiornamento dello stato rispetto alla chiamata esterna, senza alterare il comportamento previsto. |
| **shadowing-local** | Low | Può essere risolta evitando di riutilizzare le variabili. Essendo non particolarmente grave non si è provveduto ad applicare la remediation. |
| **events-access** | Low | La vulnerabilità segnalata può essere risolta mediante l'aggiunta di opportuni eventi nelle funzioni indicate.<br>Tuttavia, nel nostro caso specifico non è stato necessario applicare tale remediation, poiché la funzione interessata è una funzione interna adibita esclusivamente a compiti di utilità del sistema. |
| **missing-zero-check** | Low | Può essere risolta controllando che i parametri passati alle funzioni segnalate siano diversi dallo zero. |
| **reentrancy-benign** | Low | Può essere risolto allo stesso modo della reentrancy classica. Anche in questo caso non è stato possibile provvedere per lo stesso motivo esplicato in precedenza. |

| | | |
|---|---|---|
| **reentrancy-events** | Low | Effettuare il cambio di stato prima delle chiamate esterne è una buona pratica di sicurezza per prevenire questo tipo di vulnerabilità |
| **timestamp** | Low | Questa vulnerabilità non può essere risolta semplicemente lato software: se è possibile modificare l'hardware o l'ambiente di esecuzione, allora qualsiasi meccanismo di sicurezza può essere aggirato. Di conseguenza, l'unico modo per garantire realmente l'integrità in questi casi è impedire completamente modifiche non autorizzate, cosa attualmente non attuabile. |
| **naming-convention** | Informational | Può essere risolta modificando il codice in modo da seguire una convenzione stilistica dei nomi. |
| **reentrancy-unlimited-gas** | Informational | Effettuare il cambio di stato prima delle chiamate esterne è una buona pratica di sicurezza per prevenire questo tipo di vulnerabilità |
| **too-many-digits** | Informational | Il problema può essere mitigato esprimendo i numeri segnalati nella vulnerabilità in una forma più leggibile, riducendo il rischio di errori e potenziali overflow. In ogni caso, non si tratta di una vulnerabilità critica, poiché la versione di solc utilizzata include già controlli nativi contro gli overflow. |
| **constable-states** | Optimization | Può essere risolta dichiarando come costanti le variabili indicate dalla vulnerabilità. |

*Table 5. tabella riassuntiva delle remediation in relazione alle vulnerabilità evidenziate*

## Riferimenti


Clark, P. (2025, February 13). *As fake online reviews spread, so do consumers' concerns*. eMarketer. Retrieved April 28, 2025, from https://www.emarketer.com/content/fake-online-reviews-spread--do-consumers--concerns

Handy, J. (2012, Agosto 16). Think Yelp is Unbiased? Think Again!! *Forbes*. https://www.forbes.com/sites/jimhandy/2012/08/16/think-yelp-is-unbiased-think-again/

Parkin, S. (2018, Marzo 31). The Never-Ending War on Fake Reviews. *The New Yorker*. https://www.newyorker.com/tech/annals-of-technology/the-never-ending-war-on-fake-reviews

*The Shed at Dulwich*. (n.d.). Wikipedia. Retrieved April 28, 2025, from https://en.wikipedia.org/wiki/The_Shed_at_Dulwich

*Truffle Suite Ganache overview.* (n.d.). https://trufflesuite.com/ganache/


## *Indice delle figure*



## *Indice delle tabelle*