OBERON

Analisi dei Requisiti

A.A. 2021-2022

Componenti del gruppo:

Casazza Domenico, matr. 1201136

Casonato Matteo, matr. 1227270

Chen Xida, matr. 1217780

Pavin Nicola, matr. 1193215

Poloni Alessandro, matr. 1224444

Scudeler Letizia, matr. 1193546

Stojkovic Danilo, matr. 1222399

Indirizzo repository GitHub:

https://github.com/TeamOberon07/ShopChain

Università degli Studi di Padova

Indice

1 Registro delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
2.0.0	17/02/2022	Casazza Domenico	Responsabile	Approvazione del documento
1.1.3	10/02/2022	Poloni Alessandro	Verificatore	Controllo grammaticale e lessicale
1.1.2	22/01/2022	Pavin Nicola	Amministratore	Aggiunta e Modifica requisiti (§5)
1.1.1	12/01/2022	Casazza Domenico	Analista	Modifica (§3.3)
1.1.0	07/12/2021	Chen Xida e Pavin Nicola	Amministratore e Analista	Stesura requisiti (§5)
1.1.0	07/12/2021	Casazza Domenico e Stojkovic Danilo	Responsabile e Analista	Stesura casi d'uso (§4)
1.0.0	07/12/2021	Chen Xida	Responsabile	Approvazione del documento
0.2.0	05/12/2021	Chen Xida	Verificatore	Verifica completa del documento
0.1.0	03/12/2021	Chen Xida	Analista	Definiti alcuni termini
0.0.1	01/12/2021	Oberon	Analisti	Creazione bozza documento

2 Introduzione

2.1 Scopo documento

Questo documento si pone come obiettivo l'identificazione dei requisiti, sia quelli del proponente che quelli interni al team, al fine di una corretta realizzazione dell'applicazione.
Più precisamente, il documento conterrà il diagramma dei casi d'uso che riporteranno le informazioni degli attori del sistema e le interazioni con essa dove verranno descritti gli scenari principali e quelli alternativi.

2.2 Obiettivi del prodotto

Al giorno d'oggi, numerosi sono gli e-commerce che non hanno un sistema affinché l'acquirente e il venditore possano creare transizioni sicure. Difatti, l'acquirente può venire truffato dal venditore se dopo il pagamento non gli viene consegnato il prodotto o viceversa.

ShopChain è un applicativo in grado di affiancare un e-commerce nelle fasi di pagamento fino alla consegna usando la tecnologia delle blockchain. La blockchain è incaricata di ricevere l'ammontare speso dall'acquirente in criptovaluta, consegnandola al venditore, solo quando il pacco gli viene recapitato.

Nel momento della consegna del pacco l'acquirente dovrà necessariamente inquadrare il QR code applicato sul collo che certifica l'avvenuta consegna. Quindi verrà effettuato il passaggio della cryptovaluta dal wallet della piattaforma al wallet del venditore.

2.3 Vincoli Obbligatori

2.3.1 Espliciti

- Realizzazione dello/degli smart-contract nella blockchain per la gestione dei pagamenti agli e-commerce che si affidano a ShopChain.
  In particolare le fasi da coordinare saranno:
  - caricamento ordine;
  - gestione pagamento con qualsiasi cryptovaluta nella chain;
  - sblocco pagamento al venditore.
- Realizzazione piattaforma web per la gestione amministrativa dei pagamenti.
- Realizzazione della app per lo sblocco mediante QR code del pagamento al venditore nel momento della ricezione del pacco.
- Realizzazione di un database relazionale che conservi i dati e lo stato dei vari ordini.

2.3.2 Impliciti

Nello studio del problema i membri del gruppo hanno riscontrato i seguenti requisiti derivati.

- Ideazione di un sistema di conferma più robusto rispetto al semplice QR code in balia della buona volontà del compratore.

2.4 Vincoli Opzionali

Durante uno degli incontri con l'azienda il team ha proposto l'ampliamento delle funzionalità di questa parte della soluzione aggiungendo ulteriori servizi in-App per l'utente.

2.5 Riferimenti

- È stato creato il documento *glossario.pdf* per chiarire il significato dei termini tecnici che possono creare dubbi e perplessità.
- La pianificazione è divisa in Scrum, seguendo la metodologia agile. Le modalità e il modello di sviluppo sono riportate nel documento *NormeDiProgetto.pdf*

3 Tecnologie

3.1 Piattaforme di Esecuzione

Prodotto	Piattaforma di esecuzione
Applicazione web ShopChain	Chrome, Firefox, Safari e Opera
Applicazione mobile ShopChain	Android e iOS

3.2 Stack Front-end

Tecnologia	Descrizione
HTML	Linguaggio di markup per la struttura
CSS	Linguaggio di markup per lo stile
JavaScript	Linguaggio di programmazione lato client
React	Framework per lo sviluppo front-end
MetaMask	Plugin per la gestione dei wallet

3.3 Framework Front-end

Per lo sviluppo dell'applicazione web, il team ha considerato i tre principali framework: Angular, React e Vue. Le principali caratteristiche di questi framework, che servono a facilitare lo sviluppo lato client dell'applicazione, sono riportate nella seguente tabella:

Caratteristiche	Angular	React	Vue
Manipolazione UI/DOM	Sì	Sì	Sì
Gestione stati	Sì	Parziale	Sì
Routing	Sì	No	Sì
Gestione e validazione form	Sì	No	No
HTTP Client	Sì	No	No

Dopo attente ricerche su queste tecnologie e basandosi anche su quanto emerso dal documento *VerbaleEsterno_2021_11_29.pdf*, inizialmente il team Oberon ha deciso di utilizzare Vue che ha come punti di forza la semplicità e una curva di apprendimento molto meno ripida rispetto ad Angular e React, riuscendo comunque a soddisfare pienamente i requisiti tecnici necessari per la buona riuscita della webApp. Durante la prima implementazione del Proof of Concept, però, il gruppo si è accorto che tale framework non è il più adatto all'applicazione web da sviluppare. In seguito ad un incontro con il proponente, descritto nel documento *VerbaleEsterno_2022_01_12.pdf*, si è quindi deciso di passare al framework React, perché permette di gestire meglio la logica di visualizzazione dei componenti che formano le pagine web.
Per quanto riguarda la comunicazione tra webApp e blockchain si è deciso di utilizzare ethers.js a discapito di Avalanche.js in quanto la prima è una libreria più supportata, compatibile con la C-chain di Avalanche e supportata da una grande community, la seconda invece è una libreria più giovane e ancora non completa al 100%.

3.4 Stack Back-end

Tecnologia	Descrizione
Solidity	Linguaggio di programmazione per gli smart contract
Avalanche	Blockchain

Come si può notare anche dalla tabella, è assente un vero back-end dato che il progetto ha come focus primario la decentralizzazione dei dati.
Durante la discussione con il proponente (riportato nel *VerbaleEsterno_2022_02_16.pdf*) siamo giunti alla conclusione che non è necessario nè la costruzione di un backend nè la gestione di un database, infatti l'intero applicativo si poggia direttamente sulla blockchain, tramite l'uso di SmartContract.

3.5 BlockChain

L'analisi della blockchain da utilizzare è iniziata con lo studio delle tre categorie principali (Layer 0, 1, 2). Segue un elenco di pro e contro:

- Layer 0
  - Pro: interoperabilità tra le "sotto-blockchain": ad esempio, si possono effettuare chiamate a smart contract che risiedono in blockchain differenti, se collegate allo stesso Layer 0;
  - Contro: attualmente la tecnologia è in uno stato embrionale: ad esempio su Polkadot non è ancora collegata una "Parachain" (in gergo) adibita agli smart contract (c'è però su Kusama e Moonriver, la quale non è ancora molto utilizzata ma sta prendendo piede in questi mesi);
- Layer 1
  - EVM compatibili
    - Pro: le transazioni costano molto meno e sono molto più veloci rispetto ad Ethereum; per lo sviluppo e deployment degli smart contract si utilizzeranno le tecnologie più famose (Solidity, Metamask, Truffle...). Inoltre, queste blockchain hanno generalmente un numero di utenti maggiore rispetto alle altre, dato che questi trovano meno ostacoli arrivando da Ethereum;
    - Contro: l'unico fattore potrebbe riguardare la questione della decentralizzazione: Ethereum è la blockchain più decentralizzata, mentre quelle più recenti hanno un numero di nodi minori (per il semplice fatto che esistono da meno tempo);
  - Non EVM compatibili
    - Pro: In confronto ad Ethereum, le transazioni sono molto più veloci e molto meno costose (caratteristiche simili alle EVM compatibili).
    - Contro: Non supportano le stesse tecnologie di Ethereum per lo sviluppo di smart contract (Solidity, Metamask, Truffle) e quindi c'è il rischio che si trovi meno materiale di supporto online;
- Layer 2
  - Pro: molto più veloci di Ethereum e molto meno costosi;
  - Contro: al momento meno utilizzati dei Layer 1, community più piccole.

Abbiamo citato, oltre ad Ethereum, altri tre Layer 1: Avalanche, Binance Smart Chain e Fantom. Tutti e tre hanno caratteristiche simili: velocità, bassi costi, utenti numerosi.
Dal punto di vista del team, le differenze tra queste blockchain sono minime per i motivi sopra citati.

Dal punto di vista del proponente, se il progetto avesse la possibilità di avere un utilizzo "reale" allora ci sarebbe qualche piccola differenza: BSC è la blockchain più centralizzata ed è legata all'exchange Binance, per fare un esempio. Fantom ha circa 60 nodi validatori, mentre Avalanche ne ha più di 1000.

Inoltre, per quanto riguarda la community di sviluppatori, quella di Avalanche è quella

più ricca e disposta ad aiutare anche i progetti più piccoli. Per esempio, non sarebbe un problema mettersi in contatto con gli sviluppatori di Trader Joe (l'equivalente di Uniswap su Ethereum, exchange decentralizzato) in caso di eventuali difficoltà per la parte del progetto in cui dovremo effettuare la conversione in stablecoin.

Dopo aver analizzato tutte le possibilità offerte dall'universo Crypto, la blockchain scelta risulta essere Avalanche. Le sue caratteristiche (rete molto utilizzata, community di sviluppatori presente e attiva, velocità elevata, bassi costi, EVM compatibile, testnet robusta) la rendono la più adatta al progetto, anche nel caso quest'ultimo dovesse avere un utilizzo "reale".

3.6 Linguaggio per Smart Contract

Gli smart contract sono programmi salvati all'interno di una blockchain, eseguiti quando si verificano particolari precondizioni, quali rilascio di fondi tramite un pagamento, registrazione di un veicolo, invio di una notifica o ticket, ecc.
Vengono quindi tipicamente utilizzati per automatizzare l'esecuzione di un accordo, o contratto, appunto, senza l'intervento di terze parti, rendendo così tutti i partecipanti immediatamente certi del suo esito; oppure per automatizzare un workflow, scatenando la prossima azione da eseguire.

Per sviluppare smart contract sono disponibili vari linguaggi, in particolare i più famosi ed utilizzati sono i seguenti:

- Solidity
  - Pro: linguaggio di alto livello, object-oriented, tipizzazione statica, simile a C++ e Java, molte risorse e tutorial disponibili in rete, buon supporto da una vasta e consolidata community, disponibilità di vari tool al supporto dello sviluppo, quali Remix, Truffle e Hardhat;
  - Contro: più complesso scrivere programmi senza vulnerabilità rispetto agli altri linguaggi, possibilità di overflow;
- Vyper
  - Pro: linguaggio di alto livello, contract-oriented, tipizzazione forte, simile a Python, possibilità di calcolare un limite superiore preciso sul consumo di gas di ogni chiamata a funzione, più semplice scrivere smart contract più sicuri, disponibilità di vari tool al supporto dello sviluppo, quali Brownie ed Etherscan, controllo bound ed overflow su Array e String, limitandone però la dimensione al momento della dichiarazione;
  - Contro: meno supporto offerto dalla community in quanto il linguaggio è meno diffuso ed utilizzato, ancora in via di sviluppo;
- Yul/Yul+
  - Pro: permette agli sviluppatori di avvicinarsi di più alla EVM rispetto agli altri linguaggi, permette di migliorare il consumo di Gas;
  - Contro: meno risorse e supporto disponibili, linguaggio complicato, richiede una familiarità con la EVM. Sconsigliato come primo linguaggio per sviluppare smart contracts;

- Fe
  - Pro: linguaggio di alto livello, tipizzazione statica, ispirato a Python e Rust, utilizza lo stesso linguaggio intermedio di Solidity (Yul), limita il comportamento dinamico per garantire un costo Gas più preciso, presenza di libreria standard, punta ad essere semplice da imparare, anche per sviluppatori non familiari con la EVM;
  - Contro: linguaggio molto nuovo, ancora in stato di alpha, scarsità di supporto in quanto la community non è ancora diffusa, disponibile soltanto per Linux e MacOS.

Dopo aver visto e considerato tutti i punti critici sopra riportati, la scelta più sensata ricade tra Solidity e Vyper. Infatti Yul richiede una familiarità con la EVM che non è posseduta dai membri del Team, mentre Fe è ancora troppo nuovo. Inoltre, entrambi questi linguaggi sono scarsamente utilizzati e la possibilità di supporto, sia in rete che fornita dall'azienda, è quasi nulla.

Tra Solidity e Vyper invece la scelta definitiva presa dal gruppo, anche sotto consiglio diretto del proponente, risulta essere Solidity. Le sue caratteristiche, tra cui la sua maggiore maturità, diffusione, e semplicità, lo rendono il più adatto all'utilizzo per i nostri scopi.

3.7 App Mobile

Durante il primo incontro con il committente del progetto, in merito allo sviluppo dell'app mobile è stato consigliato di preferire l'utilizzo del framework Flutter rispetto a linguaggi di programmazione nativi. Questa linea guida è stata accolta conducendo una ricerca sugli aspetti positivi e negativi della tecnologia consigliata.

Flutter è un progetto open-source sviluppato da Google nel 2018 al fine di creare interfacce native per iOS e Android, usando Dart come unico linguaggio di programmazione.

- Pro:
  - un unico linguaggio di programmazione e un unico codice sorgente per app iOS e Android;
  - il team di sviluppo può dedicare le sue forze allo sviluppo di un unico prodotto per entrambi i sistemi operativi ottimizzando i tempi;
  - permette di raggiungere performance molto vicine a quelle di un app nativa;
  - tutti gli elementi dell'applicazione sono widget, è possibile crearne di complessi combinando insieme widget più semplici. Questo dà molte opportunità allo sviluppatore che può esprimere molta creatività;
  - sono presenti librerie molto ricche per l'interfaccia utente;
  - fase di sviluppo molto rapida con la funzionalità "hot reload" che non richiede di ricompilare il codice;
- Contro:
  - i molteplici widget rendono il codice molto nidificato;

- Dart è un linguaggio di programmazione nuovo e poco diffuso.

Sulla base di questi elementi è chiaro che i fattori positivi superino nettamente quelli negativi, perciò utilizzare Flutter risulta l'opzione migliore.

4 Casi d'uso

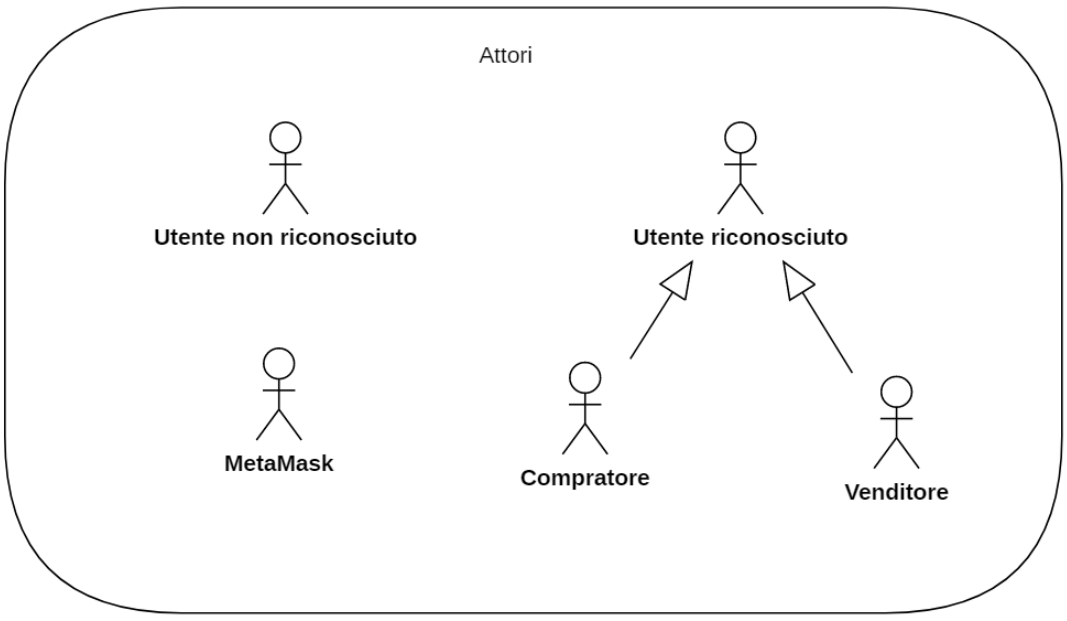
4.1 Scopo

In questa sezione sono state descritte tutte le funzionalità che offre il sistema e gli scenari in cui si possono ritrovare gli attori. Le funzionalità e le relazioni tra gli attori e il sistema sono stati rappresentati attraverso i diagrammi dei casi d'uso.

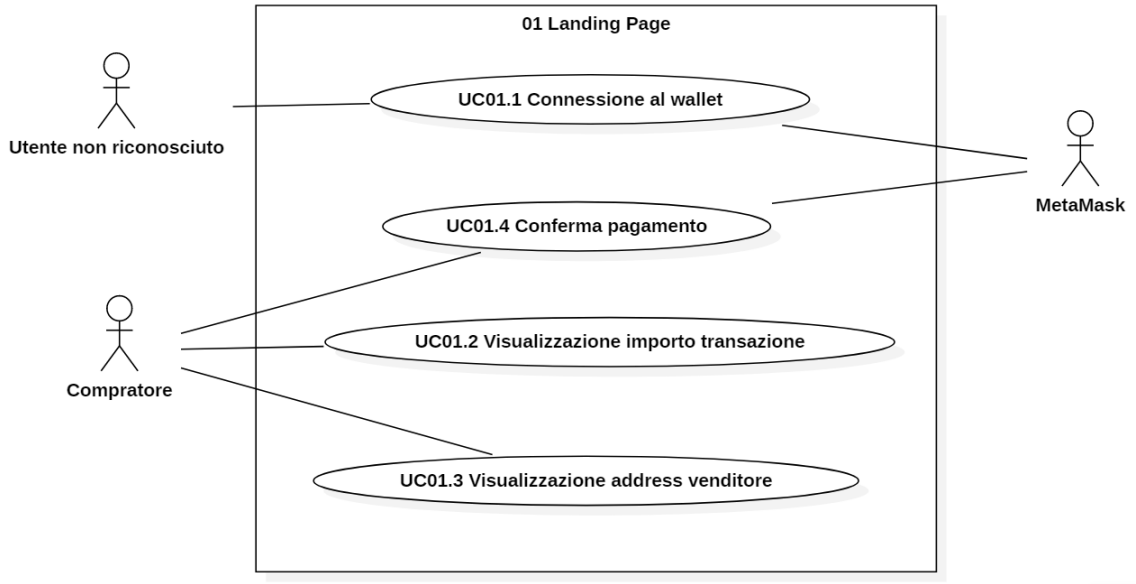
4.2 Attori

Sono stati individuati gli attori principali che interagiscono con il sistema e sono stati descritti brevemente le principali funzionalità a cui possono accedere:

- Utente non riconosciuto: può accedere al proprio wallet;
- Utente riconosciuto: può accedere alle funzionalità di Compratore o Venditore;
- MetaMask: è il plugin esterno al sistema che gestisce i wallet dei compratori e venditori.
- Compratore: può creare ordini, pagare il Venditore o effettuare resi;
- Venditore: può gestire le transazioni o annullarne alcune.



4.3 01 Landing Page



UC01.1 Connessione al wallet

Titolo: UC01.1 Connessione al wallet

Attore primario: Utente non riconosciuto

Attore secondario: MetaMask

Precondizioni: Il Compratore non ha effettuato il collegamento al proprio wallet.

Postcondizioni: Il Compratore si è identificato e può proseguire con il pagamento.

Scenario principale: Il Compratore vuole interagire con lo smart contract di ShopChain, quindi deve collegare il wallet ad esso per poter autorizzare successivamente le transazioni.

1. Il Compratore clicca su “Connetti wallet”;
2. Si apre il pop-up di MetaMask dove egli inserisce i dati;
3. L’utente dà il permesso allo smart contract di interagire con il proprio wallet.

UC01.2 Visualizzazione importo transazione

Titolo: UC01.2 Visualizzazione importo transazione

Attore primario: Compratore

Precondizioni: Il Compratore ha iniziato la fase di pagamento nella piattaforma e-commerce e ha eseguito l'accesso al wallet.

Postcondizioni: Il Compratore ha visualizzato l'importo della transazione.

Scenario principale: Il Compratore visualizza nella Landing Page l'importo corrispondente al/i prodotto/i selezionato/i, con eventualmente sommato il costo di consegna e le fee stimate (medie).

UC01.3 Visualizzazione address Venditore

Titolo: UC01.3 Visualizzazione address Venditore

Attore primario: Compratore

Precondizioni: Il Compratore ha iniziato la fase di pagamento nella piattaforma e-commerce.

Postcondizioni: Il Compratore ha visualizzato correttamente l'address del Venditore.

Scenario principale: Il Compratore visualizza nella Landing Page l'address del Venditore nella blockchain Avalanche, in formato esadecimale (esempio: 0x7949635E2877ef8ca37B8526507AC214B0423Ebf).

UC01.4 Conferma Pagamento

Titolo: UC01.4 Conferma Pagamento

Attore primario: Compratore

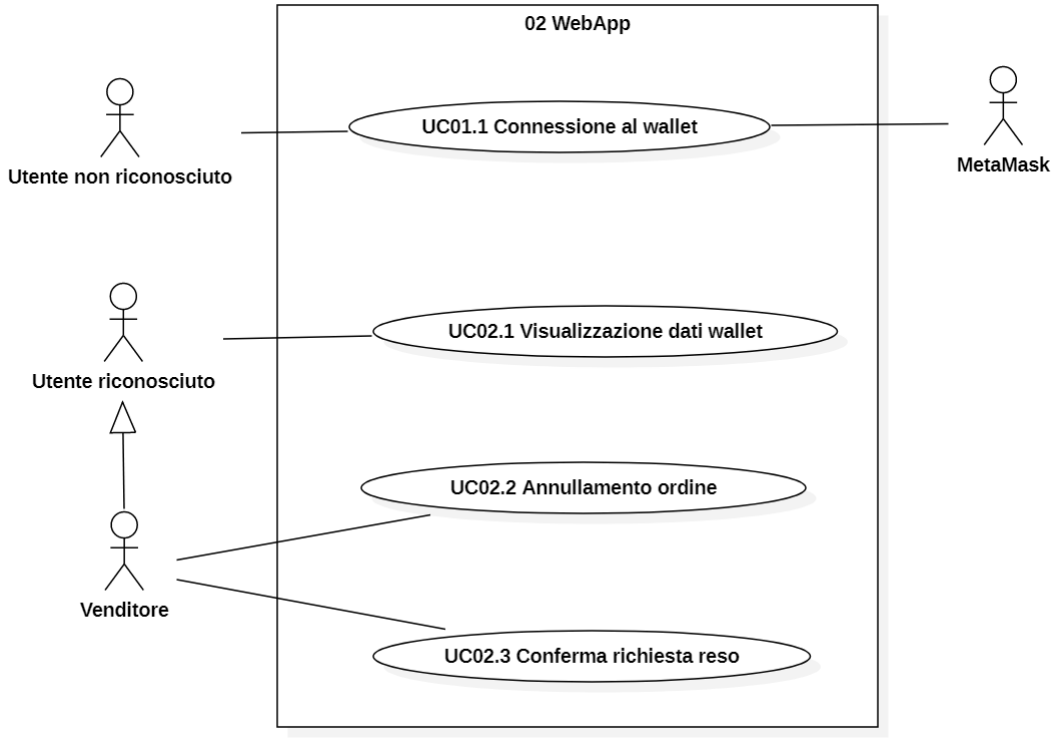
Attore secondario: MetaMask

Precondizioni: Il Compratore è stato riconosciuto e ha visualizzato l'importo dovuto.

Postcondizioni: La transazione è stata avviata e le cryptovalute si trovano nello smart contract.

Scenario principale: Il Compratore autorizza la transazione iniziata e causa il trasferimento dell'importo dal suo wallet allo smart contract.

4.4 02 WebApp



UC02.1 Visualizzazione dati wallet

Titolo: UC02.1 Visualizzazione dati wallet.

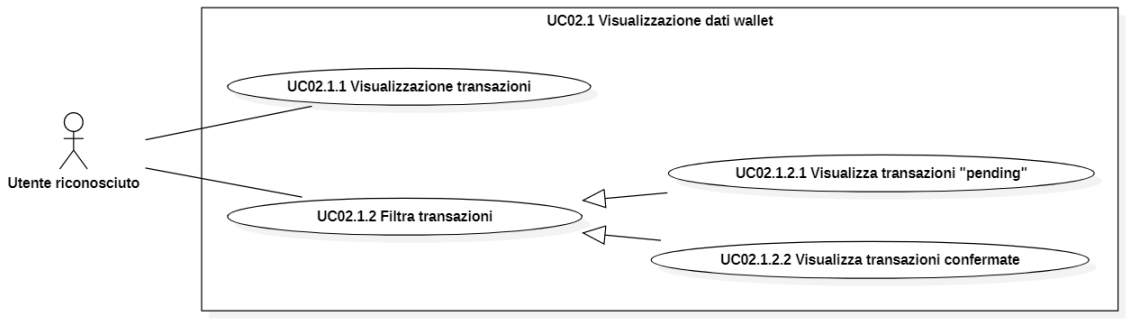
Attore primario: Utente riconosciuto.

Precondizioni: L'utente è stato riconosciuto tramite la connessione al suo wallet.

Postcondizioni: L'utente ha visualizzato i dati relativi al wallet collegato.

Scenario principale: L'utente può:

1. visualizzare tutte le transazioni relative al wallet con cui è connesso (UC02.2.1);
2. filtrare le transazioni (UC02.2.2) per visualizzare le transazioni "pending" (UC02.2.2.1), cioè non confermate, e le transazioni confermate (UC02.2.2.2).



UC02.1.1 Visualizzazione transazioni

Titolo: UC02.1.1 Visualizzazione transazioni.

Attore primario: Utente riconosciuto.

Precondizioni: L'utente è stato riconosciuto tramite la connessione al suo wallet.

Postcondizioni: L'utente ha visualizzato le transazioni relative al wallet collegato.

Scenario principale: L'utente accede alla pagina di visualizzazione di tutte le transazioni relative al wallet con cui è collegato.

UC02.1.2 Filtra transazioni

Titolo: UC02.1.2 Filtra transazioni.

Attore primario: Utente riconosciuto.

Precondizioni: L'utente riconosciuto visualizza tutte le transazioni del suo wallet.

Postcondizioni: L'utente riconosciuto visualizza le transazioni filtrate.

Scenario principale: L'utente riconosciuto può decidere se:

1. Filtrare e visualizzare le transazioni "pending" (UC02.2.2.1);
2. Filtrare e visualizzare le transazioni "confermate" (UC02.2.2.2).

UC02.1.2.1 Visualizzazione transazioni "pending"

Titolo: UC02.1.2.1 Visualizzazione transazioni "pending".

Attore primario: Utente riconosciuto.

Precondizioni: L'utente riconosciuto ha scelto di visualizzare le transazioni "pending".

Postcondizioni: L'utente riconosciuto visualizza solo le transazioni in stato "pending".

Scenario principale: L'utente riconosciuto vuole visualizzare solo le transazioni in stato "pending" e clicca quindi il pulsante che ha la funzionalità di filtrare le transazioni che hanno questo stato.

UC02.1.2.2 Visualizzazione transazioni “confermate”

Titolo: UC02.1.2.2 Visualizzazione transazioni “confermate”.

Attore primario: Utente riconosciuto.

Precondizioni: L’utente riconosciuto ha scelto di visualizzare le transazioni “confermate”.

Postcondizioni: L’utente riconosciuto visualizza solo le transazioni in stato “confermato”.

Scenario principale: L’utente riconosciuto vuole visualizzare solo le transazioni in stato “confermato” e clicca quindi il pulsante che ha la funzionalità di filtrare le transazioni che hanno questo stato.

UC02.2 Annullamento ordine

Titolo: UC02.2 Annullamento ordine.

Attore primario: Venditore.

Precondizioni: Il Venditore desidera annullare uno degli acquisti sul proprio e-commerce.

Postcondizioni: I soldi precedentemente depositati dal Compratore nello smart contract vengono restituiti al Compratore.

Scenario principale:

1. Il Venditore visualizza le transazioni attive e individua quella da annullare;
2. Clicca sul bottone annulla e conferma la scelta;
3. La WebApp attiva un metodo dello smart contract che procederà a rendere il deposito al wallet del Compratore;
4. Il Compratore viene notificato via App del fatto che la transazione è stata vanificata e la merce non arriverà.

UC02.3 Conferma richiesta reso

Titolo: UC02.3 Conferma richiesta reso.

Attore primario: Venditore.

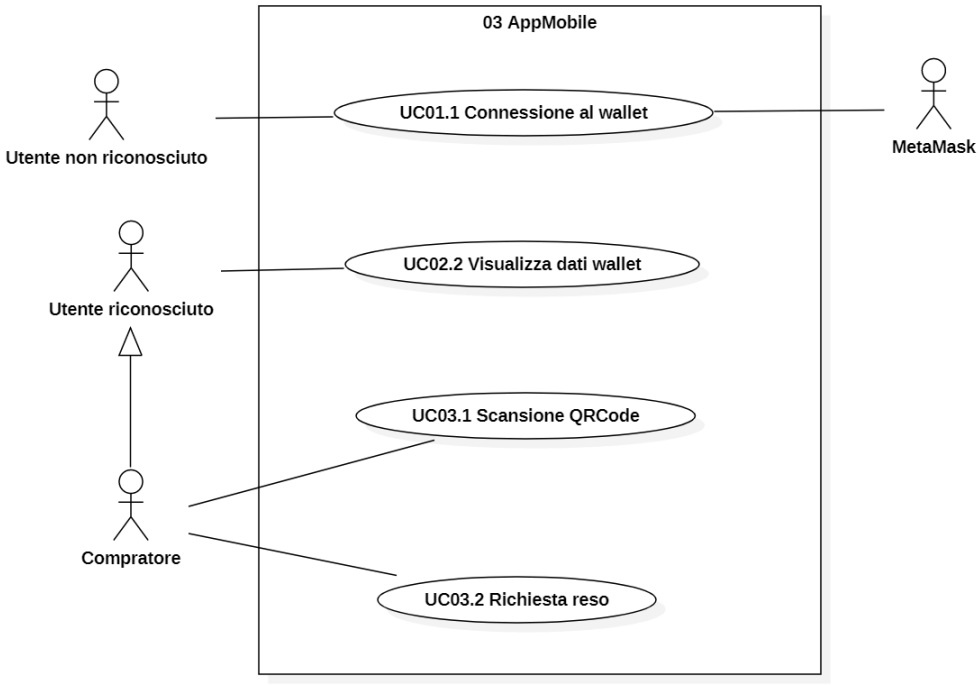
Precondizioni: Il Venditore desidera confermare la richiesta di reso effettuata da un Compratore.

Postcondizioni: Il Venditore restituisce l’importo corrispondente all’ordine di cui è stato richiesto il reso al Compratore.

Scenario principale:

1. Il Venditore visualizza le transazioni che corrispondono agli ordini di cui è stato chiesto il reso;
2. Il Venditore conferma le richieste di reso che ritiene più opportune.

4.5 03 App Mobile



UC03.1 Scansione QR code

Titolo: UC03.1 Scansione QR code

Attore primario: Compratore

Precondizioni: L'utente si è già identificato per mezzo del wallet.

Postcondizioni: Il Compratore ha sbloccato le cryptovalute depositate nello smart contract.

Scenario principale:

1. Il pacco con QR code raggiunge l'abitazione del Compratore;
2. L'utente accede all'app e seleziona "conferma transazione";
3. Dopo aver aperto la fotocamera inquadra il codice ed attende che l'app lo identifichi;
4. Con le informazioni contenute l'App ritrova la transazione attiva e invoca un metodo dello smart contract che procede all'inoltro delle cryptovalute nel wallet del Venditore;
5. La transazione "pending" diventa "completata" nella web app del Venditore.

UC03.2 Richiesta reso

Titolo: UC03.2 Richiesta reso

Attore primario: Compratore

Precondizioni: Il Compratore è già connesso al wallet ed ha effettuato acquisti su ShopChain in passato.

Postcondizioni: La richiesta di reso è stata inoltrata al Venditore.

Scenario principale: Il Compratore visualizza le transazioni effettuate in passato. Da qui può cliccare su un pulsante apposito tramite il quale potrà inizializzare la procedura di reso.

5 Requisiti

5.1 Introduzione

Di seguito vengono riportati, in forma tabellare, i requisiti del progetto, classificati e identificati come specificato seguentemente.

- **Codice**: I requisiti sono identificati da un codice identificativo formato con la seguente regola:

  R[Tipologia][Codice]

  - Con tipologia si indica la natura del requisito, tramite una lettera:
    - **F:** Funzionale
    - **Q:** Qualitativo
    - **V:** Vincolo
    - **P:** Prestazionale
  - Con codice si intende un identificativo univoco espresso in forma gerarchica padre.figlio.

- **Classificazione:** Con classificazione si indica l'importanza del requisito, riconducibile a tre categorie:
  - Obbligatorio
  - Desiderevole
  - Opzionale
- **Descrizione:** Descrizione sintetica del requisito
- **Fonti:** fonti d'origine del requisito

5.2 Requisiti Funzionali

Alcune funzionalità sono state riportate più volte perchè nonostante siano le stesse appartengono a contesti diversi e necessitano comunque di test di unità separati (ad esempio, la connessione al wallet di Metamask avviene sia tramite landing page che app mobile).

Codice	Classificazione	Descrizione	Fonti
RF1	Obbligatorio	Landing page che permetta di completare il pagamento	UC01
RF1.1	Obbligatorio	L'utente deve potersi connettere al proprio wallet tramite Metamask	UC01.1
RF1.2	Obbligatorio	L'utente deve poter visualizzare l'importo totale della transazione	UC01.2
RF1.3	Obbligatorio	L'utente deve poter visualizzare l'address del venditore	UC01.3
RF1.4	Obbligatorio	L'utente deve poter autorizzare la transazione, completando l'acquisto del prodotto	UC01.4

Codice	Classificazione	Descrizione	Fonti
RF1.4.1	Obbligatorio	Il sistema deve convertire l'importo pagato in stablecoin	Verbale 14/12/21
RF1.4.2	Obbligatorio	Il sistema deve poter richiedere il caricamento sulla blockchain dello smart contract, contenente i dati della transazione e trattenendo l'importo destinato al venditore	Capitolato, sezione 1.2
RF1.4.3	Obbligatorio	Il sistema deve poter creare un QR code relativo alla transazione, il quale successivamente permetterà di sbloccare i soldi	Capitolato, sezione 1.2
RF2	Obbligatorio	WebApp che permetta di visualizzare ed interagire con i propri ordini	UC02
RF2.1	Obbligatorio	L'utente deve potersi connettere al proprio wallet tramite Metamask	UC01.1
RF2.2	Obbligatorio	L'utente deve poter visualizzare tutte le transazioni effettuate con ShopChain dai suoi indirizzi wallet collegati	UC02.1, UC02.1.1
RF2.2.1	Desiderevole	L'utente deve poter filtrare le transazioni in base al loro stato	UC02.1.2
RF2.2.1.1	Desiderevole	L'utente deve poter visualizzare soltanto le transazioni filtrate	UC02.1.2.1, UC02.1.2.2
RF2.3	Desiderevole	L'utente deve poter annullare un qualsiasi ordine attivo in cui egli risulti essere un venditore	UC02.2
RF2.3.1	Desiderevole	Il sistema deve poter sbloccare i soldi e restituirli al compratore	UC02.2
RF2.4	Desiderevole	L'utente venditore deve poter confermare una richiesta di reso	UC02.3
RF3	Obbligatorio	App mobile che permetta di visualizzare ed interagire con i propri ordini, e di completare le transazioni sbloccando i soldi	UC03
RF3.1	Obbligatorio	L'utente deve potersi connettere al proprio wallet tramite Metamask	UC01.1
RF3.1.1	Obbligatorio	L'app mobile di ShopChain deve poter interagire con l'app mobile di Metamask	UC01.1
RF3.2	Obbligatorio	L'utente deve poter visualizzare tutte le transazioni effettuate con ShopChain dai suoi indirizzi wallet collegati	UC02.1, UC02.1.1
RF3.2.1	Desiderevole	L'utente deve poter filtrare le transazioni in base al loro stato	UC02.1.2
RF3.2.1.1	Desiderevole	L'utente deve poter visualizzare soltanto le transazioni filtrate	UC02.1.2.1, UC02.1.2.2
RF3.3	Obbligatorio	L'utente compratore deve poter completare l'acquisto tramite scansione del QR code alla ricevuta del pacco	UC03.1
RF3.3.1	Obbligatorio	L'app mobile deve poter accedere alla fotocamera	UC03.1
RF3.3.2	Obbligatorio	L'app mobile deve poter leggere correttamente il QR Code	UC03.1

Codice	Classificazione	Descrizione	Fonti
RF3.3.3	Obbligatorio	L'app mobile deve poter richiedere di sbloccare i soldi legati alla transazione, inoltrandoli nel wallet del venditore	UC03.1
RF3.4	Desiderevole	L'utente deve poter richiedere il reso di un prodotto acquisitato	UC03.2

5.3 Requisiti di prestazionali

In questa fase del progetto non sono stati individuati requisiti prestazionali cruciali in modo quantitativo, tuttavia il team ha individuato le seguenti possibile problematiche:

- numero di accessi all'app;
- numero di accessi alla webApp;

Si nota che la maggior parte delle operazioni complesse è inoltrata alla blockchain e a Metamask. Inoltre il database è assente (leggere sezione 5.6), quindi anche i suoi requisiti prestazionali. Il capitolato non ha definito esplicitamente tali requisiti e anche la dimensione dell'audience potenziale è indefinita, perciò il team non ritiene opportuno fare ulteriori considerazioni, se non seguire le buone prassi durante la codifica, per un codice efficiente e un'architettura che abbia un grado medio di scalabilità ("architettura a servizi").

5.4 Requisiti di Qualità

Codice	Classificazione	Descrizione	Fonti
RQ1	Obbligatorio	Devono essere presenti test che dimostrino il corretto funzionamento dei servizi e delle funzionalità previste	Capitolato
RQ1.1	Obbligatorio	Tutte le componenti applicative devono essere correlate da test unitari e d'integrazione	Capitolato
RQ1.2	Obbligatorio	Il sistema deve essere testato nella sua interezza tramite test end-to-end	Capitolato
RQ1.3	Obbligatorio	I test devono avere una copertura minima dell' 80%	Capitolato
RQ1.4	Obbligatorio	I test devono essere correlati da report	Capitolato
RQ2	Obbligatorio	Devono essere redatti specifici documenti	Capitolato
RQ2.1	Obbligatorio	Documento che motivi e relazioni le scelte implementative e progettuali prese	Capitolato
RQ2.1	Obbligatorio	Documento che esponga i problemi aperti e le eventuali soluzioni da esplorare	Capitolato
RQ3	Obbligatorio	Devono essere rispettate le metriche individuate descritte nel documento "Piano di Progetto"	Documento: Piano di Progetto

5.5 Vincoli di progetto

Codice	Classificazione	Descrizione	Fonti
RV1	Obbligatorio	Realizzazione degli smart contract nella blockchain per la gestione dei pagamenti agli e-commerce che si affidano a ShopChain	Capitolato
RV2	Obbligatorio	Realizzazione piattaforma web per la gestione amministrativa dei pagamenti	Capitolato
RV3	Obbligatorio	Realizzazione della app per lo sblocco mediante QR code del pagamento al venditore nel momento della ricezione del pacco	Capitolato
RV4	Obbligatorio	Durante tutte le fasi di processo è fondamentale che i passaggi più critici ai fini della sicurezza (pagamento e sblocco del pagamento) vengano realizzati mettendo in comunicazione diretta la blockchain con le altre componenti applicative (app mobile/webApp).	Capitolato
RV5	Opzionale	Ideazione di un sistema di conferma più robusto rispetto al semplice QR code in balia della buona volontà del compratore	Capitolato
RV6	Opzionale	Durante uno degli incontri con l'azienda il team ha proposto l'ampliamento delle funzionalità di questa parte della soluzione aggiungendo ulteriori servizi in-App per l'utente	Capitolato
RV7	Obbligatorio	ShopChain dovrà utilizzare React come framework per lo sviluppo del front-end	Verbale 08/1/2022
RV8	Obbligatorio	ShopChain dovrà poter interagire con il plugin Metamask	Verbale 29/11/21
RV9	Obbligatorio	ShopChain dovrà utilizzare Java ed il framework Spring per lo sviluppo del back-end	Capitolato
RV10	Obbligatorio	ShopChain dovrà utilizzare Solidity per lo sviluppo degli Smart Contract	Verbale 29/11/21
RV11	Obbligatorio	ShopChain dovrà utilizzare blockchain Avalanche	Verbale 29/11/21
RV12	Obbligatorio	ShopChain dovrà utilizzare il framework web3.js per interagire con la blockchain	Verbale 29/11/21
RV13	Obbligatorio	ShopChain dovrà utilizzare Flutter per lo sviluppo dell'app mobile	Verbale 29/11/21

5.6 Considerazioni con il proponente

Si riporta la lista di tutte le considerazioni che hanno ridefinito alcuni requisiti, raggiunte a seguito dei meeting con il proponente e descritte nei verbali esterni.

- Il database per l'amministrazione delle transazioni non è più un requisito dato che la blockchain ha i dati necessari e sufficienti per il prodotto, inoltre realizzare un database andrebbe contro la filosofia della tecnologia delle blockchain (anonimità).