OBERON

Piano di Qualifica v. 1.0.5

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 **2**

2 Introduzione **3**
2.1 Scopo del documento . . . 3
2.2 Obiettivi del prodotto . . . 3
2.3 Riferimenti . . . 3

3 Qualità di processo **4**
3.1 Processi primari . . . 4
3.1.1 Fornitura . . . 4
3.1.2 Sviluppo . . . 4
3.2 Processi di supporto . . . 4
3.2.1 Verifica . . . 4
3.3 Processi organizzativi . . . 4

4 Qualità di prodotto **5**
4.1 Obiettivi . . . 5
4.2 Metriche . . . 6
4.2.1 Budget Individuale per Sprint . . . 9
4.2.2 Errori ortografici . . . 9
4.2.3 Indice Gulpease . . . 10
4.2.4 SLOC/metodo . . . 10
4.2.5 Comprensibilità del codice . . . 11
4.2.6 Complessità ciclomatica . . . 11
4.2.7 Copertura dei requisiti . . . 12
4.2.8 Varianza rispetto al preventivo . . . 12
4.2.9 Errori per linee di codice . . . 13
4.2.10 Numero di click . . . 13
4.2.11 Tempo di accesso ad una funzionalità . . . 14
4.2.12 Uniformità al cambio di Sistema Operativo . . . 14
4.2.13 Uniformità al cambio di Browser . . . 15

5 Specifica test **16**
5.1 Test d'unità . . . 16
5.1.1 Test d'unità - WebApp . . . 16
5.1.2 Tracciamento Test d'unità - WebApp . . . 18
5.1.3 Test d'unità - Smart Contract . . . 22
5.1.4 Tracciamento Test d'unità - Smart Contract . . . 24
5.2 Test d'integrazione . . . 24
5.3 Test di sistema . . . 25
5.4 Test di accettazione . . . 25
5.5 Test di regressione . . . 25

1 Registro delle modifiche

v	Data	Nominativo	Ruolo	Descrizione
1.0.5	23/05/2022	Pavin Nicola	Progettista	Modifica e completamento descrizione e tracciamento test d'unità §(5.1.1) e §(5.1.2)
1.0.4	22/05/2022	Chen Xida	Verificatore	Aggiunta grafici sezione metriche §(4.2)
1.0.3	22/05/2022	Casonato Matteo	Programmatore	Descrizione e tracciamento test d'unità §(5.1.3) e §(5.1.4)
1.0.2	21/05/2022	Casazza Domenico	Amministratore	Descrizione e tracciamento test d'unità §(5.1.1) e §(5.1.2)
1.0.1	21/05/2022	Scudeler Letizia	Progettista	Modifiche sezione §(4)
1.0.0	17/02/2022	Casazza Domenico	Responsabile	Approvazione del documento
0.2.0	12/02/2022	Poloni Alessandro	Verificatore	Controllo grammaticale e lessicale
0.1.2	01/02/2022	Chen Xida, Stojkovic Danilo	Analista, Programmatore	Aggiunta nuove metriche §(4.2), alcuni grafici §(4.2.1-2), riferimenti e verifica
0.1.1	22/01/2022	Chen Xida	Analista	Stesura iniziale sezione test §(5) e verifica documento versione 0.0.3
0.0.3	21/01/2022	Poloni Alessandro	Analista	Aggiunta metriche §(4.2) qualità di prodotto e verifica
0.0.2	09/01/2022	Chen Xida	Amministratore	Stesura descrizione e obiettivi delle sezioni e verifica
0.0.1	03/01/2022	Chen Xida	Amministratore	Creazione scheletro documento

2 Introduzione

2.1 Scopo del documento

Questo documento fornisce le informazioni sulle metriche individuate dal team per il miglioramento e la manutenzione del software e dei processi che concorrono all'avanzamento del progetto.

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 transazioni 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 ne certifica l'avvenuta consegna. Quindi verrà effettuato il passaggio della criptovaluta dal wallet della piattaforma al wallet del venditore.

2.3 Riferimenti

- È stato creato il documento *Glossario_1.0.0.pdf* per chiarire il significato dei termini tecnici che possono creare dubbi e perplessità.
- La pianificazione è divisa in sprint, seguendo la metodologia agile. Le modalità e il modello di sviluppo sono riportate nel documento *NormeDiProgetto_1.0.0.pdf*
- Indice ISO/IEC 12207: `https://it.wikipedia.org/wiki/ISO_12207`
- Standard ISO/IEC 9126: `https://en.wikipedia.org/wiki/ISO/IEC_9126`
- Indice Gulpease: `https://it.wikipedia.org/wiki/Indice_Gulpease`
- Average Cyclomatic Complexity: `https://eslint.org/docs/rules/complexity`

3 Qualità di processo

Questo progetto ha come standard di riferimento ISO/IEC 12207 per garantire la qualità di processo.
Il team ha deciso di adottare alcuni dei processi individuati dallo standard in base alle necessità del progetto.

3.1 Processi primari

Per processi primari si intende l'insieme di attività attive che concorrono all'avanzamento del progetto stesso.
In particolare nel nostro caso i processi primari sono la fornitura e lo sviluppo.

3.1.1 Fornitura

La fornitura è formata dall'insieme di attività propedeutiche allo sviluppo, come ad'esempio la gestione dei rapporti con il cliente o la scelta sull'allocazione delle risorse temporali ed economiche del progetto.

3.1.2 Sviluppo

Per sviluppo si intende l'insieme delle attività necessarie per la realizzazione del software richiesto, a partire dall'individuazione dei problemi e fino alla concretizzazione delle soluzioni proposte.

3.2 Processi di supporto

I processi di supporto sono necessari per il controllo ed il monitoraggio delle attività che contribuscono all'avanzamento del progetto e ha come scopo garantire la qualità sia di queste attività che del prodotto finale realizzato, in termini di efficienza ed efficacia.

3.2.1 Verifica

La verifica è un processo per il controllo del codice scritto in modo tale da correggere tempestivamente errori che potrebbero diventare costosi durante il ciclo di vita del software, soprattutto in fase di uso e manutenzione.

3.3 Processi organizzativi

Per processi organizzativi si intendono le attività trasversali al progetto, che creano struttura nelle procedure e nel team affinchè l'approccio sia sistematico, disciplinato e quantificabile durante le fasi del progetto. I software usati per il coordinamento del team e per la gestione del progetto sono più dettagliamente descritti nelle norme di progetto.

4 Qualità di prodotto

Il gruppo ha individuato i seguenti obiettivi e metriche per garantire la qualità del prodotto, facendo riferimento allo standard ISO/IEC 9126.

4.1 Obiettivi

Obiettivo	Descrizione	Metriche
Chiarezza	I documenti sono stati sottoposti a una verifica di tipo ortografica e grammaticale in modo da massimizzarne la chiarezza e la comprensione	M1, M2
Funzionalità	Il software soddisfa tutti i requisiti individuati dall'analisi dei requisiti, in modo accurato e sicuro	M7
Efficienza	Il software usa la quantità minima, quindi ottima, di risorse e tempi necessari per svolgere una funzionalità	M8
Usabilità	L'uso del software è intuitivo, chiaro e piacevole sotto il punto di vista sia visivo che interattivo	M10, M11
Affidabilità	Il software è robusto nella gestione di eccezioni, avendo una alta tolleranza ai guasti e usando sistemi di controllo a monte	M9
Portabilità	Il software è eseguibile indipendentemente dall'ambiente di esecuzione, mantenendo le funzionalità originali senza effetti collaterali	M12, M13
Manutenibilità	Il software è un prodotto divisibile, quindi sostituibile nelle sue parti, permettendo modifiche di quest'ultime a costi contenuti e nel modo più agevole possibile	M3, M4, M5, M6

4.2 Metriche

Lo scopo della seguente sezione è descrivere le metriche adottate dal *Team Oberon* per misurare la qualità del proprio prodotto.

M1 Correttezza grammaticale

La correttezza grammaticale deve essere garantita dai controlli del Verificatore ad ogni ispezione.

M2 Indice di Gulpease

Indice che riporta il grado di leggibilità di un testo redatto in lingua italiana. La formula adottata è la seguente:

$$GUL = 89 + \frac{300*(totfrasi)-10*(totlettere)}{(totparole)}$$

M3 Numero di SLOC

Source lines of code (SLOC) è una metrica software che misura le dimensioni di un software basandosi sul numero di linee di codice sorgente.

M4 Densità dei commenti

Questa metrica misura la densità dei commenti all'interno del codice sorgente prodotto dal team di sviluppo.

M5 Comprensibilità del codice

Questa metrica misura la comprensibilità del codice ad una persona che non ha scritto quel codice, fa riferimento ai nomi autoesplicativi delle variabili, delle classi e dei metodi.

M6 Complessità ciclomatica

Questa metrica è utilizzata per misurare la complessità di un programma. Misura direttamente il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso.

M7 Copertura dei requisiti

Indice che misura in ogni istante la percentuale di requisiti obbligatori soddisfatti. La formula adottata è la seguente:

$$CRO = \frac{ROC}{RO} * 100$$

dove:

- **ROC** indica il numero di requisiti obbligatori coperti dall'implementazione;
- **RO** indica il numero totale dei requisiti obbligatori.

M8 Varianza rispetto al preventivo

Questa metrica misura la varianza del prezzo del consuntivo finale rispetto al preventivo iniziale.

M9 Errori per linee di codice

Questa metrica indica la correttezza del codice prodotto dal team di sviluppo ed è data dal numero di errori diviso il numero di righe di codice totale.

M10 Numero di click

Questa metrica misura il numero di click (o tocchi) necessari per attivare una determinata funzionalità dell'applicativo prodotto dal *Team Oberon*.

M11 Tempo di accesso ad una funzionalità

Questa metrica misura il tempo necessario per accedere ad una determinata funzionalità dell'applicativo prodotto dal *Team Oberon*.

M12 Uniformità al cambio del browser

Questa metrica indica l'uniformità della webApp al variare del browser e della versione su cui viene eseguita.

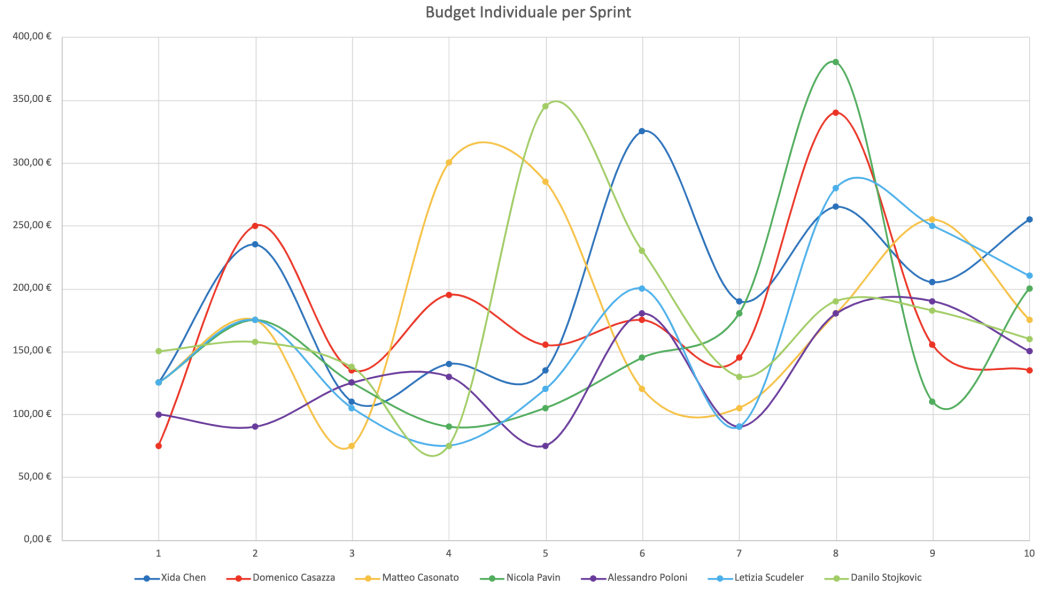
M13 Uniformità al cambio del sistema operativo

Questa metrica indica l'uniformità della mobile App al variare del sistema operativo e della versione su cui viene eseguita.

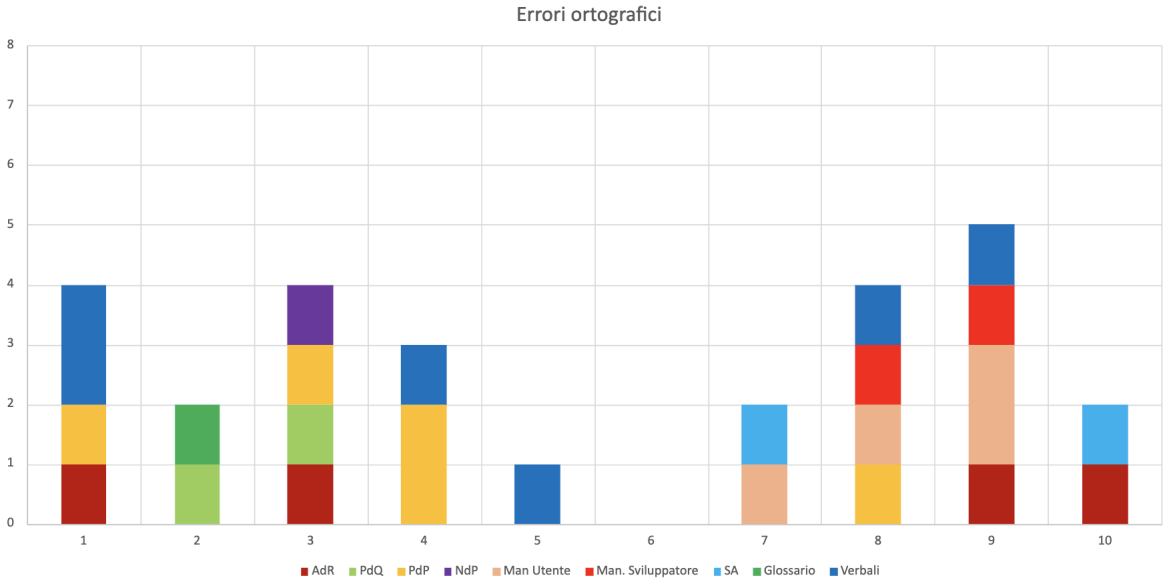
Codice	Descrizione	Valori ottimi	Valori accettabili
M1	Correttezza grammaticale	100%	100%
M2	Indice Gulpease	80	60
M3	Numero di SLOC	20 SLOC/metodo	40 SLOC/metodo
M4	Densità dei commenti	30%	10%
M5	Comprensibilità del codice	80-100%	60-80%
M6	Complessità ciclomatica	5	10
M7	Copertura dei requisiti	100% requisiti individuati	100% requisiti obbligatori
M8	Varianza ripetto al preventivo	0%	30%
M9	Errori per linee di codice	0	0.1
M10	Numero di click	1	5
M11	Tempo di accesso ad una funzionalità	0.0s	2.5s
M12	Uniformità al cambio di Browser (webApp)	100%	80%
M13	Uniformità al cambio di Sistema Operativo (app mobile)	100%	80%

L'andamento delle metriche individuate durante il progetto sono state riportate nei grafici sottostanti.

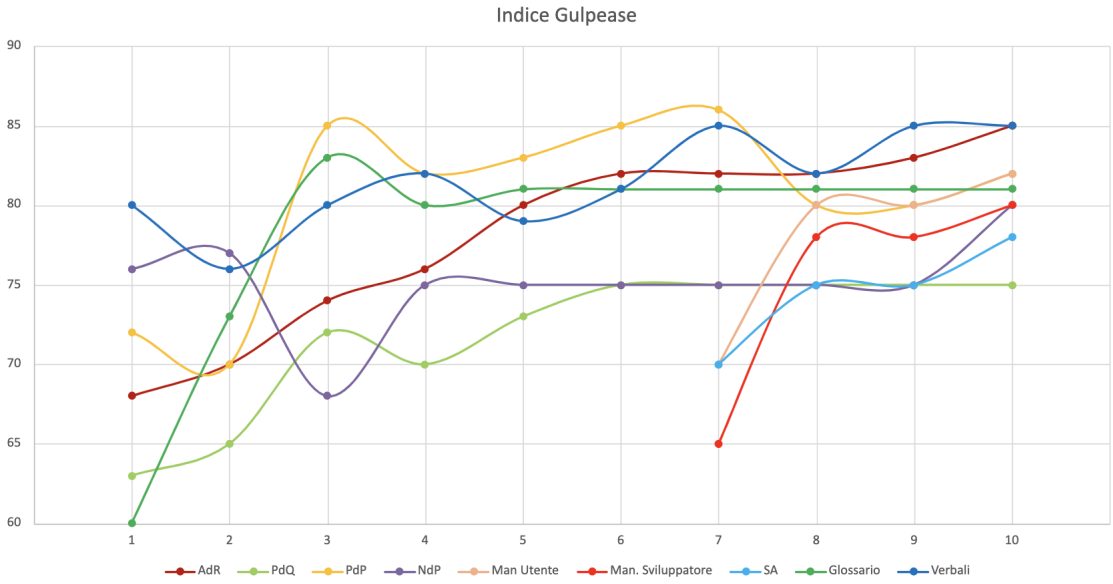
4.2.1 Budget Individuale per Sprint



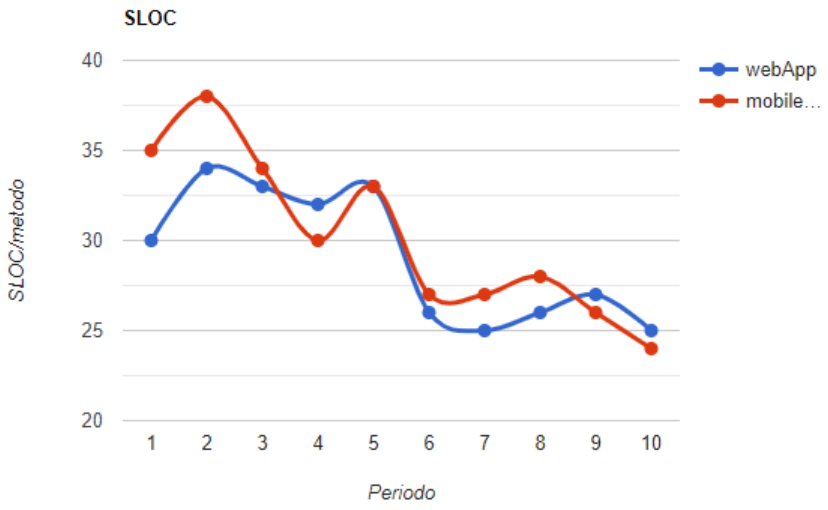
4.2.2 Errori ortografici



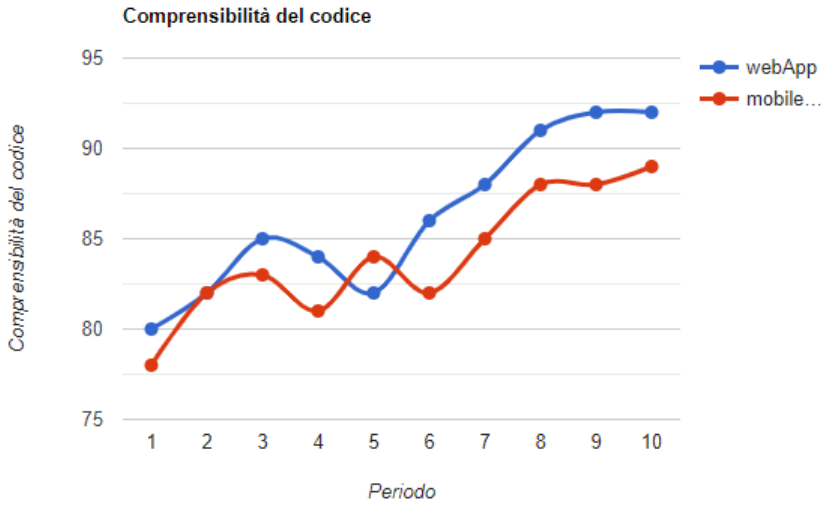
4.2.3 Indice Gulpease



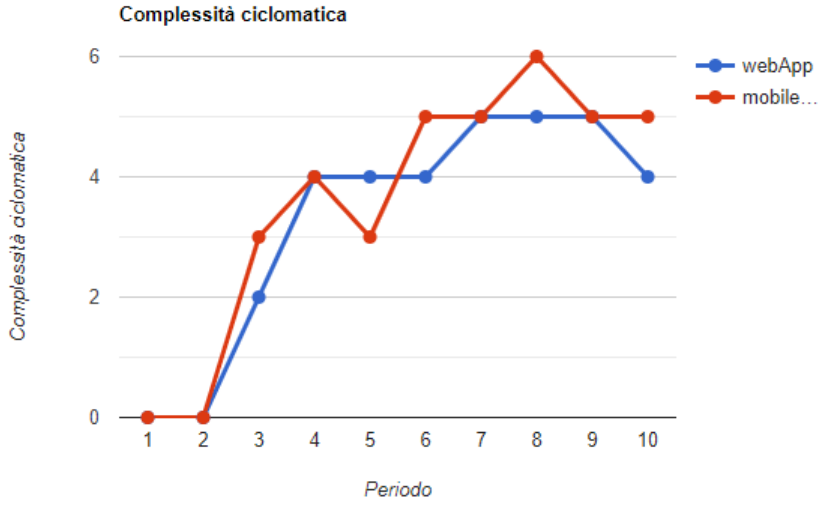
4.2.4 SLOC/metodo



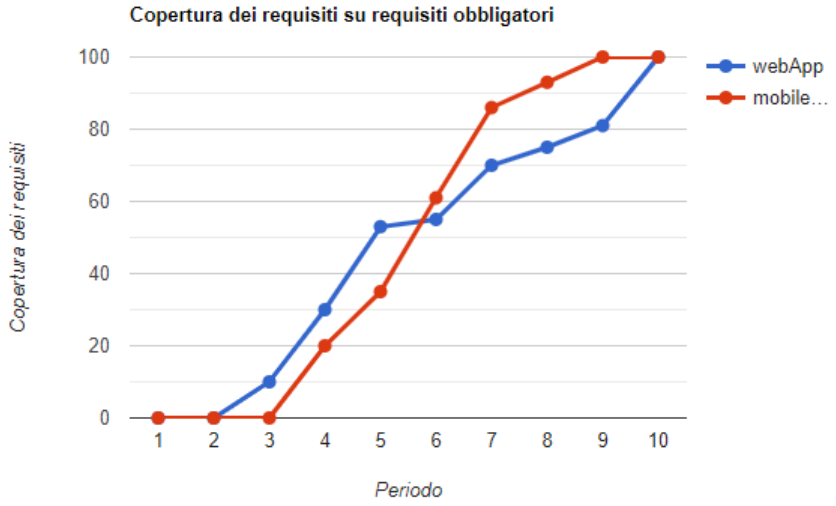
4.2.5 Comprensibilità del codice



4.2.6 Complessità ciclomatica



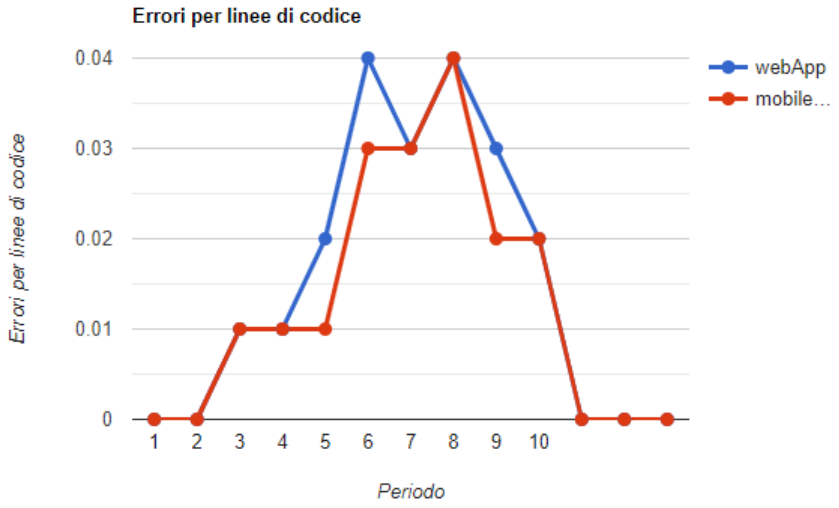
4.2.7 Copertura dei requisiti



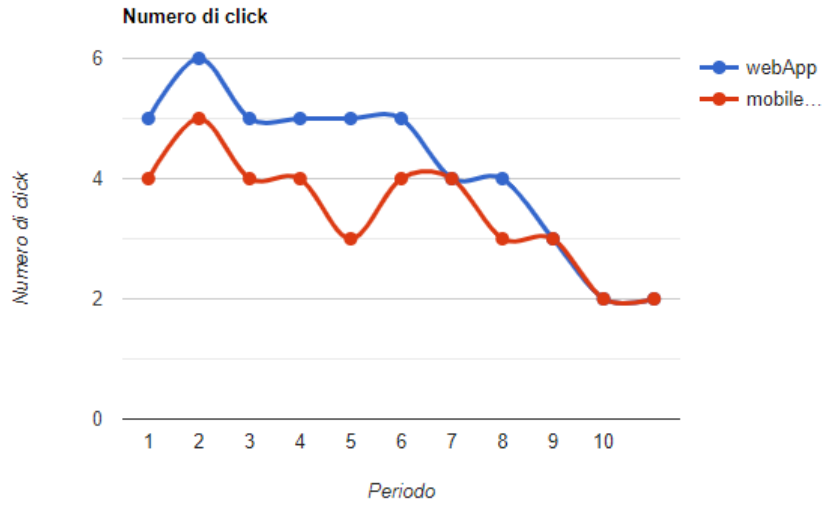
4.2.8 Varianza rispetto al preventivo

Solo durante gli sprint 1 5 7 non si è riusciti ad avere una varianza <30% dato da una cattiva pianificazione per le seguenti ragioni: inesperienza (sprint 1), esami universitari (sprint 7) e stima errata delle ore necessarie al ruolo di programmatore in vista della realizzazione del PoC per RTB (sprint 5).

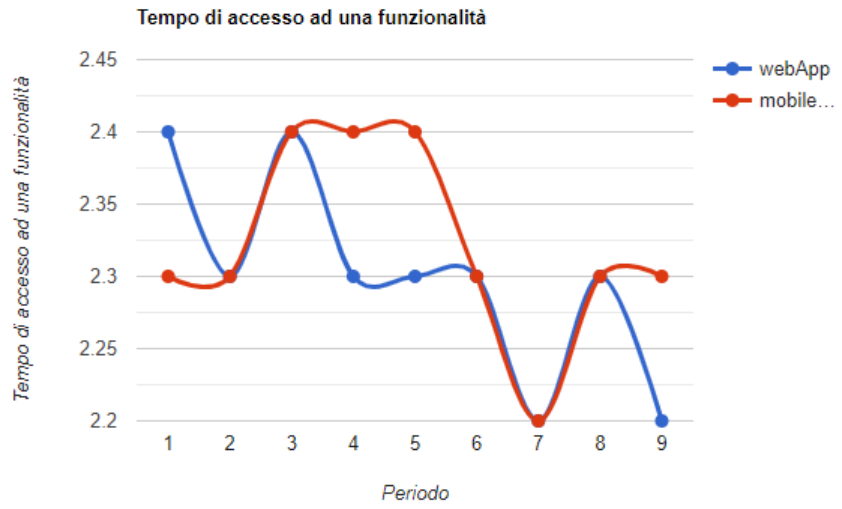
4.2.9 Errori per linee di codice



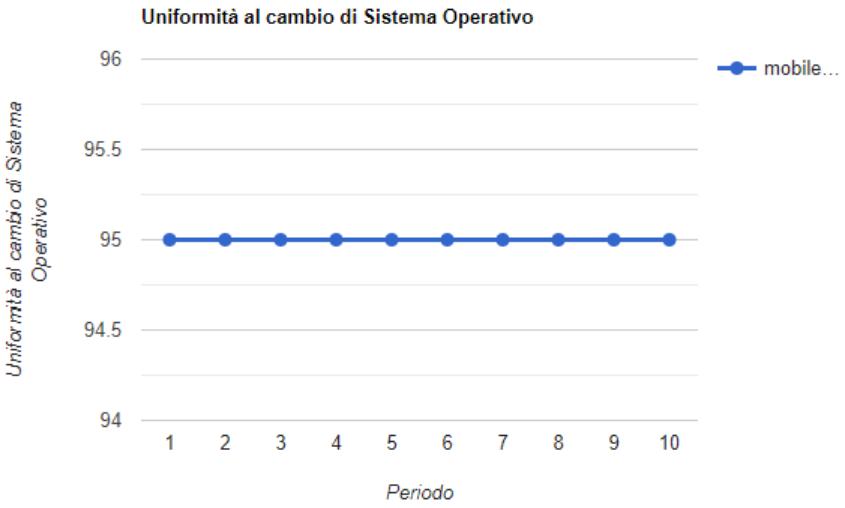
4.2.10 Numero di click



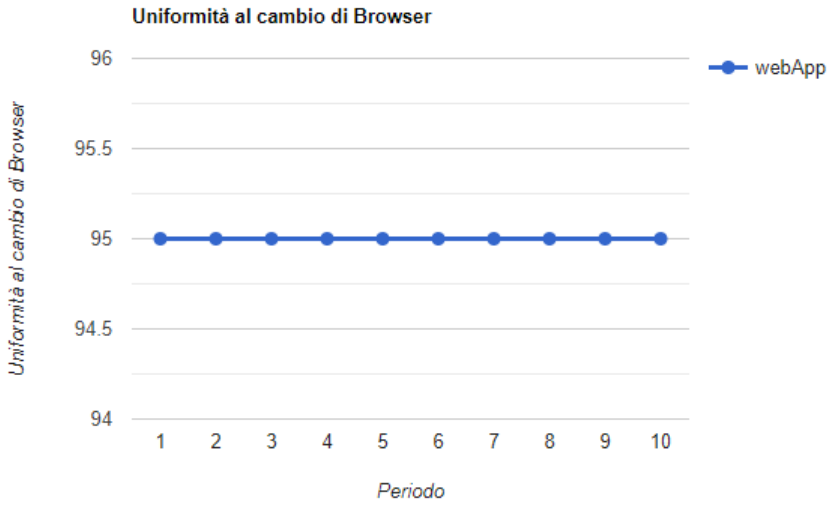
4.2.11 Tempo di accesso ad una funzionalità



4.2.12 Uniformità al cambio di Sistema Operativo



4.2.13 Uniformità al cambio di Browser



5 Specifica test

I test individuati sono necessari per accertarsi che tutte le funzionalità richieste siano corrette e assumino un comportamento atteso.
I test hanno un codice identificativo e possono assumere due stati: superato (S) o non implementato (NI). Le prime due lettere indicano il tipo di test (es: test d'unita è contrassegnato da TU), mentre la terza lettera indica se è un test per un requisito obbligatorio (assenza terza lettera), desiderevole (D) o opzionale (O).

5.1 Test d'unità

Sono stati definiti parzialmente alcuni test d'unità utili anche per la realizzazione del PoC.

5.1.1 Test d'unità - WebApp

Codice	Descrizione	Stato
TU01	Si verifica che il componente *ConnectWallet.jsx* renderizzi correttamente	S
TU02	Si verifica che il componente *Error.jsx* renderizzi correttamente	S
TU03	Si verifica che il componente *Header.jsx* renderizzi correttamente	S
TU04	Si verifica che il componente *Loading.jsx* renderizzi correttamente	S
TU05	Si verifica che il componente *Log.jsx* renderizzi correttamente	S
TU06	Si verifica che il componente *NoWalletDetected.jsx* renderizzi correttamente	S
TU07	Si verifica che *Orders.jsx* renderizzi le transazioni correttamente quando l'utente è un Buyer	S
TU08	Si verifica che *Orders.jsx* renderizzi le transazioni correttamente quando l'utente è un Seller	S
TU09	Si verifica che il componente *RegisterSeller.jsx* renderizzi correttamente	S
TU10	Si verifica che il componente *RegisterSeller.jsx* renderizzi il componente *SwitchNetwork.jsx* quando il network è errato	S
TU11	Si verifica che il componente *RegisterSeller.jsx* renderizzi il componente *NoWalletDetected.jsx* quando non è stato rilevato MetaMask	S
TU12	Si verifica che il componente *RegisterSeller.jsx* renderizzi il componente *ConnectWallet.jsx* quando il wallet non è connesso	S
TU13	Si verifica che il componente *SwitchNetwork.jsx* renderizzi correttamente	S
TU14	Si verifica che il componente *OrderPage.jsx* renderizzi correttamente per il Seller	S

Codice	Descrizione	Stato
TU15	Si verifica che il componente *OrderPage.jsx* renderizzi correttamente per il Buyer	S
TU16	Si verifica che il componente *OrderPage.jsx* renderizzi il componente *NoWalletDetected.jsx* quando non è stato rilevato MetaMask	S
TU17	Si verifica che il componente *OrderPage.jsx* renderizzi il componente *ConnectWallet.jsx* quando il wallet non è connesso	S
TU18	Si verifica che il componente *OrderPage.jsx* renderizzi il componente *SwitchNetwork.jsx* quando il network è errato	S
TU19	Si verifica che il bottone *RefundBuyer* di *OrderPage.jsx* funzioni correttamente con un ordine in stato "Asked Refund" per il Seller	S
TU20	Si verifica che il bottone *RefundBuyer* di *OrderPage.jsx* non venga renderizzato con un ordine in stato "Asked Refund" per il Buyer	S
TU21	Si verifica che il componente *Notify.jsx* renderizzi il messaggio di caricamento correttamente	S
TU22	Si verifica che il componente *Notify.jsx* renderizzi il messaggio di avvenuta notifica dell'e-commerce correttamente	S
TU23	Si verifica che il componente *TxHash.jsx* renderizzi correttamente	S
TU24	Si verifica che il componente *TokenDialog.jsx* renderizzi correttamente tutti i token disponibili	S
TU25	Si verifica che il componente *TokenDialog.jsx* si comporti correttamente quando si seleziona il token AVAX	S
TU26	Si verifica che il componente *TokenDialog.jsx* si comporti correttamente quando si seleziona un token diverso da AVAX	S
TU27	Si verifica che il componente *OrderData.jsx* renderizzi correttamente il prezzo dell'ordine	S
TU28	Si verifica che il componente *OrderData.jsx* abbia inizialmente il token AVAX selezionato con il corretto balance	S
TU29	Si verifica che il componente *OrderData.jsx* renderizzi correttamente una finestra con tutti i token disponibili	S
TU30	Si verifica che il componente *OrderData.jsx* aggiorni correttamente il balance dopo aver selezionato un nuovo token	S
TU31	Si verifica che il componente *OrderData.jsx* renderizzi correttamente il bottone "Approve" dopo aver selezionato un token diverso da AVAX	S
TU32	Si verifica che il componente *OrderData.jsx* renderizzi il bottone "Create Transaction" quando il balance è superiore all'importo da pagare	S

Codice	Descrizione	Stato
TU33	Si verifica che il componente *OrderData.jsx* renderizzi il bottone "Insufficient Balance" quando il balance è inferiore all'importo da pagare	S
TU34	Si verifica che il componente *OrderData.jsx* crei un ordine con i parametri corretti quando si clicca il bottone "Create Transaction"	S
TU35	Si verifica che il componente *OrderData.jsx* rimuova il tasto "Create Transaction" dopo averlo cliccato	S
TU36	Si verifica che la funzione *isValidAmount* di *useFetch.jsx* funzioni correttamente	S
TU37	Si verifica che *useFetch.jsx* imposti le corrette opzioni con il metodo GET	S
TU38	Si verifica che *useFetch.jsx* imposti le corrette opzioni con il metodo PUT	S
TU39	Si verifica che *useFetch.jsx* imposti correttamente l'ordine quando fetch ritorna OK e l'ordine è valido	S
TU40	Si verifica che *useFetch.jsx* imposti un errore quando fetch ritorna OK ma il seller non è autorizzato	S
TU41	Si verifica che *useFetch.jsx* imposti un errore quando fetch ritorna OK ma il prezzo non è in un formato valido	S
TU42	Si verifica che *useFetch.jsx* imposti un errore quando fetch ritorna OK ma avviene un errore di comunicazione con lo smart contract	S
TU43	Si verifica che *useFetch.jsx* imposti un errore quando fetch ritorna NON OK	S
TU44	Si verifica che *useFetch.jsx* imposti un errore quando l'ordine è già stato processato ed inserito nello smart contract	S
TU45	Si verifica che *useFetch.jsx* notifichi correttamente il successo della PUT request	S
TU46	Si verifica che *useFetch.jsx* abortisca la fetch request quando viene smontato	S

5.1.2 Tracciamento Test d'unità - WebApp

ID Test	Metodo
TU01	webApp/src/components/__tests__/webApp_tests/ConnectWallet.test.js: describe("ConnectWallet", () => {test(" TU01 visualizzazione bottone connect wallet", () => {...

ID Test	Metodo
TU02	webApp/src/components/__tests__/webApp_tests/Error.test.js: describe("Error", () => { test('TU02 renders Error correctly', () => {...
TU03	webApp/src/components/__tests__/webApp_tests/Header.test.js: describe("Header", () => { test("TU03 visualizzazione address utente", () => {...
TU04	webApp/src/components/__tests__/webApp_tests/Loading.test.js: describe("loading", () => { test("TU04 visualizzazione pagina caricamento", () => {...
TU05	webApp/src/components/__tests__/webApp_tests/Log.test.js: describe('Log', () => { test('TU05 renders Log with correct data', async () => {...
TU06	webApp/src/components/__tests__/webApp_tests/NoWalletDetected.test.js: describe("NoWalletDetected", () => { test("TU06 visualizzazione wallet non caricato", () => {...
TU07	webApp/src/components/__tests__/webApp_tests/Orders.test.js: describe('Orders', () => { test("TU07 visualizzazione delle transazioni (isBuyer = true)", async () => {...
TU08	webApp/src/components/__tests__/webApp_tests/Orders.test.js: describe('Orders', () => { test("TU08 visualizzazione delle transazioni (isBuyer = false)", async () => {...
TU09	webApp/src/components/__tests__/webApp_tests/RegisterSeller.test.js: describe("RegisterSeller", () => { test("TU09 RegisterSeller standard behaviour", () => {...
TU10	webApp/src/components/__tests__/webApp_tests/RegisterSeller.test.js: describe("RegisterSeller", () => { test("TU10 RegisterSeller Switch Network", () => {...
TU11	webApp/src/components/__tests__/webApp_tests/RegisterSeller.test.js: describe("RegisterSeller", () => { test("TU11 RegisterSeller No Metamask", async () => {...
TU12	webApp/src/components/__tests__/webApp_tests/RegisterSeller.test.js: describe("RegisterSeller", () => { test("TU12 RegisterSeller No Wallet", () => {...
TU13	webApp/src/components/__tests__/webApp_tests/SwitchNetwork.test.js: describe("SwitchNetwork", () => { test("TU13 visualizzazione bottone switch network", () => {...
TU14	webApp/src/components/__tests__/webApp_tests/OrderPage/Created.test.js: describe('OrderPage', () => { test('TU14 renders OrderPage with correct data for Seller', async () => {...

ID Test	Metodo
TU15	webApp/src/components/__tests__/webApp_tests/OrderPage/Created.test.js: describe('OrderPage', () => { test('TU15 renders OrderPage with correct data for Buyer', async () => {...
TU16	webApp/src/components/__tests__/webApp_tests/OrderPage/Created.test.js: describe('OrderPage', () => { test('TU16 renders OrderPage No Metamask', async () => {...
TU17	webApp/src/components/__tests__/webApp_tests/OrderPage/Created.test.js: describe('OrderPage', () => { test('TU17 renders OrderPage No Wallet', async () => {...
TU18	webApp/src/components/__tests__/webApp_tests/OrderPage/Created.test.js: describe('OrderPage', () => { test('TU18 renders OrderPage Switch Network', async () => {...
TU19	webApp/src/components/__tests__/webApp_tests/OrderPage/ AskedRefund.test.js: describe('OrderPage AskedRefund', () => { test('TU19 refund buyer when user is Seller', async () => {...
TU20	webApp/src/components/__tests__/webApp_tests/OrderPage/ AskedRefund.test.js: describe('OrderPage AskedRefund', () => { test('TU20 refund buyer button is not rendered when user is Buyer', async () => {...
TU21	webApp/src/components/__tests__/LandingPage_tests/Notify.test.js: describe('Notify', () => { test('TU21 renders "Notifying e-commerce..." if !hasNotified', () => {...
TU22	webApp/src/components/__tests__/LandingPage_tests/Notify.test.js: describe('Notify', () => { test('TU22 renders "E-commerce notified correctly" if hasNotified', () => {...
TU23	webApp/src/components/__tests__/LandingPage_tests/TxHash.test.js: describe('TxHash', () => { test('TU23 renders TxHash with correct data', () => {...
TU24	webApp/src/components/__tests__/LandingPage_tests/TokenDialog.test.js: describe('TokenDialog', () => { test('TU24 renders all tokens avaiable', () => {...
TU25	webApp/src/components/__tests__/LandingPage_tests/TokenDialog.test.js: describe('TokenDialog', () => { test('TU25 calls onClose after selecting AVAX', () => {...
TU26	webApp/src/components/__tests__/LandingPage_tests/TokenDialog.test.js: describe('TokenDialog', () => { test('TU26 calls onClose after selecting a token', () => {...
TU27	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test display order data', () => { test('TU27 renders order price as passed in prop', async () => {...

ID Test	Metodo
TU28	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to change token', () => { test('TU28 initially renders AVAX as selected token with correct balance', async () => {...
TU29	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to change token', () => { test('TU29 displays all avaiable tokens when clicking button to change selected token', async () => {...
TU30	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to change token', () => { test('TU30 updates balance when token changed', async () => {...
TU31	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to approve', () => { test('TU31 renders Approve button when selecting token different to AVAX', async () => {...
TU32	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to confirm Tx', () => { test('TU32 checks that transaction button is enabled when balance > price', async () => {...
TU33	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to confirm Tx', () => { test('TU33 checks that transaction button is NOT enabled when balance < price', async () => {...
TU34	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to confirm Tx', () => { test('TU34 calls createOrder with correct selected value and amount when button createOrder is clicked', async () => {...
TU35	webApp/src/components/__tests__/LandingPage_tests/OrderData.test.js: describe('OrderData', () => { describe('Test button to confirm Tx', () => { test('TU35 removes button to confirm Tx after it has been clicked', async () => {...
TU36	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('isValidAmount', () => { test('TU36 isValidAmount behaviour', () => {...
TU37	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test fetchOptions', () => { test('TU37 calls fetch with correct GET options', () => {...
TU38	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test fetchOptions', () => { test('TU38 calls fetch with correct PUT options', () => {...

ID Test	Metodo
TU39	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "GET" method', () => { describe('Test checkOrder', () => { test('TU39 calls setOrder and setIsloaded when GET request OK and order is valid', () => {...
TU40	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "GET" method', () => { describe('Test checkOrder', () => { test('TU40 calls setError when GET request OK but seller is not authorized', () => {...
TU41	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "GET" method', () => { describe('Test checkOrder', () => { test('TU41 calls setError when GET request OK but price is invalid', () => {...
TU42	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "GET" method', () => { describe('Test checkOrder', () => { test('TU42 calls setError when GET request OK but isAuthorizedSeller throws an error', () => {...
TU43	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "GET" method', () => { test('TU43 recieves !ok response from fetch and calls setError with code and statusText as recieved', () => {...
TU44	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "GET" method', () => { test('TU44 realizes that order is alrady in chain and calls setOrder, setError, setIsLoaded', () => {...
TU45	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Test "PUT" method', () => { test('TU45 calls setHashNotified when PUT request OK', () => {...
TU46	webApp/src/components/__tests__/LandingPage_tests/useFetch.test.js: describe('useFetch', () => { describe('Abort error', () => { test('TU46 identifies AbortError and does not call setError', () => {...

5.1.3 Test d'unità - Smart Contract

Nota: lo smart contract è stato testato su un fork della mainnet di Avalanche in locale, per garantire il corretto funzionamento dopo il deploy nella rete principale, e non sulle testnet (se funziona in mainnet a maggior ragione funzionerà in testnet).

Codice	Descrizione	Stato
TU01	Si verifica che lo smart contract venga deployato correttamente	S
TU02	Si verifica che un Seller possa registrarsi correttamente	S
TU03	Si verifica che un Seller non possa registrarsi più volte	S

Codice	Descrizione	Stato
TU04	Si verifica che il metodo *createOrderWithAVAXToStable()* funzioni correttamente	S
TU05	Si verifica che il metodo *createOrderWithStable()* funzioni correttamente	S
TU06	Si verifica che il metodo *createOrderWithTokensToStable()* funzioni correttamente	S
TU07	Si verifica che il Seller possa modificare correttamente lo stato di un ordine in "Shipped"	S
TU08	Si verifica che il Buyer possa modificare correttamente lo stato di un ordine in "Confirmed"	S
TU09	Si verifica che il Seller possa modificare correttamente lo stato di un ordine in "Deleted"	S
TU10	Si verifica che il Buyer possa modificare correttamente lo stato di un ordine in "Asked Refund"	S
TU11	Si verifica che il Seller possa modificare correttamente lo stato di un ordine in "Refunded"	S
TU12	Si verifica che il setter *setStablecoinDataFeed()* funzioni correttamente	S
TU13	Si verifica che il setter *setStablecoinAddress()* funzioni correttamente	S
TU14	Si verifica che il setter *setStablecoinPegThreshold()* funzioni correttamente	S
TU15	Si verifica che il getter *getOrdersOfUser()* restituisca correttamente tutti gli ordini di un Buyer	S
TU16	Si verifica che il getter *getOrdersOfUser()* restituisca correttamente tutti gli ordini di un Seller	S
TU17	Si verifica che il getter *getOrdersOfUser()* fallisca correttamente se chiesti ordini di un utente non registato	S
TU18	Si verifica che il getter *getTotalSellers()* funzioni correttamente	S
TU19	Si verifica che il getter *getOrder()* funzioni correttamente	S
TU20	Si verifica che il getter *getBalance()* funzioni correttamente	S
TU21	Si verifica che il getter *getBalance()* funzioni correttamente	S
TU22	Si verifica che il getter *getTotalOrders()* funzioni correttamente	S
TU22	Si verifica che il getter *stablecoinIsPegged()* funzioni correttamente	S
TU23	Si verifica che lo smart contract si comporti adeguatamente in caso di chiamate scorrette (ad esempio, settare come oracolo un Token ERC-20 tramite il setter)	S

5.1.4 Tracciamento Test d'unità - Smart Contract

Tutti i test d'unità relativi allo smart contract si trovano in un unico file, situato in https://github.com/TeamOberon07/contract/test/test.js.

5.2 Test d'integrazione

Sono stati definiti parzialmente alcuni test d'integrazione utili anche per la realizzazione del PoC.

Codice	Descrizione	Stato
TI1	Si verifica che il collegamento tra la webApp e MetaMask avvenga correttamente	NI
TI2	Si verifica che il collegamento tra la app mobile e MetaMask avvenga correttamente	NI
TI3	Si verifica che il recupero dei dati dalla blockchain avvenga correttamente	NI
TI4	Si verifica che l'integrazione con la libreria web3.js avvenga correttamente	NI

5.3 Test di sistema

I test di sistema servono per testare l'applicativo completo nel suo complesso dal punto di vista dell'utente finale.

Codice	Descrizione	Stato
TS1	L'utente non riconosciuto deve poter connettersi al wallet (webApp)	NI
TS2	L'utente non riconosciuto deve poter connettersi al wallet (app mobile)	NI
TS3	Il compratore deve poter confermare il pagamento	NI
TS4	Il compratore deve poter visualizzare l'importo della transazione	NI
TS5	Il compratore deve poter visualizzare l'address del venditore	NI
TS6	L'utente riconosciuto deve poter visualizzare i dati del proprio wallet (webApp)	NI
TS7	L'utente riconosciuto deve poter visualizzare i dati del proprio wallet (mobile)	NI
TS8	Il compratore deve poter scannerizzare il QR code per sbloccare il pagamento	NI
TSD1	L'utente deve poter richiedere il reso	NI

5.4 Test di accettazione

In questa fase del progetto i test di accettazione non sono ancora stati ben definiti dato che il prodotto non è maturo.

5.5 Test di regressione

I test di regressione servono per localizzare errori causati da nuovi versioni del prodotto. In questa fase del progetto non sono ancora stati trovati test di regressione utili.