Piano di Qualifica
Progetto ShopChain

yakuzaishi.swe@gmail.com

Informazioni sul documento

Responsabile	Matteo Midena
Redattori	Francesco Bugno
	Luca Carturan
Verificatori	Michele Filosofo
	Matteo Midena
Uso	Esterno
Destinatari	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Ing. Fabio Pallaro - Sync Lab S.r.l.
Versione	3.0.0

Sommario

Questo documento raccoglie le strategie di qualifica e gli obiettivi di qualità perseguiti dal gruppo *Yakuzaishi* nello svolgimento del progetto ShopChain.

Registro delle Modifiche

Versione	Data	Autore	Ruolo	Descrizione
3.0.0	2022/06/07	Luca Carturan	Responsabile	Approvato per il rilascio
2.1.0	2022/06/06	Matteo Midena	Verificatore	Verifica generale del documento.
2.0.3	2022/06/05	Francesco Bugno Michele Filosofo	Progettista Verificatore	Aggiornamento §A e verifica.
2.0.2	2022/06/04	Luca Carturan Matteo Midena	Progettista Verificatore	Aggiornamento §4 e verifica.
2.0.1	2022/06/03	Francesco Bugno Michele Filosofo	Progettista Verificatore	Aggiornamento post PB e verifica.
2.0.0	2022/05/15	Matteo Midena	Responsabile	Approvato per il rilascio
1.1.0	2022/05/15	Michele Filosofo	Verificatore	Verifica generale del documento
1.0.3	2022/05/14	Luca Carturan Matteo Midena	Progettista Verificatore	Aggiornamento §2.2.2, §2.2.3 e verifica
1.0.2	2022/04/26	Luca Carturan Matteo Midena	Progettista Verificatore	Stesura §4.4.1 e verifica
1.0.1	2022/04/23	Luca Carturan Matteo Midena	Progettista Verificatore	Aggiornamento §4.4.2, stesura §4.4.3 e verifica
1.0.0	2022/02/21	Matteo Midena	Responsabile	Approvato per il rilascio
0.4.0	2022/02/20	Michele Filosofo	Verificatore	Verifica generale documento
0.3.2	2022/02/10	Francesco Bugno	Progettista	Sistemati grafici §A
0.3.1	2022/02/04	Luca Carturan	Progettista	Stesura §A
0.3.0	2022/01/25	Michele Filosofo	Verificatore	Verifica generale documento
0.2.4	2022/01/22	Francesco Bugno	Progettista	Correzione errori ortografici, aggiornata §4
0.2.3	2022/01/20	Luca Carturan	Progettista	Terminata stesura §4
0.2.2	2022/01/18	Francesco Bugno	Progettista	Aggiornamento §3, iniziata stesura §4
0.2.1	2022/01/18	Luca Carturan	Progettista	Continuo stesura §3, aggiornate tabelle
0.2.0	2022/01/17	Michele Filosofo	Verificatore	Verifica generale del documento

Versione	Data	Autore	Ruolo	Descrizione
0.1.3	2022/01/14	Francesco Bugno	Amministratore	Correzione errori ortografici, iniziata stesura §3
0.1.2	2022/01/09	Luca Carturan	Amministratore	Aggiornamento tabella §2, stesura §2.2.2 e §2.2.3
0.1.1	2022/01/04	Francesco Bugno	Amministratore	Iniziata stesura §2
0.1.0	2022/01/06	Michele Filosofo	Verificatore	Verifica generale del documento
0.0.2	2021/12/27	Luca Carturan	Amministratore	Stesura §1
0.0.1	2021/12/26	Luca Carturan	Amministratore	Creata struttura del documento

Contenuti

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di descrivere le strategie che il gruppo usa per perseguire gli obbiettivi di qualità da applicare al progetto.
Esso deve implementare degli standard che permettano il miglioramento continuo, tramite misurazioni periodiche dei risultati ottenuti, sfruttandoli per ottenere azioni migliorative.
All'interno del *Piano di progetto* vengono raccolte le definizioni dei test, il loro stato e il loro tracciamento rispetto ai requisiti individuati nell'*Analisi dei requisiti v3.0.0*.

1.2 Scopo del capitolato

L'avvento delle tecnologie BlockChain$_G$ ha portato e porterà nei prossimi anni a grandi cambiamenti nella società.
In particolare, ha aperto le porte a una nuova forma di finanza, la cosiddetta "DeFi" (Finanza Decentralizzata) che ha permesso a chiunque sia dotato di connessione internet di creare un Wallet$_G$ e possedere quindi criptovalute$_G$.
Questo ha delineato due profili critici strettamente legati: da un lato il controllo del proprio portafoglio è passato completamente nelle mani dell'utente, dall'altro lato questo comporta la mancanza di un ente terzo che si occupi di gestire transazioni e offrire garanzie.
Nel capitolato in questione si vuole proprio risolvere questo problema, in uno scenario che comprende un e-commerce$_G$ basato su BlockChain$_G$ in cui si vuole tutelare entrambe le parti coinvolte in un acquisto tramite criptovalute$_G$.
Il fine del progetto è la realizzazione di un prototipo di una piattaforma integrabile con un "crypto-ecommerce"$_G$, che si occupi di gestire gli ordini dalle fasi di pagamento alla consegna.

1.3 Glossario

I termini utilizzati in questo documento potrebbero generare dubbi riguardo al loro significato, richiedendo pertanto una definizione al fine di evitare ambiguità. Tali termini vengono contrassegnati da una G maiuscola finale a pedice della parola. La loro spiegazione è riportata nel *Glossario v2.0.0*.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- ***Norme di progetto v3.0.0***;
- **Regolamento del progetto didattico:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/PD2.pdf

1.4.2 Riferimenti informativi

- ***Analisi dei requisiti v3.0.0***;
- **Capitolato d'appalto C2 - ShopChain:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Progetto/C2.pdf
- **Software Engineering - Ian Sommerville - 10th Edition:**
  - Capitolo 24 - Quality management.
- **Qualità di prodotto - slide T12 del corso di Ingegneria del Software:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/T12.pdf

- **Qualità di processo - slide T13 del corso Ingegneria del Software:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/T13.pdf
- **Verifica e validazione: introduzione - slide T14 del corso Ingegneria del Software:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/T14.pdf
- **Verifica e validazione: analisi statica - slide T15 del corso Ingegneria del Software:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/T15.pdf
- **Verifica e validazione: analisi dinamica - slide T16 del corso Ingegneria del Software:**
  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/T16.pdf
- **ISO/IEC 9126:**
  http://www.colonese.it/00-Manuali_Pubblicatii/07-ISO-IEC9126_v2.pdf
- **ISO/IEC 12207:1997:**
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
- **Indice di Gulpease:**
  https://it.wikipedia.org/wiki/Indice_Gulpease

2 Qualità di processo

Per garantire la qualità dei processi, è stato scelto di utilizzare come riferimento lo standard ISO/IEC 12207:1997, dal quale sono stati semplificati e adattati alle necessità alcuni dei processi in esso elencati. In questa sezione vengono esposti i valori ottimali e accettabili riguardanti le metriche scelte, per consultare nel dettaglio le metriche sotto riportate, fare riferimento al documento *Norme di progetto v3.0.0*.

2.1 Obiettivi di qualità di processo

Obiettivo	Descrizione	Metriche
Processi primari		
Fornitura	Processo che consiste nel decidere procedure e risorse adatte a soddisfare le necessità del cliente.	MPC01 MPC02 MPC03 MPC04 MPC05 MPC06 MPC07
Sviluppo	Processo che ha lo scopo di sviluppare un prodotto software che indirizzi le esigenze del cliente.	MPC08
Processi di supporto		
Verifica	Processo che ha lo scopo di confermare che ciascun servizio realizzato soddisfi i requisiti specificati.	MPC09
Gestione della qualità	Processo che ha lo scopo di assicurare che il prodotto e i servizi offerti siano conformi agli standard definiti.	MPC10
Processi organizzativi		
Gestione organizzativa	Processo che si occupa di esporre le modalità di coordinamento del gruppo.	MPC11

Tabella 2: Obiettivi di qualità di processo

2.2 Metriche utilizzate

2.2.1 Processi primari

Codice	Nome metrica	Valore accettabile	Valore ottimale
Fornitura			
MPC01	Earned Value (EV)	> 0	$\leq EAC$
MPC02	Actual Cost (AC)	≥ 0	$\leq EAC$
MPC03	Planned Value (PV)	≥ 0	$\leq BAC$
MPC04	Cost Variance (CV)	$\geq -10\%$	$\geq 0\%$
MPC05	Schedule Variance (SV)	$\geq -10\%$	$\geq 0\%$
MPC06	Estimated At Completion (EAC)	$\geq BAC - 3\%; \leq BAC + 3\%$	$= BAC$
MPC07	Estimate To Complete (ETC)	≥ 0	$\leq EAC$
Sviluppo			
MPC08	Requirements Stability Index (RSI)	$\geq 80\%$	100%

Tabella 3: Metriche di qualità dei processi primari

2.2.2 Processi di supporto

Codice	Nome metrica	Valore accettabile	Valore ottimale
Verifica			
MPC09	Passed Tests	$\geq 80\%$	100%
Gestione della qualità			
MPC10	Metrics Satisfied	$\geq 85\%$	100%

Tabella 4: Metriche di qualità dei processi di supporto

2.2.3 Processi organizzativi

Codice	Nome metrica	Valore accettabile	Valore ottimale
Gestione organizzativa			
MPC11	Risks Found	≤ 5	0

Tabella 5: Metriche di qualità dei processi organizzativi

3 Qualità di prodotto

Per garantire la qualità di prodotto, è stato scelto di utilizzare come riferimento lo standard ISO/IEC 9126. In questa sezione vengono esposti i valori ottimali e accettabili riguardanti le metriche selezionate dal gruppo *Yakuzaishi*. Per consultare nel dettaglio le metriche sotto riportate, fare riferimento al documento *Norme di progetto v3.0.0*.

Obiettivo	Descrizione	Metriche
Monitoraggio documentazione		
Leggibilità documenti	I documenti devono essere comprensibili agli utenti.	MPD01
Correttezza linguistica	Tutti gli errori grammaticali devono essere corretti.	MPD02
Monitoraggio software		
Funzionalità	Capacità del prodotto di offrire tutte le funzioni individuate nell'*Analisi dei requisiti*, perseguendo accuratezza e adeguatezza.	MPD03 MPD04 MPD05
Usabilità	Capacità di essere comprensibile in modo da rendere piacevole l'esperienza dell'utente. Le funzionalità devono essere compatibili con le aspettative.	MPD06
Portabilità	Capacità di poter funzionare in diversi ambienti di esecuzione. Gli obiettivi da perseguire sono adattabilità e sostituibilità.	MPD07
Copertura test	Il codice prodotto dovrà essere verificato nella sua interezza al fine di garantire la corretta implementazione dei requisiti individuati.	MPD08 MPD09 MPD10 MPD11 MPD12 MPD13 MPD14 MPD15

Tabella 6: Obiettivi di qualità di prodotto

3.1 Metriche utilizzate

Codice	Nome Metrica	Valore accettabile	Valore ottimale
Documenti			
MPD01	Indice di Gulpease	$\geq 60\%$	$\geq 80\%$
MPD02	Errori ortografici	0	0
Software			
MPD03	Copertura requisiti obbligatori	100%	100%
MPD04	Copertura requisiti desiderabili	$\geq 90\%$	100%
MPD05	Copertura requisiti opzionali	$\geq 80\%$	100%
MPD06	Facilità di utilizzo	5 click	4 click
MPD07	Versioni browser supportate	$\geq 80\%$	100%
MPD08	Solidity Statement Coverage	$\geq 80\%$	100%
MPD09	Solidity Branch Coverage	$\geq 80\%$	100%
MPD10	Solidity Function Coverage	$\geq 80\%$	100%
MPD11	Solidity Line Coverage	$\geq 80\%$	100%
MPD12	Frontend Statement Coverage	$\geq 80\%$	100%
MPD13	Frontend Branch Coverage	$\geq 80\%$	100%
MPD14	Frontend Function Coverage	$\geq 80\%$	100%
MPD15	Frontend Line Coverage	$\geq 80\%$	100%

Tabella 7: Metriche di qualità di prodotto

4 Specifica dei test

Il codice utilizzato per l'identificazione dei test è specificato dettagliatamente nelle *Norme di progetto v3.0.0*, mentre delle sigle utili per la comprensione delle tabelle seguenti sono:

- **S** = test superato;
- **NI** = test non implementato.

4.1 Test di accettazione

I test di accettazione sono necessari per dimostrare che il prodotto soddisfi i requisiti minimi concordati con il proponente.
Essi si compongono dei test di sistema e vengono eseguiti durante il collaudo finale sia dai membri del gruppo che dall'azienda proponente, sotto supervisione del gruppo stesso.

4.2 Test di sistema

ID Test	Descrizione	ID Requisiti	Stato
TS1F1	Verificare che la richiesta di checkout da un E-Commerce$_G$ avvenga correttamente.	R1F1	S
TS1F2	Verificare che l'utente visualizzi correttamente le diverse tipologie di pagamento.	R1F2	S
TS1F2.1	Verificare che l'utente possa scegliere la tipologia di pagamento unico correttamente.	R1F2.1	S
TS2F2.2	Verificare che l'utente possa scegliere la tipologia di pagamento MoneyBox$_G$ correttamente.	R2F2.2	S
TS2F2.2.1	Verificare che l'utente possa visualizzare lo stato di completamento della MoneyBox$_G$ correttamente.	R2F2.2.1	S
TS2F2.2.2	Verificare che l'utente possa copiare l'invito di partecipazione alla MoneyBox$_G$ correttamente.	R2F2.2.2	S
TS3F2.2.3	Verificare che l'utente possa visualizzare una traduzione visiva della MoneyBox$_G$ correttamente.	R3F2.2.3	NI
TS2F2.2.4	Verificare che in caso di chiusura della MoneyBox$_G$ i soldi vengano restituiti correttamente.	R2F2.2.4	NI
TS2F2.2.5	Verificare che l'utente possa visualizzare l'elenco delle transazione dei partecipanti alla MoneyBox$_G$ correttamente.	R2F2.2.5	S

Continua nella pagina successiva

ID Test	Descrizione	ID Requisiti	Stato
TS1F3	Verificare che l'utente possa visualizzare il totale dell'ordine correttamente.	R1F3	S
TS1F4	Verificare che l'utente possa effettuare la connessione a Metamask$_G$ correttamente.	R1F4	NI
TS1F5	Verificare che l'utente possa pagare correttamente.	R1F5	NI
TS1F5.1	Verificare che il versamento per il pagamento unico copra l'intera somma richiesta.	R1F5.1	NI
TS2F5.2	Verificare che il versamento per il pagamento tramite MoneyBox$_G$ possa coprire solo una parte della somma richiesta.	R2F5.2	NI
TS1F5.4	Verificare che l'utente possa visualizzare correttamente un messaggio d'errore nel caso in cui la transazione fallisca.	R1F5.4	NI
TS1F6	Verificare che la richiesta di rimborso funzioni correttamente.	R1F6	NI
TS1F7	Verificare che il proprietario dell'ordine possa sbloccare, correttamente, i fondi dallo Smart Contract$_G$ dopo avvenuta ricezione.	R1F7	S
TS1F7.1	Verificare che il proprietario dell'ordine possa visualizzare correttamente il codice di sblocco.	R1F7.1	S
TS1F8	Verificare che l'utente possa visualizzare le transazioni.	R1F8	S
TS2F8.1	Verificare che il venditore possa visualizzare le transazioni in entrata pagate.	R2F8.1	S
TS2F8.1.1	Verificare che il venditore possa visualizzare le transazioni in entrata pagate ma non sbloccate.	R2F8.1.1	S
TS2F8.1.2	Verificare che il venditore possa visualizzare le transazioni in entrata pagate e sbloccate.	R2F8.1.2	S
TS2F8.1.3	Verificare che il venditore possa visualizzare le transazioni in entrata pagate cancellate.	R2F8.1.3	S
TS1F8.2	Verificare che il proprietario dell'ordine possa visualizzare le transazioni in uscita.	R1F8.2	S

Continua nella pagina successiva

ID Test	Descrizione	ID Requisiti	Stato
TS2F8.2.1	Verificare che il proprietario dell'ordine possa visualizzare le transazioni in uscita non pagate.	R2F8.2.1	S
TS2F8.2.2	Verificare che il proprietario dell'ordine possa visualizzare le transazioni in uscita pagate ma non sbloccate.	R2F8.2.2	S
TS2F8.2.3	Verificare che il proprietario dell'ordine possa visualizzare le transazioni in uscita pagate e sbloccate.	R2F8.2.3	S
TS2F8.2.4	Verificare che il proprietario dell'ordine possa visualizzare le transazioni in uscita cancellate.	R2F8.2.4	S
TS3F9	Verificare che si possa convertire correttamente l'ammontare depositato in stable coin$_{\mathbf{G}}$.	R3F9	NI
TS3F10	Verificare che, dopo la transazione, una fee percentuale venga destinata al fondo ShopChain correttamente.	R3F10	NI
TS1F11	Verificare che l'utente possa visualizzare correttamente l'indirizzo del suo wallet$_{\mathbf{G}}$.	R1F11	S
TS1F11.1	Verificare che l'utente possa visualizzare correttamente l'indirizzo del suo wallet$_{\mathbf{G}}$ in forma testuale.	R1F11.1	S
TS1F11.2	Verificare che l'utente possa visualizzare correttamente un avviso della mancata connessione a Metamask$_{\mathbf{G}}$.	R1F11.2	S
TS3F11.3	Verificare che l'utente possa visualizzare correttamente l'indirizzo del suo wallet$_{\mathbf{G}}$ sotto forma di sequenza di emoji.	R3F11.3	S
TS3F12	Verificare che gli utenti partecipanti ad una MoneyBox$_{\mathbf{G}}$ possano ricevere una notifica al suo completamento.	R3F12	NI
TS1F14	Verificare che l'utente possa visualizzare correttamente lo stato di connessione a Metamask$_{\mathbf{G}}$.	R1F14	S
TS1F14.1	Verificare che l'utente possa visualizzare un suggerimento se connesso correttamente.	R1F14.1	S
TS1F14.2	Verificare che l'utente possa visualizzare nel caso in cui non sia installato Metamask$_{\mathbf{G}}$.	R1F14.2	NI

Continua nella pagina successiva

ID Test	Descrizione	ID Requisiti	Stato
TS1F14.3	Verificare che l'utente possa visualizzare un errore nel caso in cui la blockchain$_G$ selezionata non sia corretta.	R1F14.3	S
TS1F14.4	Verificare che l'utente possa visualizzare un errore nel caso in cui non abbia connesso un account a ShopChain.	R1F14.4	NI
TS1F15	Verificare che l'utente possa visualizzare un messaggio di avviso nel caso in cui la transazione sia già presente in blockchain$_G$.	R1F15	NI
TS1F16	Verificare che l'utente possa visualizzare una pagina con i dettagli dell'ordine pagato.	R1F16	NI
TS1F16.1	Verificare che l'utente possa visualizzare i dettagli dell'ordine pagato.	R1F16.1	NI
TS1F16.1.1	Verificare che l'utente possa visualizzare l'id dell'ordine pagato.	R1F16.1.1	NI
TS1F16.1.2	Verificare che l'utente possa visualizzare l'indirizzo del venditore dell'ordine pagato.	R1F16.1.2	NI
TS1F16.1.3	Verificare che l'utente possa visualizzare l'ammontare dell'ordine pagato.	R1F16.1.3	NI
TS1F16.1.4	Verificare che l'utente possa visualizzare lo stato dell'ordine pagato.	R1F16.1.4	NI
TS1F16.1.5	Verificare che l'utente possa visualizzare la data dell'ordine pagato.	R1F16.1.5	NI

Tabella 8: Test di sistema

4.3 Test d'integrazione

Codice	Descrizione	Stato
TI1	Verificare che il collegamento con Metamask$_G$ avvenga correttamente.	S
TI2	Verificare che il collegamento tra Front-End$_G$ e Smart Contract$_G$ tramite libreria Web3.js$_G$ avvenga correttamente.	S

Tabella 9: Test d'integrazione

4.4 Test di unità

4.4.1 Test di unità - Frontend

Codice	Descrizione	Stato
TUF01	Si verifica che un'istanza di OrderManager venga creata correttamente (costruttore parametrizzato).	S
TUF02	Si verifica che un'istanza di OrderManager venga creata correttamente (costruttore di default).	S
TUF03	Si verifica che ritorni correttamente il contatore degli ordini (OrderManager - istanza definita).	S
TUF04	Si verifica che ritorni correttamente il contatore degli ordini (OrderManager - istanza non definita).	S
TUF05	Si verifica che ritorni correttamente il bilancio presente nel contratto (OrderManager - istanza definita).	S
TUF06	Si verifica che ritorni correttamente il bilancio presente nel contratto (OrderManager - istanza non definita).	S
TUF07	Si verifica che un'istanza di MoneyBoxManager venga creata correttamente (costruttore parametrizzato).	S
TUF08	Si verifica che un'istanza di MoneyBoxManager venga creata correttamente (costruttore di default).	S
TUF09	Si verifica che un'istanza di OrderManagerRepo venga creata correttamente.	S
TUF10	Si verifica che ritorni correttamente il bilancio presente nel contratto (OrderManagerRepo - istanza definita).	S
TUF11	Si verifica che ritorni correttamente il bilancio presente nel contratto (OrderManagerRepo - istanza non definita).	S
TUF12	Si verifica che ritorni correttamente il contatore degli ordini (OrderManagerRepo - istanza definita).	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF13	Si verifica che ritorni correttamente il contatore degli ordini (OrderManagerRepo - istanza non definita).	S
TUF14	Si verifica che un'istanza di MoneyBoxManagerRepo venga creata correttamente.	S
TUF15	Si verifica che un'istanza di ContractStore venga creata correttamente.	S
TUF16	Si verifica che un'istanza di Amount venga creata correttamente.	S
TUF17	Si verifica che l'ammontare venga aggiornato correttamente.	S
TUF18	Si verifica che il valore in $\text{FTM}_{\mathbf{G}}$ ritornato sia corretto.	S
TUF19	Si verifica che l'ammontare venga assegnato correttamente.	S
TUF20	Si verifica che un'istanza di MoneyBox venga creata correttamente.	S
TUF21	Si verifica che l'ammontare deciso venga inserito correttamente all'interno della $\text{MoneyBox}_{\mathbf{G}}$.	S
TUF22	Si verifica che ritorni correttamente la lista dei pagamenti relativi la $\text{MoneyBox}_{\mathbf{G}}$.	S
TUF23	Si verifica che un'istanza di Order venga creata correttamente.	S
TUF24	Si verifica che un ordine venga costruito correttamente.	S
TUF25	Si verifica che ritorni correttamente l'ammontare relativo all'ordine.	S
TUF26	Si verifica che l'ammontare relativo all'ordine venga sbloccato correttamente.	S
TUF27	Si verifica che l'ammontare relativo all'ordine venga rimborsato correttamente.	S
TUF28	Si verifica che lo stato dell'ordine venga aggiornato correttamente.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF29	Si verifica che l'ordine venga aggiornato correttamente.	S
TUF30	Si verifica che la collezione di ordini aggiunga gli ordini correttamente.	S
TUF31	Si verifica che la collezione di ordini ritorni l'ordine corretto dato l'ID.	S
TUF32	Si verifica che la collezione di ordini ritorni gli ordini corretti dato il venditore.	S
TUF33	Si verifica che la collezione di ordini ritorni gli ordini corretti dato il proprietario degli ordini.	S
TUF34	Si verifica che la collezione di ordini venga aggiornata correttamente.	S
TUF35	Si verifica che un'istanza di OrderState venga creata correttamente.	S
TUF36	Si verifica che lo stato dell'ordine sia "Not Created".	S
TUF37	Si verifica che lo stato dell'ordine sia "Created".	S
TUF38	Si verifica che lo stato dell'ordine sia "Paid".	S
TUF39	Si verifica che lo stato dell'ordine sia "Closed".	S
TUF40	Si verifica che lo stato dell'ordine sia "Cancelled".	S
TUF41	Si verifica che lo stato dell'ordine venga modificato correttamente.	S
TUF42	Si verifica che l'indirizzo del pagamento sia corretto.	S
TUF43	Si verifica che l'ammontare del pagamento sia corretto.	S
TUF44	Si verifica che la data del pagamento sia corretta.	S
TUF45	Si verifica che il pagamento venga creato correttamente.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF46	Si verifica che un'istanza di OrderRepo venga creata correttamente.	S
TUF47	Si verifica che l'ordine venga creato correttamente (OrderRepo - istanza definita).	S
TUF48	Si verifica che l'ordine venga creato correttamente (OrderRepo - istanza non definita).	S
TUF49	Si verifica che l'ordine venga sbloccato correttamente (OrderRepo - istanza definita).	S
TUF50	Si verifica che l'ordine venga sbloccato correttamente (OrderRepo - istanza non definita).	S
TUF51	Si verifica che l'ordine venga rimborsato correttamente (OrderRepo - istanza definita).	S
TUF52	Si verifica che l'ordine venga rimborsato correttamente (OrderRepo - istanza non definita).	S
TUF53	Si verifica che ritorni l'ordine corretto dato l'ID (OrderRepo - istanza definita).	S
TUF54	Si verifica che ritorni l'ordine corretto dato l'ID (OrderRepo - istanza non definita).	S
TUF55	Si verifica che ritorni gli ordini corretti dato il venditore (OrderRepo - istanza definita).	S
TUF56	Si verifica che ritorni gli ordini corretti dato il venditore (OrderRepo - istanza non definita).	S
TUF57	Si verifica che ritorni gli ordini corretti dato l'acquirente (OrderRepo - istanza definita).	S
TUF58	Si verifica che ritorni gli ordini corretti dato l'acquirente (OrderRepo - istanza non definita).	S
TUF59	Si verifica che un'istanza di MoneyBoxOrderRepo venga creata correttamente.	S
TUF60	Si verifica che un nuovo pagamento venga effettuato correttamente (MoneyBoxOrderRepo - istanza definita).	S
TUF61	Si verifica che un nuovo pagamento venga effettuato correttamente (MoneyBoxOrderRepo - istanza non definita).	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF62	Si verifica che ritorni la lista dei pagamenti relativi ad una $\text{MoneyBox}_{\mathbf{G}}$ correttamente (MoneyBoxOrderRepo - istanza definita).	S
TUF63	Si verifica che ritorni la lista dei pagamenti relativi ad una $\text{MoneyBox}_{\mathbf{G}}$ correttamente (MoneyBoxOrderRepo - istanza non definita).	S
TUF64	Si verifica che ritorni l'ammontare mancante al completamento di una $\text{MoneyBox}_{\mathbf{G}}$ (MoneyBoxOrderRepo - istanza definita).	S
TUF65	Si verifica che ritorni l'ammontare mancante al completamento di una $\text{MoneyBox}_{\mathbf{G}}$ (MoneyBoxOrderRepo - istanza non definita).	S
TUF66	Si verifica che un'istanza di OrderStore venga creata correttamente.	S
TUF67	Si verifica che un ordine venga creato correttamente tramite OrderStore.	S
TUF68	Si verifica che un istanza di LockOverlayViewModel venga creata correttamente.	S
TUF69	Si verifica che il LockOverlayViewModel sia connesso al provider (provider corretto).	S
TUF70	Si verifica che il LockOverlayViewModel sia connesso al provider (provider non corretto).	S
TUF71	Si verifica che makeAutoObservable sia chiamata alla creazione di ConnectMetamaskViewModel.	S
TUF72	Si verifica che la funzione connect di ConnectMetamaskViewModel sia chiamata correttamente.	S
TUF73	Si verifica che la severità di MetamaskErrorViewModel sia esattamente quella data dal provider.	S
TUF74	Si verifica che il nome di MetamaskErrorViewModel sia esattamente quello dato dal provider.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF75	Si verifica che la descrizione di MetamaskErrorViewModel sia esattamente quella data dal provider.	S
TUF76	Si verifica che makeAutoObservable sia chiamata alla creazione di NavViewModel.	S
TUF77	Si verifica che l'address sia corretto in caso di connessione al provider di NavViewModel.	S
TUF78	Si verifica che l'address sia non corretto in caso di fallita connessione al provider di NavViewModel.	S
TUF79	Si verifica che un istanza di ECommerceViewModel sia creata correttamente.	S
TUF80	Si verifica che l'istanza di ECommerceViewModel ritorni l'amount corretto.	S
TUF81	Si verifica che l'istanza di ECommerceViewModel ritorni i wei corretti.	S
TUF82	Si verifica che l'istanza di ECommerceViewModel ritorni l'id corretto.	S
TUF83	Si verifica che setAmount di ECommerceViewModel sia corretta.	S
TUF84	Si verifica che handleSubmit di ECommerceViewModel sia corretta.	S
TUF85	Si verifica che l'istanza di ChoiceViewModel sia creata correttamente.	S
TUF86	Si verifica che createOrder di ChoiceViewModel sia corretta.	S
TUF87	Si verifica che canRedirect di ChoiceViewModel sia corretta.	S
TUF88	Si verifica che isBusy di ChoiceViewModel sia corretta.	S
TUF89	Si verifica che isFailed di ChoiceViewModel sia corretta.	S
TUF90	Si verifica che setAmount di ChoiceViewModel sia corretta.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF91	Si verifica che setId di ChoiceViewModel sia corretta.	S
TUF92	Si verifica che un'istanza di PickAmountViewModel venga creata correttamente.	S
TUF93	Si verifica che createMoneyBox di PickAmountViewModel sia corretta.	S
TUF94	Si verifica che canRedirect di PickAmountViewModel ritorni false.	S
TUF95	Si verifica che isBusy di PickAmountViewModel ritorni false.	S
TUF96	Si verifica che setAmount di PickAmountViewModel sia corretta.	S
TUF97	Si verifica che setId di PickAmountViewModel sia corretta.	S
TUF98	Si verifica che setInitFTM di PickAmountViewModel sia corretta.	S
TUF99	Si verifica che un'istanza di TransactionInitViewModel sia creata correttamente.	S
TUF100	Si verifica che createMoneyBox di TransactionInitViewModel sia corretta.	S
TUF101	Si verifica che setAmount di TransactionInitViewModel sia corretta.	S
TUF102	Si verifica che setId di TransactionInitViewModel sia corretta.	S
TUF103	Si verifica che getSellerAddress di TransactionInitViewModel sia settato all'indirizzo corretto.	S
TUF104	Si verifica che un'istanza di MoneyBoxBalanceViewModel sia creata correttamente.	S
TUF105	Si verifica che isBusy di MoneyBoxBalanceViewModel ritorni false.	S
TUF106	Si verifica che balanceFTM di MoneyBoxBalanceViewModel sia settato al valore di default.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF107	Si verifica che balanceWEI di MoneyBoxBalanceViewModel sia settato al valore di default.	S
TUF108	Si verifica che un'istanza di MoneyBoxCountViewModel sia creata correttamente.	S
TUF109	Si verifica che isBusy di MoneyBoxCountViewModel ritorni false.	S
TUF110	Si verifica che count di MoneyBoxCountViewModel sia impostato al valore di default.	S
TUF111	Si verifica che un'istanza di OrderBalanceViewModel sia creata correttamente.	S
TUF112	Si verifica che isBusy di OrderBalanceViewModel ritorni false.	S
TUF113	Si verifica che balanceFTM di OrderBalanceViewModel sia impostato al valore di default.	S
TUF114	Si verifica che balanceWEI di OrderBalanceViewModel sia impostato al valore di default.	S
TUF115	Si verifica che un'istanza di OrderCountViewModel sia creata correttamente.	S
TUF116	Si verifica che isBusy di OrderCountViewModel sia impostato a false.	S
TUF117	Si verifica che count di OrderCountViewModel sia impostato al valore di default.	S
TUF118	Si verifica che un'istanza di MoneyBoxDetailsViewModel sia creata correttamente.	S
TUF119	Si verifica che l'id di MoneyBoxDetailsViewModel venga impostato correttamente.	S
TUF120	Si verifica che ownerAddress di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF121	Si verifica che sellerAddress di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF122	Si verifica che ftm di MoneyBoxDetailsViewModel sia impostato al valore di default.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF123	Si verifica che wei di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF124	Si verifica che filledFtm di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF125	Si verifica che filledWei di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF126	Si verifica che ftmToFill di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF127	Si verifica che weiToFill di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF128	Si verifica che state di MoneyBoxDetailsViewModel sia impostato al valore di default.	S
TUF129	Si verifica che isPaid di MoneyBoxDetailsViewModel sia impostato a false.	S
TUF130	Si verifica che isRefunded di MoneyBoxDetailsViewModel sia impostato a false.	S
TUF131	Si verifica che isUnlocked di MoneyBoxDetailsViewModel sia impostato a false.	S
TUF132	Si verifica che un'istanza di OrderDetailsViewModel sia creata correttamente	S
TUF133	Si verifica che l'id di OrderDetailsViewModel venga impostato correttamente.	S
TUF134	Si verifica che ownerAddress di OrderDetailsViewModel sia impostato al valore di default.	S
TUF135	Si verifica che sellerAddress di OrderDetailsViewModel sia impostato al valore di default.	S
TUF136	Si verifica che ftm di OrderDetailsViewModel sia impostato al valore di default.	S
TUF137	Si verifica che wei di OrderDetailsViewModel sia impostato al valore di default.	S
TUF138	Si verifica che state di OrderDetailsViewModel sia impostato al valore di default.	S

Continua nella pagina successiva

Codice	Descrizione	Stato
TUF139	Si verifica che isPaid di OrderDetailsViewModel sia impostato a false.	S
TUF140	Si verifica che code di OrderDetailsViewModel sia impostato al valore di default.	S
TUF141	Si verifica che un'istanza di TransactionListElViewModel sia creata correttamente.	S
TUF142	Si verifica che id di TransactionListElViewModel sia impostato al valore di default.	S
TUF143	Si verifica che il tipo di ordine di TransactionListElViewModel sia corretto.	S
TUF144	Si verifica che lo stato dell'ordine di TransactionListElViewModel sia corretto.	S
TUF145	Si verifica che transaction di TransactionListElViewModel ritorni l'ordine corretto.	S

Tabella 10: Test di unità - Frontend

4.4.1.1 Tracciamento test di unità - Frontend

ID Test	Metodo
TUF01	shopchain-frontend/src/core/modules/contract/domain/__test__/OrderManager.test.ts: describe("should create an instance of OrderManager",()=>{ it("constructor with parameter", ...)
TUF02	shopchain-frontend/src/core/modules/contract/domain/__test__/OrderManager.test.ts: describe("should create an instance of OrderManager",()=>{ it("default constructor", ...)
TUF03	shopchain-frontend/src/core/modules/contract/domain/__test__/OrderManager.test.ts: describe("should get order count",()=>{it("defined instance", ...)
TUF04	shopchain-frontend/src/core/modules/contract/domain/__test__/OrderManager.test.ts: describe("should get order count",()=>{it("undefined instance", ...)
TUF05	shopchain-frontend/src/core/modules/contract/domain/__test__/OrderManager.test.ts: describe("should get contract balance",()=>{it("defined instance", ...)
TUF06	shopchain-frontend/src/core/modules/contract/domain/__test__/OrderManager.test.ts: describe("should get contract balance",()=>{it("undefined instance", ...)

Continua nella pagina successiva

ID Test	Metodo
TUF07	shopchain-frontend/src/core/modules/contract/domain/__test__/ MoneyBoxManager.test.ts:describe(”should create an instance of MoneyBoxManager”, ()=>{it(”constructor with parameter”, ...)
TUF08	shopchain-frontend/src/core/modules/contract/domain/__test__/ MoneyBoxManager.test.ts:describe(”should create an instance of MoneyBoxManager”, ()=>{it(”default constructor”, ...)
TUF09	shopchain-frontend/src/core/modules/contract/repo/implementations/__test__/ OrderManagerRepo.test.ts:it(”should create an instance of OrderManagerRepo”, ...)
TUF10	shopchain-frontend/src/core/modules/contract/repo/implementations/__test__/ OrderManagerRepo.test.ts:describe(”should get contract balance”,()=>{ it(”defined instance”, ...)
TUF11	shopchain-frontend/src/core/modules/contract/repo/implementations/__test__/ OrderManagerRepo.test.ts:describe(”should get contract balance”,()=>{ it(”undefined instance”, ...)
TUF12	shopchain-frontend/src/core/modules/contract/repo/implementations/__test__/ OrderManagerRepo.test.ts:describe(”should get order count”,()=>{ it(”defined instance”, ...)
TUF13	shopchain-frontend/src/core/modules/contract/repo/implementations/__test__/ OrderManagerRepo.test.ts:describe(”should get order count”,()=>{ it(”undefined instance”, ...)
TUF14	shopchain-frontend/src/core/modules/contract/repo/implementations/__test__/ MoneyBoxManagerRepo.test.ts:it(”should create an instance of MoneyBoxManagerRepo”, ...)
TUF15	shopchain-frontend/src/core/modules/contract/store/__test__/ ContractStore.test.ts:it(”should create an instance of ContractStore”, ...)
TUF16	shopchain-frontend/src/core/modules/order/domain/__test__/Amount.test.ts: it(”should create an instance of Amount”, ...)
TUF17	shopchain-frontend/src/core/modules/order/domain/__test__/Amount.test.ts: it(”should update the amount”, ...)
TUF18	shopchain-frontend/src/core/modules/order/domain/__test__/Amount.test.ts: it(”should get the FTM value correctly”, ...)
TUF19	shopchain-frontend/src/core/modules/order/domain/__test__/Amount.test.ts: it(”should get the amount properly”, ...)
TUF20	shopchain-frontend/src/core/modules/order/domain/__test__/MoneyBox.test.ts: it(”test constructor”, ...)

Continua nella pagina successiva

ID Test	Metodo
TUF21	shopchain-frontend/src/core/modules/order/domain/__test__/MoneyBox.test.ts: it("amount to fill is correct", ...)
TUF22	shopchain-frontend/src/core/modules/order/domain/__test__/MoneyBox.test.ts: it("payments", ...)
TUF23	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("test order constructor", ...)
TUF24	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("create", ...)
TUF25	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("amount", ...)
TUF26	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("unlock", ...)
TUF27	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("refund", ...)
TUF28	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("should update the order state using the setter", ...)
TUF29	shopchain-frontend/src/core/modules/order/domain/__test__/Order.test.ts: it("should update the order", ...)
TUF30	shopchain-frontend/src/core/modules/order/domain/__test__/OrderCollection.test.ts: it("test addOrder", ...)
TUF31	shopchain-frontend/src/core/modules/order/domain/__test__/OrderCollection.test.ts: it("test getById", ...)
TUF32	shopchain-frontend/src/core/modules/order/domain/__test__/OrderCollection.test.ts: it("test getBySeller", ...)
TUF33	shopchain-frontend/src/core/modules/order/domain/__test__/OrderCollection.test.ts: it("test getByOwner", ...)
TUF34	shopchain-frontend/src/core/modules/order/domain/__test__/OrderCollection.test.ts: it("test update", ...)
TUF35	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("test OrderState constructor", ...)
TUF36	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("isNotCreated", ...)

Continua nella pagina successiva

ID Test	Metodo
TUF37	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("isCreated", ...)
TUF38	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("isPaid", ...)
TUF39	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("isClosed", ...)
TUF40	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("isCancelled", ...)
TUF41	shopchain-frontend/src/core/modules/order/domain/__test__/OrderState.test.ts: it("setState", ...)
TUF42	shopchain-frontend/src/core/modules/order/domain/__test__/Payment.test.ts: it("from address is correct", ...)
TUF43	shopchain-frontend/src/core/modules/order/domain/__test__/Payment.test.ts: it("amount is correct", ...)
TUF44	shopchain-frontend/src/core/modules/order/domain/__test__/Payment.test.ts: it("timestamp is correct", ...)
TUF45	shopchain-frontend/src/core/modules/order/domain/__test__/Payment.test.ts: it("created payment is correct", ...)
TUF46	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:it("should create an instance of OrderRepo", ...)
TUF47	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("createOrder",()=>{it("defined contract instance", ...)
TUF48	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("createOrder",()=>{it("undefined contract instance", ...)
TUF49	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("unlock",()=>{it("defined contract instance", ...)
TUF50	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("unlock",()=>{it("undefined contract instance", ...)
TUF51	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("refund",()=>{it("defined contract instance", ...)
TUF52	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("refund",()=>{it("undefined contract instance", ...)

Continua nella pagina successiva

ID Test	Metodo
TUF53	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("getOrderById",()=>{it("defined contract instance", ...)
TUF54	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("getOrderById",()=>{it("undefined contract instance", ...)
TUF55	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("getOrdersBySeller",()=>{it("defined contract instance", ...)
TUF56	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("getOrdersBySeller",()=>{it("undefined contract instance", ...)
TUF57	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("getOrdersByBuyer",()=>{it("defined contract instance", ...)
TUF58	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ OrderRepo.test.ts:describe("getOrdersByBuyer",()=>{it("undefined contract instance", ...)
TUF59	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:it("should create an instance of MoneyBoxOrderRepo", ...)
TUF60	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:describe("should do a newPayment",()=>{it("defined contract instance", ...)
TUF61	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:describe("should do a newPayment",()=>{it("undefined contract instance", ...)
TUF62	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:describe("should get payments",()=>{it("defined contract instance", ...)
TUF63	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:describe("should get payments",()=>{it("undefined contract instance", ...)
TUF64	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:describe("should get the amount to fill",()=>{it("defined contract instance", ...)
TUF65	shopchain-frontend/src/core/modules/order/repo/implementations/__test__/ MoneyBoxOrderRepo.test.ts:describe("should get the amount to fill",()=>{it("undefined contract instance", ...)

Continua nella pagina successiva

ID Test	Metodo
TUF66	shopchain-frontend/src/core/modules/order/store/implementations/__test__/ OrderStore.test.ts:it(”should create an order store”, ...)
TUF67	shopchain-frontend/src/core/modules/order/store/implementations/__test__/ OrderStore.test.ts:it(”createOrder”, ...)
TUF68	shopchain-frontend/src/application/layout/LockOverlay/LockOverlayViewModel LockOverlayViewModel.test.ts:it(”creates LockOverlayViewModel”,...)
TUF69	shopchain-frontend/src/application/layout/LockOverlay/LockOverlayViewModel LockOverlayViewModel.test.ts:it(”successful”, ()...)
TUF70	shopchain-frontend/src/application/layout/LockOverlay/LockOverlayViewModel LockOverlayViewModel.test.ts:it(”not successful”,..)
TUF71	shopchain-frontend/src/application/layout/Nav/ConnectMetamask/ ConnectMetamaskViewModel/ConnectMetamaskViewModel.test.ts:it(”should call makeAutoObservable at creation”,...)
TUF72	shopchain-frontend/src/application/layout/Nav/ConnectMetamask/ ConnectMetamaskViewModel/ConnectMetamaskViewModel.test.ts:it(”should call connect”,...)
TUF73	shopchain-frontend/src/application/layout/Nav/MetamaskError/ MetamaskErrorViewModel/MetamaskErrorViewModel.test.ts:itit(”should return severity”,...)
TUF74	shopchain-frontend/src/application/layout/Nav/MetamaskError/ MetamaskErrorViewModel/MetamaskErrorViewModel.test.ts:itit(”should return name”,...)
TUF75	shopchain-frontend/src/application/layout/Nav/MetamaskError/ MetamaskErrorViewModel/MetamaskErrorViewModel.test.ts:itit(”should return description”,...)
TUF76	shopchain-frontend/src/application/layout/Nav/NavViewModel/NavViewModel.test.ts :it(”should call makeAutoObservable at creation”,...)
TUF77	shopchain-frontend/src/application/layout/Nav/NavViewModel/NavViewModel.test.ts :it(”should return the address”,...OK)
TUF78	shopchain-frontend/src/application/layout/Nav/NavViewModel/NavViewModel.test.ts :it(”should return the address”,...FAIL)
TUF79	shopchain-frontend/src/application/pages/ECommerce/ECommerceViewModel.test.ts :it(”should create an instance”,...)

Continua nella pagina successiva

ID Test	Metodo
TUF80	shopchain-frontend/src/application/pages/ECommerce/ECommerceViewModel.test.ts :it(”should return amount”,...)
TUF81	shopchain-frontend/src/application/pages/ECommerce/ECommerceViewModel.test.ts :it(”should return wei”,...)
TUF82	shopchain-frontend/src/application/pages/ECommerce/ECommerceViewModel.test.ts :it(”should return id”,...)
TUF83	shopchain-frontend/src/application/pages/ECommerce/ECommerceViewModel.test.ts :it(”should call setAmount”,...)
TUF84	shopchain-frontend/src/application/pages/ECommerce/ECommerceViewModel.test.ts :it(”should call handleSubmit”,...)
TUF85	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”should create an instance”,...)
TUF86	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”should call createOrder”,...)
TUF87	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”should return canRedirect”,...)
TUF88	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”should return isBusy”,...)
TUF89	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”should return isFailed”,...)
TUF90	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”setAmount”,...)
TUF91	shopchain-frontend/src/application/pages/TransactionInit/Choice/ ChoiceViewModel.test.ts:it(”setId”,...)
TUF92	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should create an instance”,...)
TUF93	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should call createMoneyBox”,...)
TUF94	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should return canRedirect”,...)
TUF95	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should return isBusy”,...)

Continua nella pagina successiva

ID Test	Metodo
TUF96	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should call setAmount”,...)
TUF97	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should call setId”,...)
TUF98	shopchain-frontend/src/application/pages/TransactionInit/PickAmount/ PickAmountViewModel.test.ts:it(”should call setInitFTM”,...)
TUF99	shopchain-frontend/src/application/pages/TransactionInit/ TransactionInitViewModel.test.ts:it(”should create an instance”,...)
TUF100	shopchain-frontend/src/application/pages/TransactionInit/ TransactionInitViewModel.test.ts:it(”should call createMoneyBox”,...)
TUF101	shopchain-frontend/src/application/pages/TransactionInit/ TransactionInitViewModel.test.ts:it(”should call setAmount”,...)
TUF102	shopchain-frontend/src/application/pages/TransactionInit/ TransactionInitViewModel.test.ts:it(”should call setId”,...)
TUF103	shopchain-frontend/src/application/pages/TransactionInit/ TransactionInitViewModel.test.ts:it(”should return the seller address”,...)
TUF104	shopchain-frontend/src/application/pages/Home/MoneyBoxBalance/ MoneyBoxBalanceViewModel.test.ts:it(”creates vm”,...)
TUF105	shopchain-frontend/src/application/pages/Home/MoneyBoxBalance/ MoneyBoxBalanceViewModel.test.ts:it(”isBusy”,...)
TUF106	shopchain-frontend/src/application/pages/Home/MoneyBoxBalance/ MoneyBoxBalanceViewModel.test.ts:it(”balanceFTM”,...)
TUF107	shopchain-frontend/src/application/pages/Home/MoneyBoxBalance/ MoneyBoxBalanceViewModel.test.ts:it(”balanceWEI”,...)
TUF108	shopchain-frontend/src/application/pages/Home/MoneyBoxCount/ MoneyBoxCountViewModel.test.ts:it(”creates vm”,...)
TUF109	shopchain-frontend/src/application/pages/Home/MoneyBoxCount/ MoneyBoxCountViewModel.test.ts:it(”isBusy”,...)
TUF110	shopchain-frontend/src/application/pages/Home/MoneyBoxCount/ MoneyBoxCountViewModel.test.ts:it(”count”,...)
TUF111	shopchain-frontend/src/application/pages/Home/OrderBalance/ OrderBalanceViewModel.test.ts:it(”creates vm”,...)

Continua nella pagina successiva

ID Test	Metodo
TUF112	shopchain-frontend/src/application/pages/Home/OrderBalance/ OrderBalanceViewModel.test.ts:it(”isBusy”,...)
TUF113	shopchain-frontend/src/application/pages/Home/OrderBalance/ OrderBalanceViewModel.test.ts:it(”balanceFTM”,...)
TUF114	shopchain-frontend/src/application/pages/Home/OrderBalance/ OrderBalanceViewModel.test.ts:it(”balanceWEI”,...)
TUF115	shopchain-frontend/src/application/pages/Home/OrderCount/ OrderCountViewModel.test.ts:it(”creates vm”,...)
TUF116	shopchain-frontend/src/application/pages/Home/OrderCount/ OrderCountViewModel.test.ts:it(”isBusy”,...)
TUF117	shopchain-frontend/src/application/pages/Home/OrderCount/ OrderCountViewModel.test.ts:it(”count”,...)
TUF118	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should create an instance”,...)
TUF119	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should set the id”,...)
TUF120	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get ownerAddress”,...)
TUF121	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get sellerAddress”,...)
TUF122	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get ftm”,...)
TUF123	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get wei”,...)
TUF124	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get filledFtm”,...)
TUF125	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get filledWei”,...)
TUF126	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get ftmToFill”,...)
TUF127	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get weiToFill”,...)

Continua nella pagina successiva

ID Test	Metodo
TUF128	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get state”,...)
TUF129	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get isPaid”,...)
TUF130	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get isRefunded”,...)
TUF131	shopchain-frontend/src/application/pages/MoneyBoxDetails/ MoneyBoxDetailsViewModel.test.ts:it(”should get isUnlocked”,...)
TUF132	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should create an instance”,...)
TUF133	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should set the id”,...)
TUF134	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get ownerAddress”,...)
TUF135	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get sellerAddress”,...)
TUF136	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get ftm”,...)
TUF137	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get wei”,...)
TUF138	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get state”,...)
TUF139	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get isPaid”,...)
TUF140	shopchain-frontend/src/application/pages/OrderDetails/ OrderDetailsViewModel.test.ts:it(”should get code”,...)
TUF141	shopchain-frontend/src/application/pages/TransactionIn/TransactionListEl TransactionListElViewModel.test.ts:it(”should create an instance”,...)
TUF142	shopchain-frontend/src/application/pages/TransactionIn/TransactionListEl TransactionListElViewModel.test.ts:it(”should return the order id”,...)
TUF143	shopchain-frontend/src/application/pages/TransactionIn/TransactionListEl TransactionListElViewModel.test.ts:it(”should return the correct order type”,...)

Continua nella pagina successiva

ID Test	Metodo
TUF144	shopchain-frontend/src/application/pages/TransactionIn/TransactionListEl TransactionListElViewModel.test.ts:it(”should return the correct order status”,...)
TUF145	shopchain-frontend/src/application/pages/TransactionIn/TransactionListEl TransactionListElViewModel.test.ts:it(”should return the order”,...)

Tabella 11: Tracciamento test di unità - Frontend

4.4.2 Test di unità - Solidity - OrderManager

ID Test	Descrizione	Stato
TUO01	Si verifica che il contatore degli ordini sia inizializzato a zero.	S
TUO02	Si verifica che l’ordine creato sia correttamente.	S
TUO03	Si verifica che il contratto abbia ricevuto l’ammontare corretto.	S
TUO04	Si verifica il caso fallimentare in cui l’utente tenta di inviare un importo inferiore a quello richiesto.	S
TUO05	Si verifica il caso fallimentare in cui l’utente prova a registrare un ordine che ha come venditore se stesso.	S
TUO06	Si verifica il caso fallimentare in cui l’utente prova a creare un ordine con un importo pari a zero.	S
TUO07	Si verifica il caso fallimentare in cui l’utente prova a creare un ordine con un importo negativo.	S
TUO08	Si verifica il caso fallimentare in cui l’utente prova a creare un ordine non avendo i fondi necessari.	S
TUO09	Si verifica il caso fallimentare in cui il frontend$_G$ richiede la creazione di un ordine con un ID precedentemente usato.	S
TUO10	Si verifica che le informazioni di ritorno dell’ordine siano corrette.	S
TUO11	Si verifica che lo stato dell’ordine sia modificato e settato su ”CLOSED” una volta confermata la ricezione dall’acquirente.	S
TUO12	Si verifica che il venditore abbia ricevuto i fondi correttamente una volta confermata la ricezione dall’acquirente.	S

Continua nella pagina successiva

ID Test	Descrizione	Stato
TUO13	Si verifica il caso fallimentare in cui l'ID dell'ordine da sbloccare sia errato.	S
TUO14	Si verifica il caso fallimentare in cui il codice di sblocco non coincide con quello registrato in $\text{blockchain}_{\mathbf{G}}$.	S
TUO15	Si verifica il caso fallimentare in cui l'ordine viene confermato da un utente con indirizzo differente da quello del compratore.	S
TUO16	Si verifica che lo stato dell'ordine venga settato su "CANCELLED" alla richiesta di rimborso da parte del compratore.	S
TUO17	Si verifica che lo stato dell'ordine venga settato su "CANCELLED" alla richiesta di rimborso da parte del venditore.	S
TUO18	Si verifica che i soldi siano restituiti correttamente al compratore quando è richiesto il rimborso dal compratore.	S
TUO19	Si verifica che i soldi siano restituiti correttamente al compratore quando è richiesto il rimborso dal venditore.	S
TUO20	Si verifica che non sia possibile chiedere il rimborso di un ordine chiuso.	S
TUO21	Si verifica che un utente con indirizzo diverso da quello del compratore o venditore non possa richiedere il rimborso.	S
TUO22	Si verifica che venga restituito correttamente il saldo del contratto.	S
TUO23	Si verifica che venga restituito correttamente l'indirizzo del $\text{wallet}_{\mathbf{G}}$ dell'acquirente.	S
TUO24	Si verifica che venga restituito correttamente l'indirizzo del $\text{wallet}_{\mathbf{G}}$ del venditore.	S
TUO25	Si verifica che venga restituito correttamente l'ammontare da pagare.	S
TUO26	Si verifica che venga restituito correttamente lo stato dell'ordine.	S
TUO27	Si verifica che vengano restituiti correttamente tutti i dati dell'ordine, partendo dal suo ID.	S
TUO28	Si verifica che vengano restituiti correttamente tutti gli ordini di un acquirente.	S

Continua nella pagina successiva

ID Test	Descrizione	Stato
TUO29	Si verifica che vengano restituiti correttamente tutti gli ordini di un venditore.	S

Tabella 12: Test di unità - Solidity - OrderManager

4.4.2.1 Tracciamento test di unità - Solidity - OrderManager

ID Test	Metodo
TUO01	smart-contract/test/1_OrderManager.test.js:it(”should have the initial order number to 0”, ...)
TUO02	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{it(’order created correctly’, ...)
TUO03	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{it(”the contract is filled with the correct amount”, ...)
TUO04	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{describe(”failure cases”,()=>{it(”user tries to insert an amount less than the required amount”, ...)
TUO05	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{describe(”failure cases”,()=>{it(”user tries to order his item, or send funds to itself”, ...)
TUO06	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{describe(”failure cases”,()=>{it(”user tries to pass value equal to zero”, ...)
TUO07	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{describe(”failure cases”,()=>{it(”user tries to send negative coin value”, ...)
TUO08	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{describe(”failure cases”,()=>{it(”user hasn’t enough founds”, ...)
TUO09	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{describe(”failure cases”,()=>{it(”front end require the creation of an order with the same order id”, ...)
TUO10	smart-contract/test/1_OrderManager.test.js:describe(”single payment order”, ()=>{it(’get the correct order infos’, ...)
TUO11	smart-contract/test/1_OrderManager.test.js:describe(”order confirmation”, ()=>{it(”should update state”, ...)
TUO12	smart-contract/test/1_OrderManager.test.js:describe(”order confirmation”, ()=>{it(”should release payment to seller”, ...)

Continua nella pagina successiva

ID Test	Metodo
TUO13	smart-contract/test/1_OrderManager.test.js:describe(”order confirmation”, ()=>{describe(”failure cases”,async ()=>{it(”order id isn’t correct”, ...)
TUO14	smart-contract/test/1_OrderManager.test.js:describe(”order confirmation”, ()=>{describe(”failure cases”,async ()=>{it(”order unlock code doesn’t match with unlock code saved”, ...)
TUO15	smart-contract/test/1_OrderManager.test.js:describe(”order confirmation”, ()=>{describe(”failure cases”,async ()=>{it(”order is unlock from an address different from buyer address”, ...)
TUO16	smart-contract/test/1_OrderManager.test.js:describe(”order refund”, async ()=>{describe(”should set the cancelled state”,()=>{it(”from buyer”, ...)
TUO17	smart-contract/test/1_OrderManager.test.js:describe(”order refund”, async ()=>{describe(”should set the cancelled state”,()=>{it(”from seller”, ...)
TUO18	smart-contract/test/1_OrderManager.test.js:describe(”order refund”, async ()=>{describe(”should move funds back to the buyer”,async ()=>{it(”from buyer [@skip-on-coverage]”, ...)
TUO19	smart-contract/test/1_OrderManager.test.js:describe(”order refund”, async ()=>{describe(”should move funds back to the buyer”,async ()=>{it(”from seller”, ...)
TUO20	smart-contract/test/1_OrderManager.test.js:describe(”order refund”, async ()=>{describe(”failure cases”,()=>{it(”should not refund an order already closed”, ...)
TUO21	smart-contract/test/1_OrderManager.test.js:describe(”order refund”, async ()=>{describe(”failure cases”,()=>{it(”should be the owner or the seller”, ...)
TUO22	smart-contract/test/1_OrderManager.test.js:describe(’check getter functions’, async ()=>{it(”check contractBalance()”, ...)
TUO23	smart-contract/test/1_OrderManager.test.js:describe(’check getter functions’, async ()=>{it(”check getOwnerAddress(string)”, ...)
TUO24	smart-contract/test/1_OrderManager.test.js:describe(’check getter functions’, async ()=>{it(”check getSellerAddress(string)”, ...)
TUO25	smart-contract/test/1_OrderManager.test.js:describe(’check getter functions’, async ()=>{it(”check getAmountToPay(string)”, ...)
TUO26	smart-contract/test/1_OrderManager.test.js:describe(’check getter functions’, async ()=>{it(”check getOrderState(string)”, ...)
TUO27	smart-contract/test/1_OrderManager.test.js:describe(’check getter functions’, async ()=>{it(”check getOrderById(string)”, ...)

Continua nella pagina successiva

ID Test	Metodo
TUO28	smart-contract/test/1_OrderManager.test.js:describe('check getter functions', async ()=>{it("check getOrdersByBuyer(address)", ...)
TUO29	smart-contract/test/1_OrderManager.test.js:describe('check getter functions', async ()=>{it("check getOrdersBySeller(address)", ...)

Tabella 13: Tracciamento test di unità - Solidity - OrderManager

4.4.3 Test di unità - Solidity - MoneyBox

ID Test	Descrizione	Stato
TUM01	Si verifica che il contatore degli ordini sia inizializzato a zero.	S
TUM02	Si verifica che la MoneyBox$_G$ venga creata correttamente.	S
TUM03	Si verifica che il proprietario dell'ordine possa pagare una parte dell'importo totale alla creazione della MoneyBox$_G$.	S
TUM04	Si verifica che il trasferimento di un nuovo importo nella MoneyBox$_G$ avvenga correttamente.	S
TUM05	Si verifica che il rimborso a partire dalla richiesta del proprietario della MoneyBox$_G$ avvenga correttamente.	S
TUM06	Si verifica che il rimborso a partire dalla richiesta del venditore dell'ordine avvenga correttamente.	S
TUM07	Si verifica il caso fallimentare in cui l'utente non invia l'importo stabilito.	S
TUM08	Si verifica il caso fallimentare in cui l'acquirente richieda il rimborso di una MoneyBox$_G$ chiusa.	S
TUM09	Si verifica il caso fallimentare in cui l'utente cerca di creare un nuovo pagamento con un ammontare negativo.	S
TUM10	Si verifica che vengano ritornate correttamente tutte le transazioni partecipanti alla MoneyBox$_G$.	S
TUM11	Si verifica che venga ritornato il corretto ammontare mancante per il completamento della MoneyBox$_G$.	S
TUM12	Si verifica che vengano ritornati correttamente tutti gli ordini relativi all'acquirente (registrati in entrambi i contratti).	S

Continua nella pagina successiva

ID Test	Descrizione	Stato
TUM13	Si verifica che vengano ritornati correttamente tutti gli ordini relativi al venditore (registrati in entrambi i contratti).	S
TUM14	Si verifica che vengano ritornati correttamente tutti gli ordini $\text{MoneyBox}_{\mathbf{G}}$ relativi all'indirizzo di un partecipante.	S

Tabella 14: Test di unità - Solidity - MoneyBox

4.4.3.1 Tracciamento test di unità - Solidity - OrderManager

ID Test	Metodo
TUM01	smart-contract/test/2_MoneyBox.test.js:it("should have the initial order number to 0", ...)
TUM02	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{it('moneybox created correctly', ...)
TUM03	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{it("user sends import at moneybox creation", ...)
TUM04	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{it("new fee transfer into moneybox", ...)
TUM05	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{it("refund all fee transfers from moneybox owner", ...)
TUM06	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{it("refund all fee transfers from moneybox seller", ...)
TUM07	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{describe("failure cases",()=>{it("should not have insufficient value", ...)
TUM08	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{describe("failure cases",()=>{it("buyer tries to call refund from a closed moneybox", ...)
TUM09	smart-contract/test/2_MoneyBox.test.js:describe("moneybox management", ()=>{describe("failure cases",()=>{it("negative fee payment", ...)
TUM10	smart-contract/test/2_MoneyBox.test.js:describe("check getter functions", ()=>{it("check getMoneyBoxPayments(string)", ...)
TUM11	smart-contract/test/2_MoneyBox.test.js:describe("check getter functions", ()=>{it("check getAmountToFill(string)", ...)

Continua nella pagina successiva

ID Test	Metodo
TUM12	smart-contract/test/2_MoneyBox.test.js:describe("check getter functions", ()=>{it("check getAllBuyerOrders(supercontract, _buyerAddress)", ...)
TUM13	smart-contract/test/2_MoneyBox.test.js:describe("check getter functions", ()=>{it("check getAllSellerOrders(OrderManager, address)", ...)
TUM14	smart-contract/test/2_MoneyBox.test.js:describe("check getter functions", ()=>{it("check getMoneyBoxesByParticipantAddress(address)", ...)

Tabella 15: Tracciamento test di unità - Solidity - MoneyBox

A Resoconto attività di verifica

A.1 Verifica dei documenti

A.1.1 Indice di Gulpease

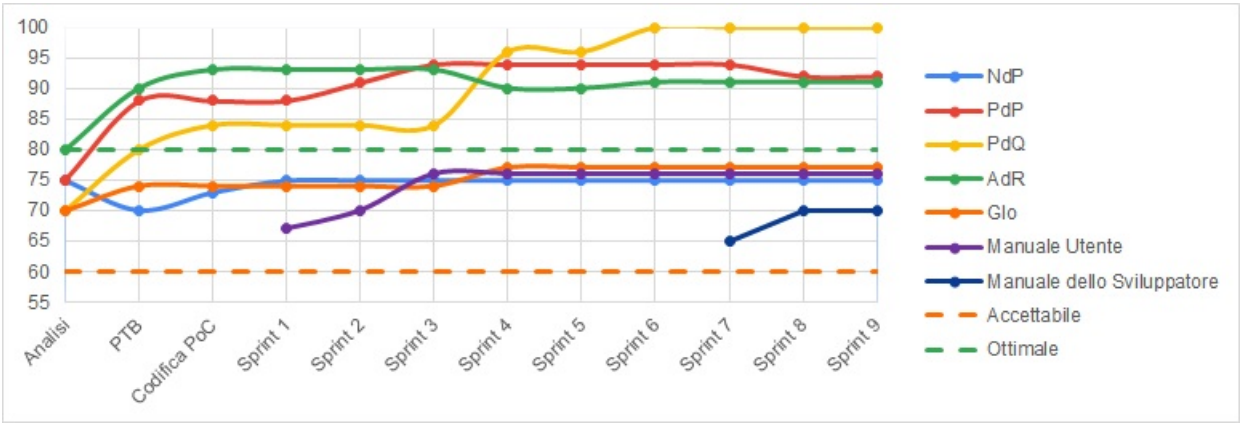

Figura 1: Indice di Gulpease di ciascun documento per periodo

A.1.2 Errori ortografici

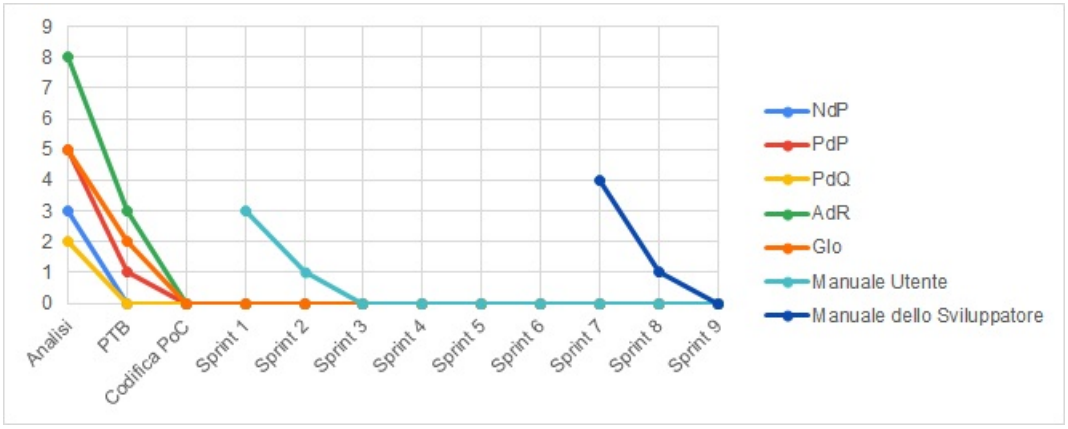

Figura 2: Errori ortografici individuati per periodo

A.2 Verifica dei processi

A.2.1 Estimated At Completion

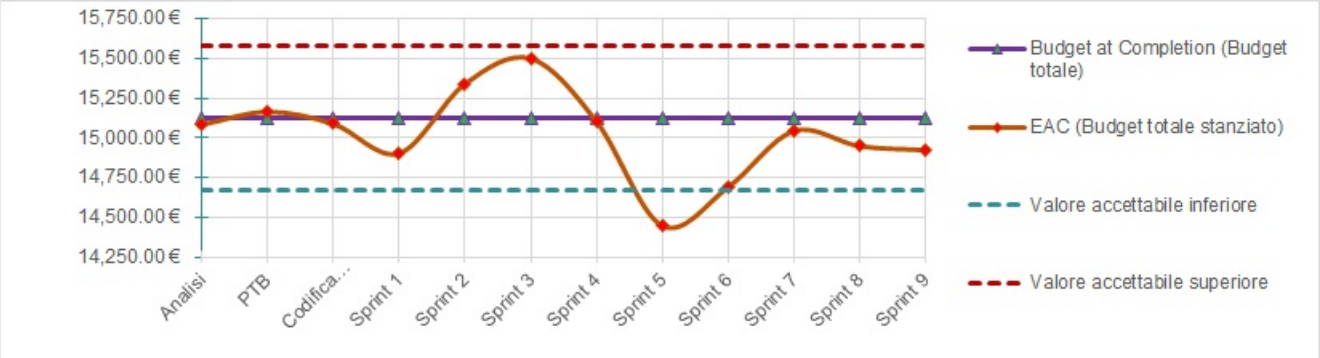

Figura 3: Revisione del valore stimato per la realizzazione del progetto

A.2.2 Earned Value & Planned Value

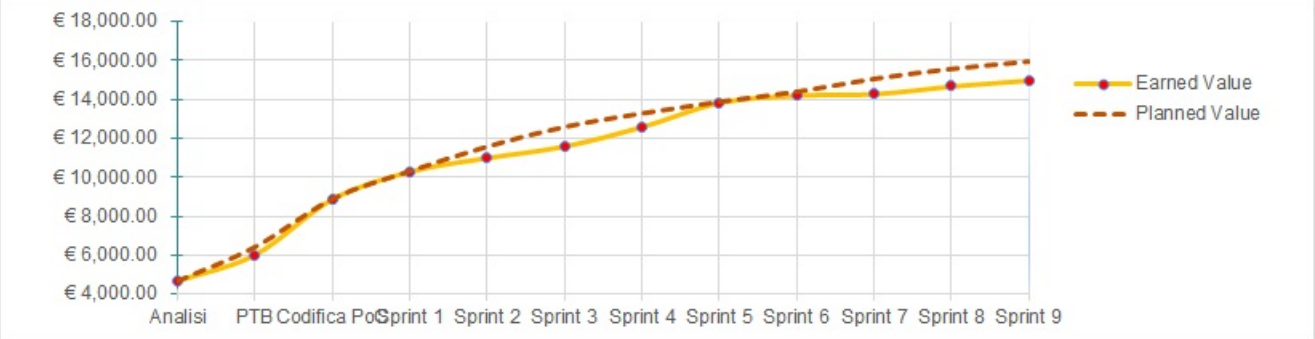

Figura 4: Valore delle attività realizzate e costo pianificato per realizzare le rimanenti

A.2.3 Actual Cost & Estimate To Complete

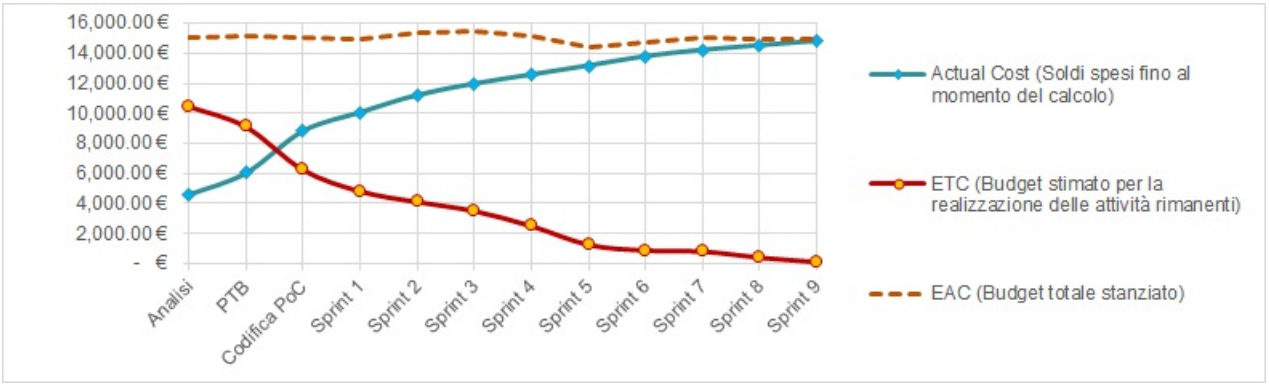

Figura 5: Costo effettivamente sostenuto e valore stimato per la realizzazione delle rimanenti attività

A.2.4 Schedule Variance & Cost Variance

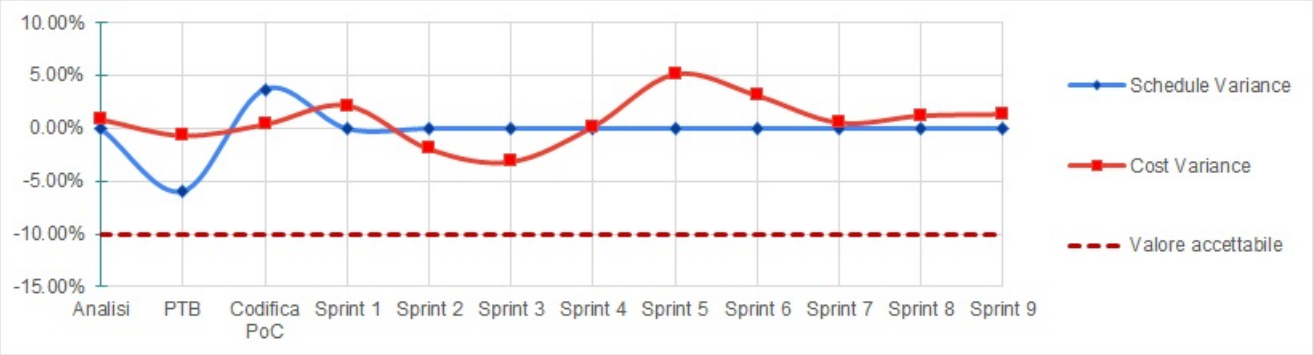

Figura 6: Cost Variance e Schedule Variance per periodo

A.2.5 Requirements Stability Index

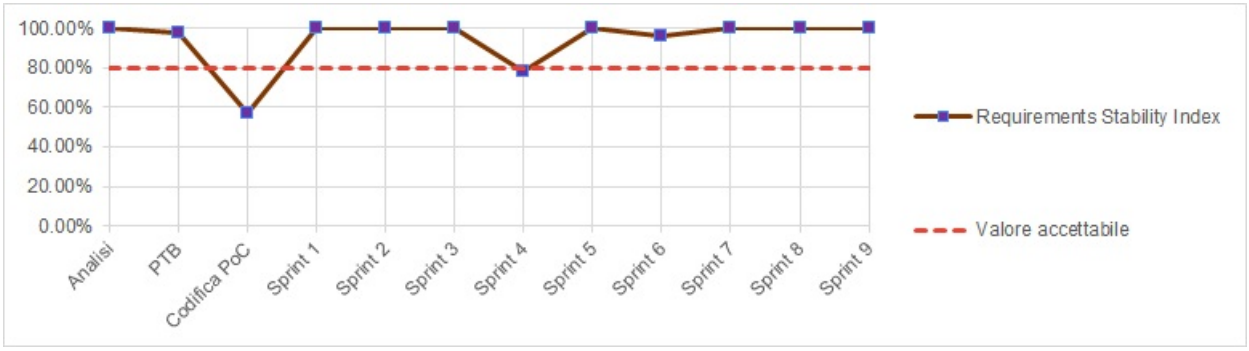

Figura 7: Variazione del numero di requisiti

A.2.6 Attualizzazione dei rischi

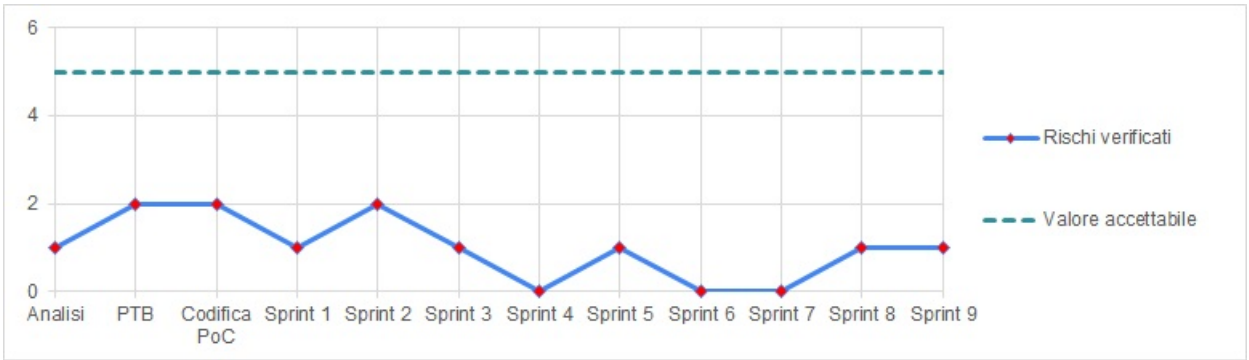

Figura 8: Rischi verificati per periodo

A.2.7 Metrics Satisfied

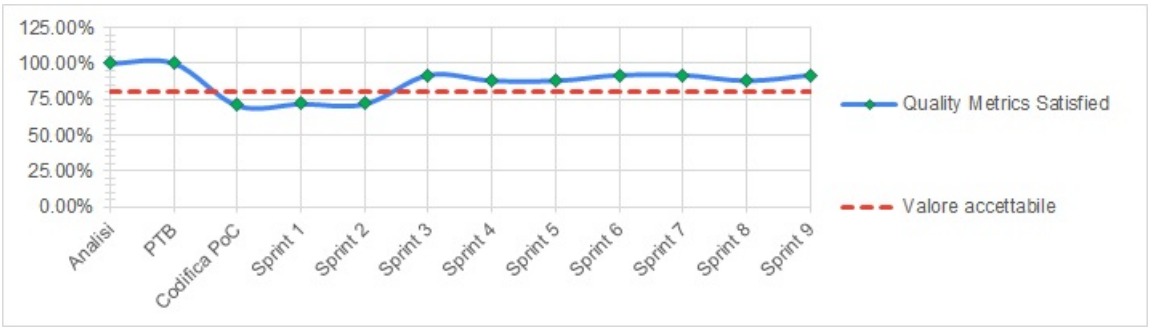

Figura 9: Percentuale di metriche soddisfatte per periodo

A.2.8 Passed Test

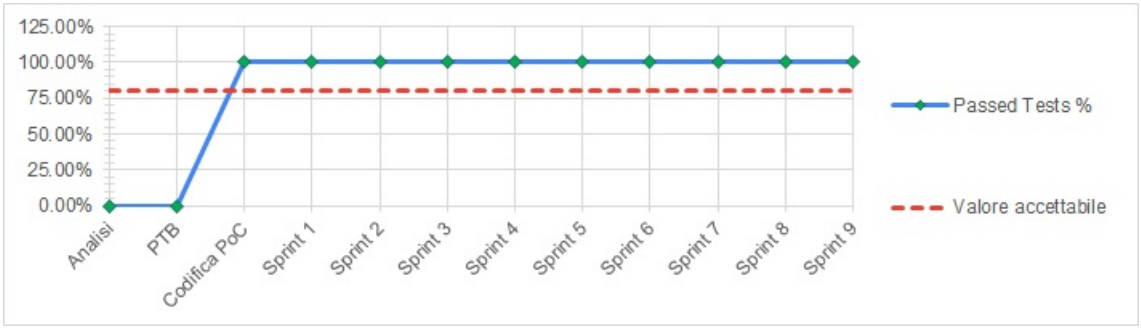

Figura 10: Percentuale di test passati rispetto a quelli implementati per periodo

A.3 Verifica dei software

A.3.1 Facilità di utilizzo

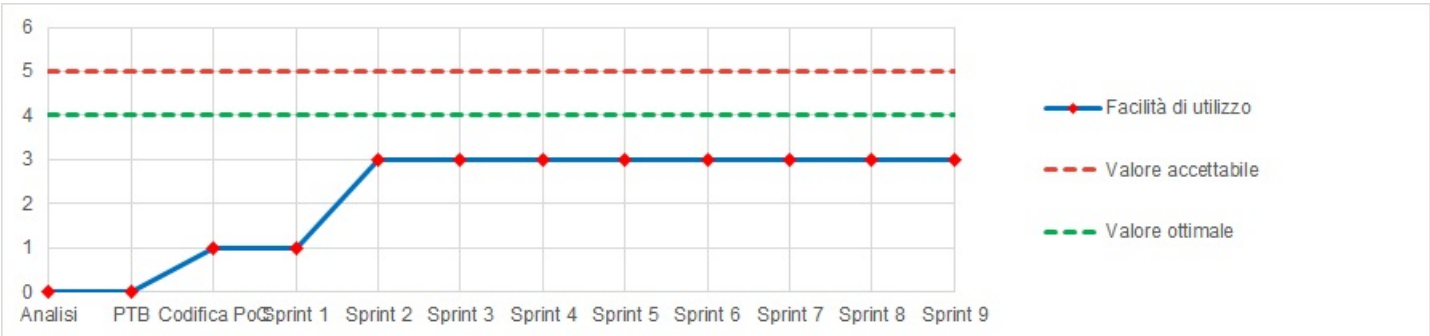

Figura 11: Quantitativo di click necessari per raggiungere la funzione desiderata

A.3.2 Versioni browser supportate

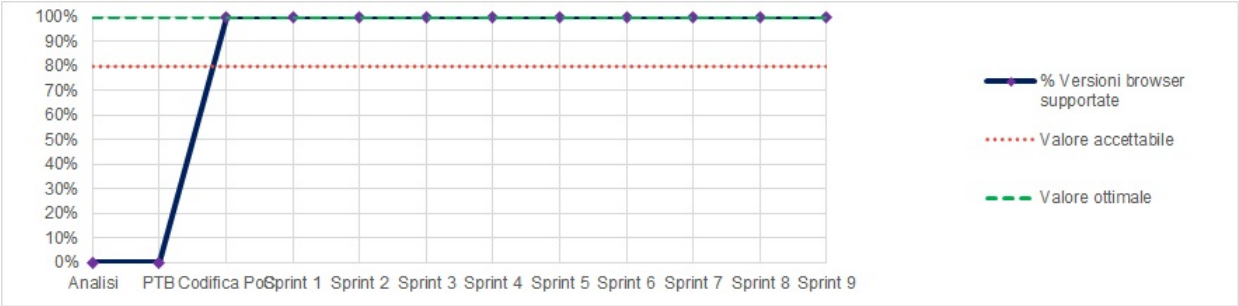

Figura 12: Versioni di browser supportate per periodo

A.3.3 Copertura requisiti obbligatori

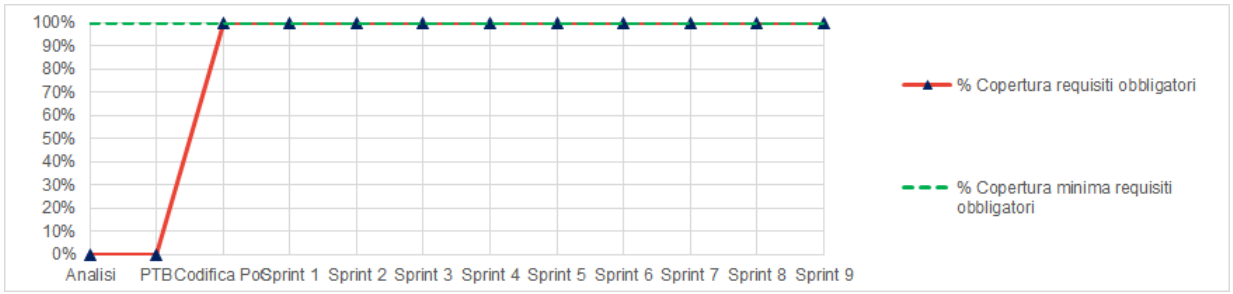

Figura 13: Percentuale di requisiti obbligatori implementati per periodo

A.3.4 Copertura requisiti desiderabili

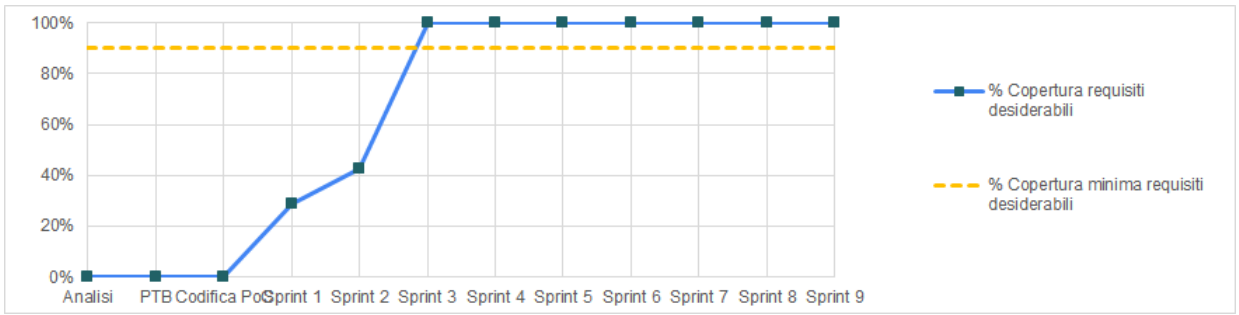

Figura 14: Percentuale di requisiti desiderabili implementati per periodo

A.3.5 Copertura requisiti opzionali

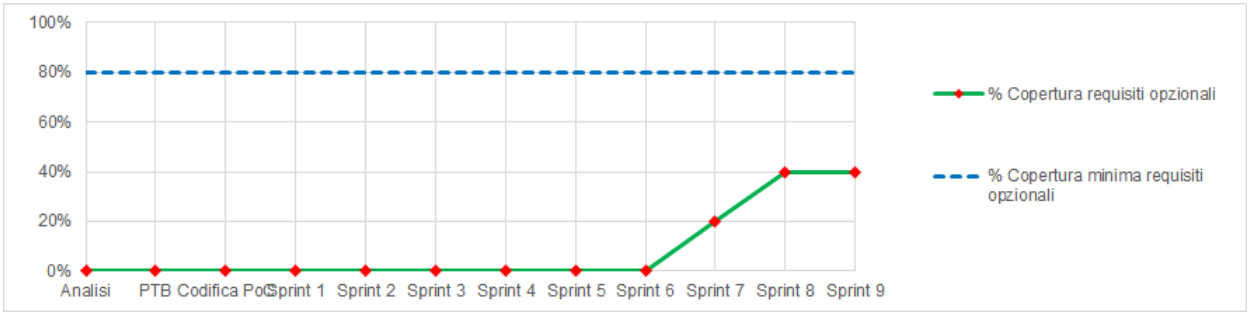

Figura 15: Percentuale di requisiti opzionali implementati per periodo

A.3.6 Solidity Statement Coverage

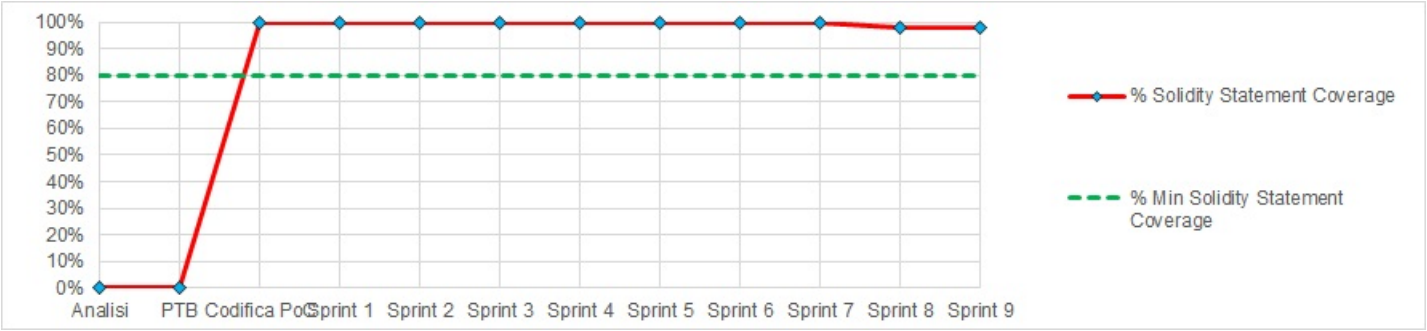

Figura 16: Statement Coverage riguardante il codice Solidity

A.3.7 Solidity Branch Coverage

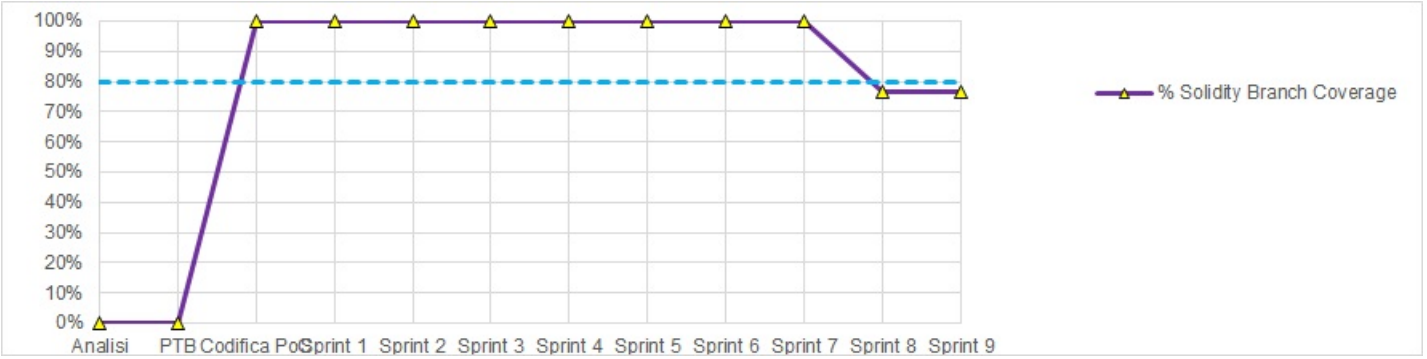

Figura 17: Branch Coverage riguardante il codice Solidity

A.3.8 Solidity Function Coverage

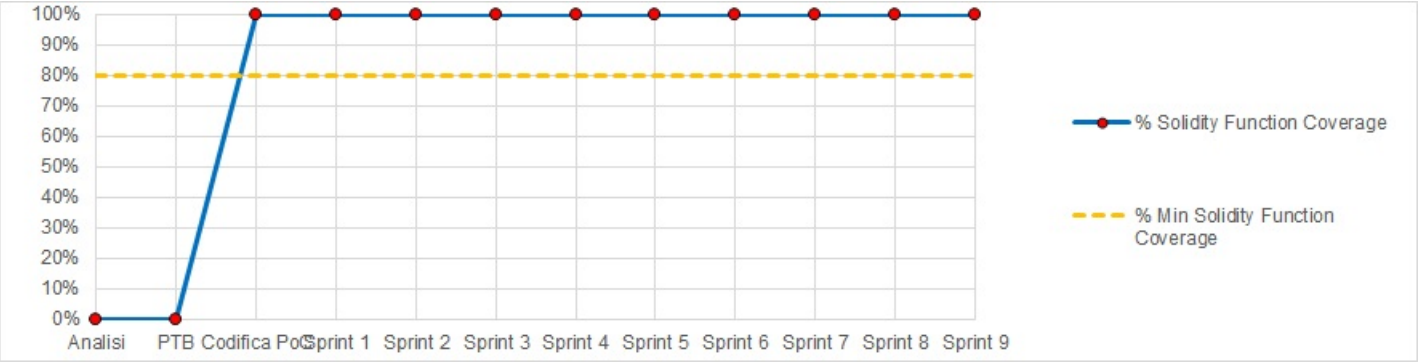

Figura 18: Function Coverage riguardante il codice Solidity

A.3.9 Solidity Line Coverage

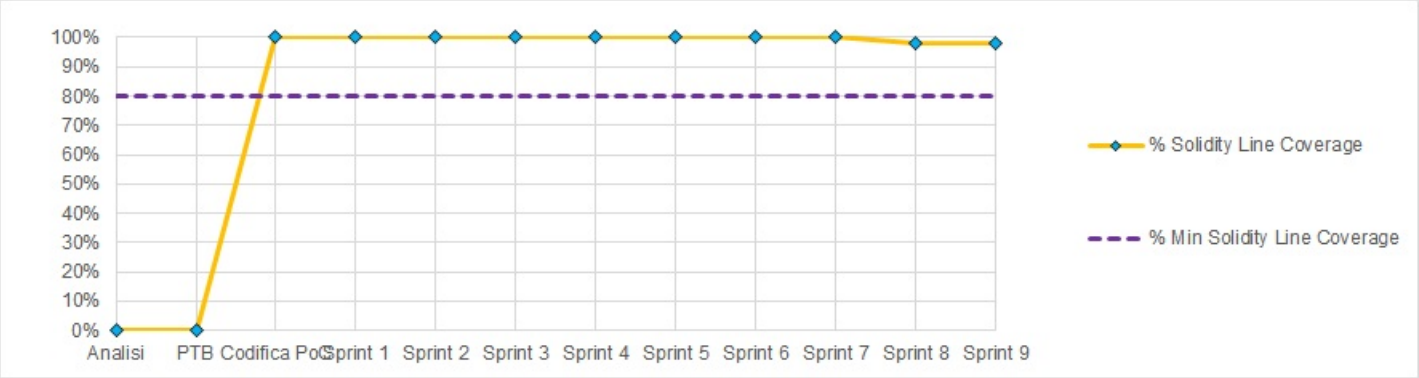

Figura 19: Line Coverage riguardante il codice Solidity

A.3.10 Frontend Statement Coverage

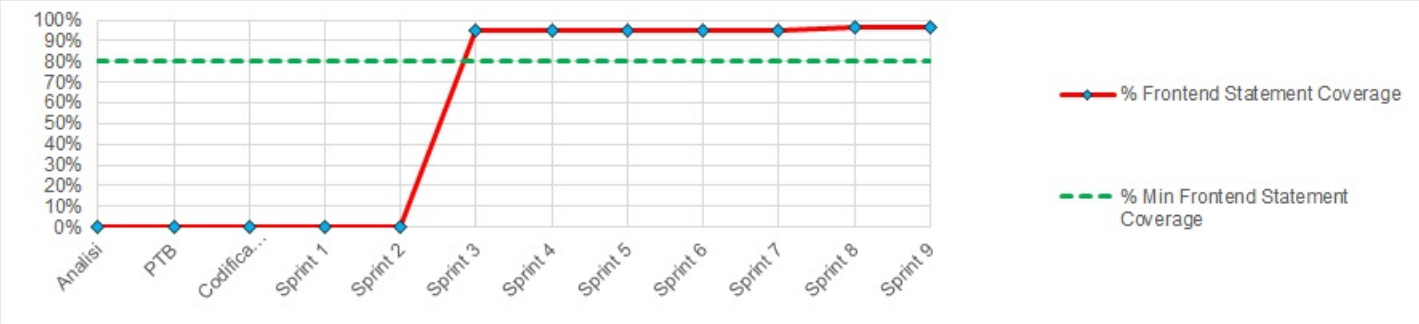

Figura 20: Statement Coverage riguardante il codice del frontend

A.3.11 Frontend Branch Coverage

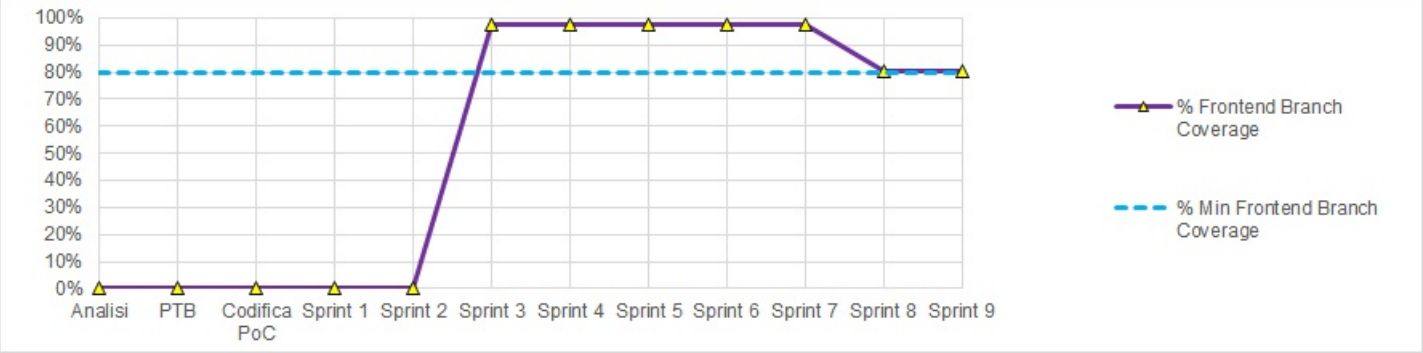

Figura 21: Branch Coverage riguardante il codice del frontend

A.3.12 Frontend Function Coverage

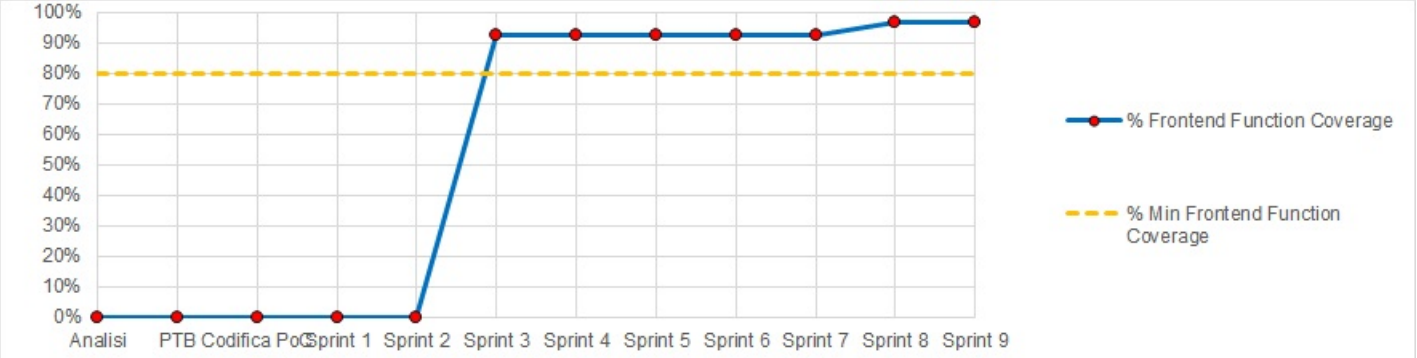

Figura 22: Function Coverage riguardante il codice del frontend

A.3.13 Frontend Line Coverage

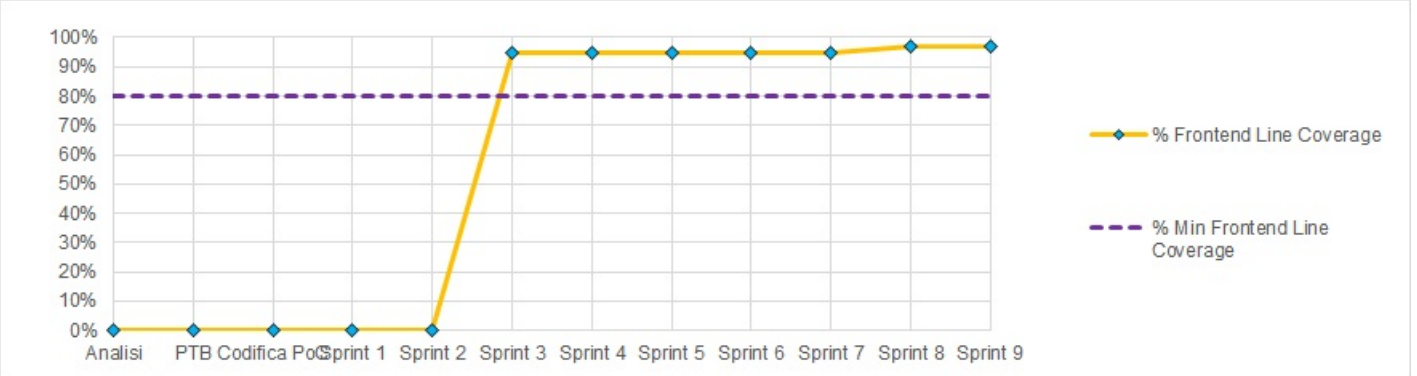

Figura 23: Line Coverage riguardante il codice del frontend