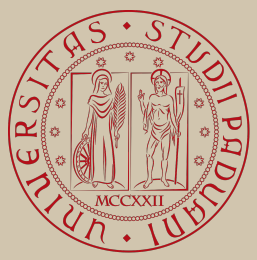
Piano di Qualifica

2025-04-15

V2.0.0

sweetenteam@gmail.com

https://sweetenteam.github.io



Destinatari	Prof. Tullio Vardanega Prof. Riccardo Cardin AzzurroDigitale
Redattori	Davide Benedetti Matteo Campagnaro Andrea Santi
Verificatori	Davide Benedetti Matteo Campagnaro Orlando Ferazzani

Registro delle modifiche

Versione	Data	Autori	Verificatori	Dettaglio
2.0.0	2025-04-15	Matteo Campagnaro	Orlando Ferazzani	Approvazione per PB
1.3.0	2025-04-11	Matteo Campagnaro	Orlando Ferazzani	Stesura sezione «Test di integrazione»
1.2.0	2025-04-11	Matteo Campagnaro	Orlando Ferazzani	Stesura sezione «Test di unità»
1.1.0	2025-04-10	Matteo Campagnaro	Orlando Ferazzani	Stesura sezione «Cruscotto di valutazione della qualità»
1.0.0	2025-02-10	Andrea Santi	Matteo Campagnaro	Approvazione per RTB
0.1.0	2025-02-08	Matteo Campagnaro	Davide Benedetti	Stesura sezioni «Cruscotto di valutazione della qualità» e «Valutazione per il miglioramento»
0.0.3	2025-02-01	Andrea Santi	Davide Benedetti	Aggiunti test di sistema e accettazione
0.0.2	2024-12-28	Andrea Santi	Matteo Campagnaro	Stesura sezione «Strategia di testing»
0.0.1	2024-12-07	Davide Benedetti	Orlando Ferazzani	Stesura introduzione e obiettivi di qualità.

Indice

Lista della immagini

Lista delle tabelle

1) Introduzione

1.1) Scopo del documento

Questo documento presenta una panoramica dettagliata delle strategie di verifica e validazione adottate per garantire la qualità del prodotto e dei processi coinvolti nel progetto. Data la natura dinamica e incrementale del documento, i contenuti saranno ampliati e modificati nel tempo per riflettere l'evoluzione del progetto e adattarsi alle esigenze mutevoli.

Il Piano di Qualifica illustra le pratiche di controllo della qualità degli artefatti e dei processi, con particolare attenzione alle metriche di valutazione del prodotto. Saranno inoltre riportati i risultati delle verifiche effettuate, con l'obiettivo di individuare e correggere tempestivamente eventuali problematiche riscontrate.

L'approccio adottato mira a promuovere il miglioramento continuo attraverso misure quantitative che permettano di monitorare e valutare il progresso del progetto. Questo impegno costante per la qualità si traduce in aggiornamenti regolari del documento, garantendo così la crescita e l'evoluzione sia del prodotto che dei processi nel tempo.

1.2) Scopo del progetto

Lo scopo del progetto è quello di sviluppare un assistente virtuale intelligente in grado di centralizzare e ottimizzare l'accesso alle informazioni aziendali. Grazie all'integrazione con piattaforme come GitHub, Confluence e Jira, BuddyBot fornisce risposte precise e personalizzate alle richieste degli utenti attraverso una chat in linguaggio naturale. Questo strumento mira a ridurre le inefficienze operative, migliorare la produttività e supportare il processo di ***onboarding***$_G$, facilitando la condivisione e il trasferimento delle conoscenze all'interno dei team.

1.3) Glossario

Per evitare ambiguità o incomprensioni riguardanti la terminologia utilizzata nei documenti, è stato redatto un Glossario che raccoglie le definizioni dei termini specifici del dominio d'uso. Ogni termine incluso nel Glossario è accompagnato dal relativo significato, al fine di garantire chiarezza e uniformità nella comprensione del testo.

La presenza di un termine nel Glossario viene segnalata direttamente nel documento adottando uno *questo stile*$_G$. L'inserimento di un termine nel Glossario è considerato completo solo dopo averne fornito una definizione chiara e accurata, contribuendo così alla coerenza del linguaggio e alla comprensione condivisa tra tutti i lettori del documento.

1.4) Riferimenti

1.4.1) Normativi

- `Norme di Progetto v1.0.0`
- Documentazione e presentazione del capitolato d'appalto C9: BuddyBot
  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C9.pdf (Ultimo accesso: 2025-04-14)
  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C9p.pdf (Ultimo accesso: 2025-04-14)
- Regolamento del progetto didattico:
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/PD1.pdf (Ultimo accesso: 2025-04-14)

1.4.2) Informativi

- `ISO/EIC 9126`
  https://en.wikipedia.org/wiki/ISO/IEC_9126 (Ultimo accesso: 2025-04-14)
- T7 - Qualità del software
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T07.pdf (Ultimo accesso: 2025-04-14)
- T8 - Qualità di processo
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T08.pdf (Ultimo accesso: 2025-04-14)
- T9 - Verifica e validazione: introduzione
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T09.pdf (Ultimo accesso: 2025-04-14)
- T10 - Verifica e validazione: analisi statica
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T10.pdf (Ultimo accesso: 2025-04-14)
- T11 - Verifica e validazione: analisi dinamica aka testing
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T11.pdf (Ultimo accesso: 2025-04-14)
- `Glossario v2.0.0`:
  https://sweetenteam.github.io/docs/PB/Documentazione_Interna/Glossario (Ultimo accesso: 2025-04-14)

2) Obiettivi di qualità

Ogni *processo*$_G$ viene valutato mediante l'applicazione di metriche specifiche, la cui definizione è dettagliata nelle sezioni Metriche di qualità del processo e Metriche di qualità del prodotto del documento Norme di Progetto v1.0.0. Queste sezioni delineano i criteri necessari affinchè le metriche siano considerate accettabili o eccellenti.

2.1) Qualità di *processo*$_G$

La qualità di processo rappresenta un'esigenza primaria nello sviluppo software, poiché per ottenere un prodotto finale di alta qualità è indispensabile partire da un'applicazione rigorosa di *best practice*$_G$ ben definite. Queste devono guidare tutte le attività, pratiche e metodologie adottate lungo l'intero ciclo di vita del software. La qualità di processo si fonda sull'idea che il raggiungimento di standard elevati nel prodotto dipenda da controlli regolari e dall'ottimizzazione continua dei processi che lo supportano, garantendo risultati che rispondano pienamente alle aspettative. **2.1.1) Processi primari**

2.1.1.1) Fornitura

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRC-EV	Earned Value	≥ 0	≤ EAC
M-PRC-PV	Planned Value	≥ 0	≤ *Budget at completion*$_G$
M-PRC-AC	Actual Cost	≥ 0	≤ EAC
M-PRC-CV	Cost Variance	≥ −7.5%	≥ 0%
M-PRC-SV	Schedule Variance	≥ −7.5%	≥ 0%
M-PRC-EAC	Estimated at Completion	Errore del ± 3% rispetto al BAC	Esattamente pari al BAC
M-PRC-ETC	Estimate to Complete	≥ 0	≤ EAC

Tabella 1: Valori ideali e accettabili per ciascuna metrica relativa al processo di fornitura.

2.1.1.2) Sviluppo

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRC-RSI	Requirements Stability Index	≥ 75%	100%
M-PRC-SFI	Structural Fan In	-	Va massimizzato
M-PRC-SFO	Structural Fan-Out	-	Va minimizzato

Tabella 2: Valori ideali e accettabili per ciascuna metrica relativa al processo di sviluppo.

2.1.2) Processi di supporto

2.1.2.1) Documentazione

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRC-GLP	Indice Gulpease	≥ 60%	≥ 80%
M-PRC-CO	Correttezza Ortografica	0 errori	0 errori

Tabella 3: Valori ideali e accettabili per ciascuna metrica relativa al processo di documentazione.

2.1.2.2) Verifica

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRC-CC	Code Coverage	≥ 90%	100%
M-PRC-PTCP	Passed test cases percentage	100%	100%

Tabella 4: Valori ideali e accettabili per ciascuna metrica relativa al processo di verifica.

2.1.2.3) Gestione della qualità

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRC-QMS	Quality Metrics Satisfied	≥ 85%	100%

Tabella 5: Valori ideali e accettabili per ciascuna metrica relativa al processo di gestione della qualità.

2.1.3) Processi organizzativi

2.1.3.1) Gestione dei processi

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRC-NCR	Non-Calculated Risk	≤ 3	0
M-PRC-TE	Temporal Efficiency	≤ 3	≤ 1

Tabella 6: Valori ideali e accettabili per ciascuna metrica relativa al processo di gestione dei processi.

2.2) Qualità di prodotto

La qualità del prodotto si riferisce all'insieme delle caratteristiche di un'entità risultante dallo sviluppo software, che ne determinano la capacità di soddisfare sia le esigenze esplicite che implicite. In altre parole, rappresenta il grado in cui un prodotto risponde alle aspettative del cliente o agli standard prestabiliti.

Essa implica una valutazione completa del software realizzato, concentrandosi su attributi fondamentali come usabilità, funzionalità, affidabilità, manutenibilità e prestazioni generali. L'obiettivo è garantire che il software non solo soddisfi le richieste del cliente e funzioni correttamente, ma lo faccia in conformità con rigorosi standard di qualità.

2.2.1) Funzionalità

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRD-CRV	Copertura Requisiti Vincolanti	100%	100%
M-PRD-CRD	Copertura Requisiti Desiderabili	≥ 50%	100%
M-PRD-CRO	Copertura Requisiti Opzionali	≥ 0%	≥ 50%

Tabella 7: Valori ideali e accettabili per ciascuna metrica relativa alle funzionalità del prodotto.

2.2.2) Affidabilità

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRD-CC	Code Coverage	≥ 80%	100%
M-PRD-BC	*Branch*$_G$ Coverage	≥ 50%	≥ 80%
M-PRD-SC	Statement Coverage	≥ 60%	≥ 80%
M-PRD-FD	Failure Density	100%	100%

Tabella 8: Valori ideali e accettabili per ciascuna metrica relativa l'affidabilità del prodotto.

2.2.3) Usabilità

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRD-FU	Facilità di Utilizzo	≤ 4 click	≤ 2 click
M-PRD-TA	Tempo di Apprendimento	≤ 10 minuti	≤ 5 minuti

Tabella 9: Valori ideali e accettabili per ciascuna metrica relativa l'usabilità del prodotto.

2.2.4) Efficienza

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRD-UR	Utilizzo Risorse	≥ 75%	100%

Tabella 10: Valori ideali e accettabili per ciascuna metrica relativa l'efficienza del prodotto.

2.2.5) Manutenibilità

Metrica	Nome	Valore accettabile	Valore Ottimo
M-PRD-CP	Complessità Ciclomatica	≤ 10	≤ 5
M-PRD-CS	Code Smell	0	0
M-PRD-MD	Module Dependency	≤ 30%	≤ 10%

Tabella 11: Valori ideali e accettabili per ciascuna metrica relativa la manutenibilità del prodotto.

3) Strategie di testing

Questa sezione riassume ed elenca i test eseguiti sul prodotto, garantendo completezza, correttezza e coerenza. In questo modo si dimostra il soddisfacimento dei requisiti utente, specificati nel capitolato d'appalto, e dei requisiti definiti nel documento ***Analisi dei Requisiti***$_G$. Verranno effettuate le seguenti tipologie di test:

- Test di **unità**: sono il punto di partenza della strategia di testing. Vengono verificate singole unità di codice con l'obiettivo di verificare che ciascun modulo funzioni correttamente, in maniera tale che ogni unità produca risultati corretti in base ai dati di input inviati.
- Test di **integrazione**: vengono eseguiti dopo il completamento/superamento dei test di unità. Verificano l'interazione tra componenti software integrate, rilevando difetti nelle interfacce e nei flussi di controllo. L'obiettivo principale di questi test è assicurare che i dati scambiati tra le componenti siano conformi alle specifiche e che i flussi di controllo funzionino regolarmente.
- Test di **sistema**: vengono eseguiti dopo il completamento dei Test di integrazione e precedono il collaudo. Si occupano di verificare l'intero sistema come unità, valutando la conformità rispetto ai requisiti presenti nel documento «Analisi dei Requisiti». L'obiettivo è quello di identificare eventuali errori (che compromettono il corretto funzionamento del sistema) e garantire che il SW soddisfi le aspettative dell'utente.
- Test di **accettazione**: corrispondono con l'ultima fase della strategia di testing, verificano e accertano il soddisfacimento dei requisiti utente (requisiti del capitolato d'appalto). Questa fase di collaudo viene effettuata in presenza del committente.

3.1) Struttura tabelle

A partire dalla sezione successiva verranno inseriti i test svolti riepilogati in una tabella. Quest'ultima sarà composta da:

- **Codice**: un breve identificativo del test eseguito. Avranno tutti questo scheletro:

  [TIPOLOGIA]-[NUMERO]

  - «Tipologia» è rappresentato dalla lettera iniziale del tipo di test eseguito
    - **TU**: Test di unità.
    - **TI**: Test di integrazione.
    - **TS**: Test di sistema.
    - **TA**: Test di accettazione.
  - «Numero»: rappresenta l'identificativo numerico assegnato a ciascun test eseguito, indicandone la sequenza.
- **Descrizione**: breve spiegazione del test effettuato
- **Esito**: risultato del test, possono essere 3:
  - V (VERIFICATO): test completato e andato a buon fine.
  - NV (NON VERIFICATO): test completato ma non andato a buon fine.
  - NI (NON IMPLEMENTATO): non è stato predisposto alcun test per la verifica della funzionalità specifica.

3.2) Test di unità

I test di unità sono concepiti per verificare il corretto funzionamento delle singole componenti del codice. Per «`unità`» si intendono funzioni, classi o, più in generale, qualsiasi entità autonoma incaricata di svolgere compiti specifici all'interno del software. Per implementare in modo efficace questa tipologia di test, abbiamo adottato il framework di unit testing **`Jest`**, che ci ha permesso di scrivere test chiari e manutenibili. A supporto della qualità del codice, abbiamo inoltre integrato **`ESLint`**, uno strumento utile per rilevare errori sintattici e problemi stilistici, garantendo così uno standard elevato durante lo sviluppo.

Codice	Descrizione	Stato
TU-01	Verificare che i componenti Header, Navbar e ChatWindow siano presenti nel DOM quando il componente Home viene renderizzato. (frontend)	V
TU-02	Verificare che il componente Header venga renderizzato correttamente con il tema chiaro e che l'immagine venga aggiornata al caricamento. (frontend)	V
TU-03	Verificare che il componente Navbar renderizzi correttamente i link esterni per GitHub, Jira e Confluence. (frontend)	V
TU-04	Verificare che le icone di FontAwesome per GitHub, Jira e Confluence siano presenti nel componente Navbar. (frontend)	V
TU-05	Verificare che il componente Navbar contenga lo switcher del tema. (frontend)	V
TU-06	Verificare che le icone del Sole e della Luna siano correttamente visualizzate in base al tema. (frontend)	V
TU-07	Verificare che il pulsante di cambio tema modifichi correttamente il tema tra «light» e «dark». (frontend)	V
TU-08	Verificare che l'Adapter venga creato una sola volta quando il componente ChatWindow viene renderizzato. (frontend)	V
TU-09	Verificare che i componenti Chat e InputForm siano correttamente renderizzati. (frontend)	V
TU-10	Verifica che il componente InputForm si renderizzi correttamente e gestisca i cambiamenti di input. (frontend)	V
TU-11	Verifica che il messaggio non venga inviato se il testo è vuoto o contiene solo spazi. (frontend)	V
TU-12	Verifica che l'altezza del textarea venga regolata dinamicamente durante la scrittura. (frontend)	V
TU-13	Verifica che il componente Chat venga renderizzato correttamente. (frontend)	V
TU-14	Verifica lo stato di caricamento quando loadingHistory è true. (frontend)	V
TU-15	Verifica che venga chiamato loadHistory quando il pulsante «Load more» viene cliccato. (frontend)	V
TU-16	Verifica che venga mostrato un alert di errore quando errorHistory è true. (frontend)	V
TU-17	Verifica che il pulsante «Load more» venga mostrato quando ci sono più messaggi. (frontend)	V

TU-18	Verifica che il componente ChatQA venga renderizzato correttamente con domanda e risposta. (frontend)	V
TU-19	Verifica che il componente Bubble venga renderizzato correttamente per un messaggio utente. (frontend)	V
TU-20	Verifica che il componente Bubble venga renderizzato correttamente per un messaggio bot. (frontend)	V
TU-21	Verifica che il componente Bubble mostri un alert quando c'è un errore. (frontend)	V
TU-22	Verifica che il componente Bubble mostri un indicatore di caricamento quando loading è true. (frontend)	V
TU-23	Verifica il comportamento di visibilità del tooltip in modalità mobile. (frontend)	V
TU-24	Verifica che l'alert venga renderizzato correttamente con il titolo e la descrizione e che abbia le classi appropriate in base al tipo di alert. (frontend)	V
TU-25	Verifica che l'avatar venga renderizzato correttamente, mostrando l'immagine se presente o un fallback altrimenti. (frontend)	V
TU-26	Verifica che il bottone venga renderizzato con il testo corretto e con le classi appropriate per dimensione e stile. (frontend)	V
TU-27	Verifica che l'ErrorAlert mostri il messaggio corretto in base al codice di stato e che venga visualizzato un timestamp. (frontend)	V
TU-28	Verifica che venga renderizzato un elemento di caricamento (spinner). (frontend)	V
TU-29	Verifica che l'indicatore di caricamento (bouncing dots) venga renderizzato correttamente. (frontend)	V
TU-30	Verifica che il componente renderizzi correttamente il contenuto markdown, anche con contenuti vuoti. (frontend)	V
TU-31	Verifica che l'avatar dell'utente o del bot venga renderizzato correttamente con l'immagine o il fallback. (frontend)	V
TU-32	Verifica che un timestamp venga formattato correttamente. (frontend)	V
TU-33	Verifica che vengano generati ID unici e incrementali. (frontend)	V
TU-34	Verifica che LOAD_HISTORY_START imposti correttamente lo stato di caricamento. (frontend)	V
TU-35	Verifica che LOAD_HISTORY_SUCCESS aggiorni i messaggi e lo stato. (frontend)	V
TU-36	Verifica che LOAD_HISTORY_ERROR imposti correttamente l'errore. (frontend)	V
TU-37	Verifica che ADD_MESSAGE_START imposti correttamente il caricamento del messaggio. (frontend)	V
TU-38	Verifica che ADD_MESSAGE_SUCCESS aggiorni il messaggio con successo. (frontend)	V
TU-39	Verifica che ADD_MESSAGE_ERROR aggiorni l'errore nel messaggio. (frontend)	V
TU-40	Verifica che SCROLL_DOWN imposti il flag hasToScroll. (frontend)	V

TU-41	Verifica che i messaggi non vengano modificati con ID non corrispondenti in ADD_MESSAGE_SUCCESS e ADD_MESSAGE_ERROR. (frontend)	V
TU-42	Verifica che lo stato rimanga invariato per azioni sconosciute. (frontend)	V
TU-43	Verifica che il componente ChatProvider renderizzi correttamente i figli. (frontend)	V
TU-44	Verifica che il ChatProvider carichi la cronologia dei messaggi al montaggio. (frontend)	V
TU-45	Verifica che il messaggio venga inviato correttamente e che la risposta venga gestita. (frontend)	V
TU-46	Verifica che venga gestito un errore quando l'invio di un messaggio fallisce. (frontend)	V
TU-47	Verifica che venga gestito correttamente un errore nel caricamento della cronologia. (frontend)	V
TU-48	Verifica che venga lanciato un errore se useChat viene usato fuori dal ChatProvider. (frontend)	V
TU-49	Verifica che vengano caricati correttamente i messaggi più vecchi. (frontend)	V
TU-50	Verifica che i messaggi con contenuti troppo lunghi vengano marcati come errore. (frontend)	V
TU-51	Verifica che venga inviato un errore con codice specifico se si verifica un CustomError. (frontend)	V
TU-52	Verifica che il componente ThemeProvider renderizzi correttamente i figli. (frontend)	V
TU-53	Verifica del recupero e adattamento della cronologia. (frontend)	V
TU-54	Verifica del recupero e adattamento di una risposta. (frontend)	V
TU-55	Gestione dell'errore quando il recupero di una risposta fallisce. (frontend)	V
TU-56	Verifica la generazione dell'ID quando data.id è mancante. (frontend)	V
TU-57	Verifica rilancio di CustomError se lanciato da fetchHistory. (frontend)	V
TU-58	Verifica rilancio di CustomError se lanciato da fetchQuestion. (frontend)	V
TU-59	Verifica che fetchHistory e fetchQuestion recuperino correttamente la cronologia quando la risposta dell'API è positiva. (frontend)	V
TU-60	Verifica che venga lanciato un errore se la risposta dell'API non è «ok». (frontend)	V
TU-61	Verifica che venga gestito correttamente un errore di rete durante la chiamata a fetchHistory e a fetchQuestion. (frontend)	V
TU-62	Verifica che venga gestito correttamente un errore di timeout o interruzione della richiesta in fetchHistory e in fetchQuestion. (frontend)	V
TU-63	Verifica che venga lanciato un errore di tipo SERVER quando il codice di stato della risposta è 500 o superiore. (frontend)	V
TU-64	Verifica che venga lanciato un errore di tipo CONNESSIONE per errori con codice di stato tra 400 e 499. (frontend)	V
TU-65	Verificare che il controller FetchHistoryController, una volta ricevuto un messaggio sulla coda fetch_queue, richiami correttamente il metodo	V

	fetchStoricoChat del caso d'uso e restituisca una lista di oggetti ChatDTO costruita a partire dai Chat ottenuti. (chat history)	
TU-66	Verificare che il service FetchHistoryService invochi correttamente il metodo fetchStoricoChat dell'adapter con i parametri attesi e restituisca il risultato senza modificarlo. (chat history)	V
TU-67	Verificare che l'adapter FetchHistoryAdapter richiami il metodo fetchStoricoChat del repository con i parametri corretti e trasformi i dati ricevuti in oggetti del dominio Chat, mappando correttamente i campi. (chat history)	V
TU-68	Verificare che il metodo fetchStoricoChat del repository recuperi correttamente lo storico delle chat dal database, gestendo sia il caso in cui l'id dell'ultima chat sia presente (con query basata su answerDate), sia il caso in cui non lo sia (recupero degli ultimi N record più recenti). (chat history)	V
TU-69	Verificare che il controller ChatConsumer, al ricevimento di un messaggio sulla coda chat_message, trasformi correttamente il CreateChatDTO in un InsertChatCmd, invochi il metodo insertChat del caso d'uso e restituisca un ChatDTO. (chat history)	V
TU-70	Verificare che il service InsertChatService invochi correttamente il metodo insertChat dell'adapter (implementazione della Port Out) con i parametri ricevuti e restituisca il risultato senza applicare ulteriori trasformazioni. (chat history)	V
TU-71	Verificare che l'adapter InsertChatAdapter invochi il metodo insertChat del repository con i parametri estratti dal InsertChatCmd e restituisca un oggetto Chat costruito correttamente a partire dalla ChatEntity restituita. (chat history)	V
TU-72	Verificare che il metodo insertChat del repository crei correttamente una nuova istanza di ChatEntity utilizzando anche la data dell'ultimo fetch delle informazioni (LastUpdateEntity) e la salvi nel database. (chat history)	V
TU-73	Verificare che il controller InsertLastUpdateController, al ricevimento di un messaggio sulla coda lastFetch_queue, costruisca correttamente un LastUpdateCmd, invochi il metodo insertLastRetrieval del service e restituisca il risultato booleano. (chat history)	V
TU-74	Verificare che il service InsertLastUpdateService invochi il metodo insertLastRetrieval dell'adapter (implementazione della Port Out) con i dati forniti e restituisca il valore booleano risultante. (chat history)	V
TU-75	Verificare che l'adapter InsertLastUpdateAdapter invochi il metodo insertLastRetrieval del repository ChatRepository passando correttamente la data di LastFetch ricevuta nel comando. (chat history)	V
TU-76	Verificare che il metodo insertLastRetrieval del repository aggiorni correttamente il record LastUpdateEntity esistente con la nuova data fornita. (chat history)	V
TU-77	Verificare che il metodo insertLastRetrieval del repository crei un nuovo record LastUpdateEntity nel caso in cui non ne esista uno. (chat history)	V
TU-78	Verificare che il controller FetchLastUpdateController, al ricevimento di un messaggio sulla coda getLastUpdate_queue, invochi il metodo fetchLa-	V

	stUpdate del service e restituisca un LastUpdateDTO contenente il valore corretto di lastFetch. (chat history)	
TU-79	Verificare che il service FetchLastUpdateService invochi correttamente il metodo fetchLastUpdate dell'adapter (implementazione della Port Out) e restituisca un oggetto LastUpdate con il valore atteso. (chat history)	V
TU-80	Verificare che l'adapter fetchLastUpdateAdapter invochi il metodo fetchLastUpdate del repository ChatRepository e trasformi correttamente la LastUpdateEntity in un oggetto LastUpdate del dominio. (chat history)	V
TU-81	Verificare che il metodo fetchLastUpdate del repository recuperi correttamente il record LastUpdateEntity con id 1 dal database e lo restituisca. (chat history)	V
TU-82	Verificare che il metodo insertChat del repository lanci un errore se il record LastUpdateEntity non è presente nel database. (chat history)	V
TU-83	Verificare che il metodo fetchStoricoChat del repository lanci un errore se l'id della chat specificata non corrisponde a nessun record. (chat history)	V
TU-84	Verificare che il metodo insertLastRetrieval del repository restituisca false nel caso si verifichi un errore durante l'operazione di aggiornamento o inserimento. (chat history)	V
TU-85	Verificare che il metodo fetchLastUpdate del repository lanci un errore se il record LastUpdateEntity non è presente nel database. (chat history)	V
TU-86	Verificare che il controller recuperi e memorizzi correttamente le informazioni di Jira con boardId e lastUpdate (JiraFetchAndStoreController.spec.ts). (information vector db)	V
TU-87	Verificare che il controller recuperi e memorizzi le informazioni di Jira senza lastUpdate (JiraFetchAndStoreController.spec.ts). (information vector db)	V
TU-88	Verificare che il controller gestisca gli errori e restituisca un risultato di errore (JiraFetchAndStoreController.spec.ts). (information vector db)	V
TU-89	Verificare che il controller recuperi correttamente le informazioni (RetrievalController.spec.ts). (information vector db)	V
TU-90	Verificare che il controller gestisca gli errori durante il recupero delle informazioni (RetrievalController.spec.ts). (information vector db)	V
TU-91	Verificare che il controller recuperi e memorizzi correttamente le informazioni di GitHub con la lista dei repository e lastUpdate (GithubFetchAndStoreController.spec.ts). (information vector db)	V
TU-92	Verificare che il controller recuperi e memorizzi le informazioni di GitHub senza lastUpdate (GithubFetchAndStoreController.spec.ts). (information vector db)	V
TU-93	Verificare che il controller gestisca gli errori e restituisca un risultato di errore (GithubFetchAndStoreController.spec.ts). (information vector db)	V
TU-94	Verificare che il controller recuperi e memorizzi correttamente le informazioni di Confluence con lastUpdate (ConfluenceFetchAndStoreController.spec.ts). (information vector db)	V

TU-95	Verificare che il controller recuperi e memorizzi le informazioni di Confluence senza lastUpdate (ConfluenceFetchAndStoreController.spec.ts). (information vector db)	V
TU-96	Verificare che il controller gestisca gli errori e restituisca un risultato di errore (ConfluenceFetchAndStoreController.spec.ts). (information vector db)	V
TU-97	Verificare che il servizio recuperi e memorizzi correttamente i documenti di Confluence (ConfluenceService.spec.ts). (information vector db)	V
TU-98	Verificare che il servizio gestisca gli errori durante il recupero dei documenti dall'API di Confluence (ConfluenceService.spec.ts). (information vector db)	V
TU-99	Verificare che il servizio gestisca gli errori durante la memorizzazione dei documenti di Confluence (ConfluenceService.spec.ts). (information vector db)	V
TU-100	Verificare che il servizio recuperi correttamente le informazioni utilizzando la porta di retrieval (RetrievalService.spec.ts). (information vector db)	V
TU-101	Verificare che il servizio gestisca correttamente i risultati vuoti (RetrievalService.spec.ts). (information vector db)	V
TU-102	Verificare che il servizio recuperi e memorizzi correttamente le informazioni di GitHub (GithubService.spec.ts). (information vector db)	V
TU-103	Verificare che il servizio gestisca gli errori durante il recupero delle informazioni da GitHub (GithubService.spec.ts). (information vector db)	V
TU-104	Verificare che il servizio gestisca gli errori durante la memorizzazione delle informazioni di GitHub (GithubService.spec.ts). (information vector db)	V
TU-105	Verificare che il servizio gestisca correttamente i risultati vuoti dalle chiamate API (GithubService.spec.ts). (information vector db)	V
TU-106	Verificare che il servizio recuperi e memorizzi correttamente le informazioni di Jira (JiraService.spec.ts). (information vector db)	V
TU-107	Verificare che il servizio gestisca gli errori durante il recupero delle informazioni dall'API di Jira (JiraService.spec.ts). (information vector db)	V
TU-108	Verificare che il servizio gestisca gli errori durante la memorizzazione delle informazioni di Jira (JiraService.spec.ts). (information vector db)	V
TU-109	Verificare che l'adattatore memorizzi correttamente più ticket (JiraStoreAdapter.spec.ts). (information vector db)	V
TU-110	Verificare che l'adattatore gestisca correttamente un array di ticket vuoto (JiraStoreAdapter.spec.ts). (information vector db)	V
TU-111	Verificare che l'adattatore fallisca e ritorni immediatamente se il salvataggio di un ticket fallisce (JiraStoreAdapter.spec.ts). (information vector db)	V
TU-112	Verificare che l'adattatore gestisca gli errori lanciati dal repository (JiraStoreAdapter.spec.ts). (information vector db)	V
TU-113	Verificare che i ticket vengano recuperati e trasformati correttamente (JiraAPIAdapter.spec.ts). (information vector db)	V
TU-114	Verificare che i campi mancanti vengano gestiti correttamente (JiraAPIAdapter.spec.ts). (information vector db)	V

TU-115	Verificare che i giorni trascorsi vengano estratti correttamente da lastUpdate (JiraAPIAdapter.spec.ts). (information vector db)	V
TU-116	Verificare che gli errori dell'API vengano gestiti correttamente (JiraAPIAdapter.spec.ts). (information vector db)	V
TU-117	Verificare che la lista di issue vuota venga gestita correttamente (JiraAPIAdapter.spec.ts). (information vector db)	V
TU-118	Verificare che il testo venga estratto correttamente dal contenuto ADF (JiraAPIAdapter.spec.ts). (information vector db)	V
TU-119	Verificare che le issue vengano recuperate con il JQL corretto quando viene fornito daysBack (JiraAPIRepository.spec.ts). (information vector db)	V
TU-120	Verificare che tutte le issue vengano recuperate quando daysBack non è fornito (JiraAPIRepository.spec.ts). (information vector db)	V
TU-121	Verificare che la paginazione venga gestita correttamente (JiraAPIRepository.spec.ts). (information vector db)	V
TU-122	Verificare che la risposta vuota venga gestita correttamente (JiraAPIRepository.spec.ts). (information vector db)	V
TU-123	Verificare che gli errori dell'API vengano gestiti correttamente (JiraAPIRepository.spec.ts). (information vector db)	V
TU-124	Verificare che i documenti vengano recuperati e trasformati correttamente (ConfluenceAPIAdapter.spec.ts). (information vector db)	V
TU-125	Verificare che i dati mancanti del documento vengano gestiti correttamente (ConfluenceAPIAdapter.spec.ts). (information vector db)	V
TU-126	Verificare che gli errori dell'API vengano gestiti correttamente (ConfluenceAPIAdapter.spec.ts). (information vector db)	V
TU-127	Verificare che le pagine vengano recuperate con l'intervallo di tempo predefinito (ConfluenceAPIRepository.spec.ts). (information vector db)	V
TU-128	Verificare che le pagine vengano recuperate con un intervallo di tempo specificato (ConfluenceAPIRepository.spec.ts). (information vector db)	V
TU-129	Verificare che la paginazione venga gestita correttamente (ConfluenceAPIRepository.spec.ts). (information vector db)	V
TU-130	Verificare che gli errori di rete vengano gestiti correttamente (ConfluenceAPIRepository.spec.ts). (information vector db)	V
TU-131	Verificare che le risposte non OK vengano gestite correttamente (errore a livello di API) (ConfluenceAPIRepository.spec.ts). (information vector db)	V
TU-132	Verificare che tutti i documenti vengano memorizzati con successo (ConfluenceStoreAdapter.spec.ts). (information vector db)	V
TU-133	Verificare che l'array di documenti vuoto venga gestito correttamente (ConfluenceStoreAdapter.spec.ts). (information vector db)	V
TU-134	Verificare che gli errori di memorizzazione per i singoli documenti vengano gestiti correttamente (ConfluenceStoreAdapter.spec.ts). (information vector db)	V
TU-135	Verificare che gli errori del repository vengano gestiti correttamente (ConfluenceStoreAdapter.spec.ts). (information vector db)	V

TU-136	Verifica che fetchGithubCommitsInfo recuperi e trasformi correttamente i commit (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-137	Verifica che fetchGithubCommitsInfo gestisca più repository (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-138	Verifica che fetchGithubFilesInfo recuperi e trasformi correttamente i file (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-139	Verifica che fetchGithubFilesInfo gestisca file di grandi dimensioni utilizzando contenuti grezzi (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-140	Verifica che fetchGithubFilesInfo salti i file binari (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-141	Verifica che fetchGithubFilesInfo gestisca gli errori di file non trovato in modo corretto (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-142	Verifica che fetchGithubPullRequestsInfo recuperi e trasformi correttamente le pull request (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-143	Verifica che fetchGithubRepositoryInfo recuperi e trasformi correttamente le informazioni del repository (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-144	Verifica che fetchGithubRepositoryInfo filtri i repository in base alla data di ultimo aggiornamento (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-145	Verifica che fetchGithubWorkflowInfo recuperi e trasformi correttamente le informazioni del workflow (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-146	Verifica che fetchGithubWorkflowRuns recuperi e trasformi correttamente le esecuzioni del workflow (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-147	Verifica che fetchGithubWorkflowRuns gestisca gli errori durante il recupero delle esecuzioni del workflow (GithubAPIAdapter.spec.ts). (information vector db)	V
TU-148	Verifica che fetchCommitsInfo recuperi i commit con i parametri di default (GithubAPIRepository.spec.ts). (information vector db)	V
TU-149	Verifica che fetchCommitsInfo recuperi i commit con il parametro lastUpdate (GithubAPIRepository.spec.ts). (information vector db)	V
TU-150	Verifica che fetchCommitsInfo gestisca gli errori durante il recupero dei commit (GithubAPIRepository.spec.ts). (information vector db)	V
TU-151	Verifica che fetchCommitModifiedFilesInfo recuperi i file modificati per un commit (GithubAPIRepository.spec.ts). (information vector db)	V
TU-152	Verifica che fetchCommitModifiedFilesInfo gestisca gli errori durante il recupero dei file modificati (GithubAPIRepository.spec.ts). (information vector db)	V
TU-153	Verifica che fetchFileInfo recuperi le informazioni del file (GithubAPIRepository.spec.ts). (information vector db)	V

TU-154	Verifica che fetchFileInfo gestisca gli errori durante il recupero delle informazioni del file (GithubAPIRepository.spec.ts). (information vector db)	V
TU-155	Verifica che fetchRawFileContent recuperi il contenuto grezzo del file (GithubAPIRepository.spec.ts). (information vector db)	V
TU-156	Verifica che fetchRawFileContent gestisca gli errori durante il recupero del contenuto grezzo del file (GithubAPIRepository.spec.ts). (information vector db)	V
TU-157	Verifica che fetchRawFileContent lanci un errore quando la risposta non è una stringa (GithubAPIRepository.spec.ts). (information vector db)	V
TU-158	Verifica che fetchPullRequestsInfo recuperi le informazioni delle pull request (GithubAPIRepository.spec.ts). (information vector db)	V
TU-159	Verifica che fetchPullRequestsInfo gestisca gli errori durante il recupero delle pull request (GithubAPIRepository.spec.ts). (information vector db)	V
TU-160	Verifica che fetchPullRequestModifiedFiles recuperi i file modificati per una pull request (GithubAPIRepository.spec.ts). (information vector db)	V
TU-161	Verifica che fetchPullRequestModifiedFiles gestisca gli errori durante il recupero dei file modificati (GithubAPIRepository.spec.ts). (information vector db)	V
TU-162	Verifica che fetchPullRequestReviewComments recuperi i commenti di revisione per una pull request (GithubAPIRepository.spec.ts). (information vector db)	V
TU-163	Verifica che fetchPullRequestReviewComments gestisca gli errori durante il recupero dei commenti di revisione (GithubAPIRepository.spec.ts). (information vector db)	V
TU-164	Verifica che fetchRepositoryInfo recuperi le informazioni del repository (GithubAPIRepository.spec.ts). (information vector db)	V
TU-165	Verifica che fetchRepositoryInfo gestisca gli errori durante il recupero delle informazioni del repository (GithubAPIRepository.spec.ts). (information vector db)	V
TU-166	Verifica che fetchWorkflowsInfo recuperi le informazioni dei workflow (GithubAPIRepository.spec.ts). (information vector db)	V
TU-167	Verifica che fetchWorkflowsInfo gestisca gli errori durante il recupero delle informazioni dei workflow (GithubAPIRepository.spec.ts). (information vector db)	V
TU-168	Verifica che fetchWorkflowRuns recuperi le esecuzioni dei workflow senza il parametro since_created (GithubAPIRepository.spec.ts). (information vector db)	V
TU-169	Verifica che fetchWorkflowRuns recuperi le esecuzioni dei workflow con il parametro since_created (GithubAPIRepository.spec.ts). (information vector db)	V
TU-170	Verifica che fetchWorkflowRuns gestisca gli errori durante il recupero delle esecuzioni dei workflow (GithubAPIRepository.spec.ts). (information vector db)	V
TU-171	Verificare che tutte le informazioni di GitHub vengano memorizzate con successo (GithubStoreAdapter.spec.ts). (information vector db)	V

TU-172	Verificare che gli errori vengano gestiti durante la memorizzazione delle informazioni (GithubStoreAdapter.spec.ts). (information vector db)	V
TU-173	Verificare che gli array vuoti per qualsiasi tipo di informazione di GitHub vengano gestiti correttamente (GithubStoreAdapter.spec.ts). (information vector db)	V
TU-174	Verificare che i fallimenti parziali durante la memorizzazione delle informazioni vengano gestiti correttamente (GithubStoreAdapter.spec.ts). (information vector db)	V
TU-175	Verificare che le entità del repository vengano convertite in informazioni di dominio (Retrieval.adapter.spec.ts). (information vector db)	V
TU-176	Verificare che i risultati vuoti vengano gestiti correttamente (Retrieval.adapter.spec.ts). (information vector db)	V
TU-177	Verificare che gli errori lanciati dal repository vengano gestiti correttamente (Retrieval.adapter.spec.ts). (information vector db)	V
TU-178	Verifica che storeInformation memorizzi correttamente le informazioni (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-179	Verifica che storeInformation gestisca contenuti lunghi suddividendoli (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-180	Verifica che storeInformation gestisca gli errori durante la memorizzazione (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-181	Verifica che retrieveRelevantInfo recuperi correttamente le informazioni rilevanti (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-182	Verifica che retrieveRelevantInfo gestisca gli errori durante il recupero (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-183	Verifica che splitDocuments divida correttamente i documenti (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-184	Verifica che splitDocuments gestisca gli errori durante la divisione (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-185	Verifica che similaritySearch esegua correttamente la ricerca di similarità (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-186	Verifica che similaritySearch gestisca gli errori durante la ricerca di similarità (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-187	Verifica che deleteByMetadata elimini correttamente i documenti in base ai metadati (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-188	Verifica che deleteByMetadata gestisca gli errori durante l'eliminazione (Qdrant-information-repository.spec.ts). (information vector db)	V
TU-189	Verificare che `ApiController.getStorico` trasformi correttamente i parametri query in un `RequestChatCMD` e invochi `GetStoricoUseCase.execute()` restituendo un array di `ChatDTO`. (api gateway)	V
TU-190	Verificare che `GetStoricoService.execute()` invochi `StoricoPort.getStorico()` con i parametri ricevuti e restituisca i dati senza modifiche. (api gateway)	V

TU-191	Verificare che `StoricoMessageAdapter.getStorico()` trasformi i dati RabbitMQ in oggetti `Chat` del dominio, mappando correttamente `content` e `timestamp`. (api gateway)	V
TU-192	Verificare che `ApiController.getRisposta` trasformi il body della richiesta in un `ReqAnswerCmd`, invochi `GetChatUseCase.execute()`, e restituisca un `ChatDTO`. (api gateway)	V
TU-193	Verificare che `GetRispostaService.execute()` invochi `ChatBotPort.getRisposta()` e `StoricoPort.postStorico()`, e restituisca il risultato di quest'ultimo. (api gateway)	V
TU-194	Verificare che `MessageAdapter.getRisposta()` trasformi la risposta RabbitMQ in un `ProvChat`, preservando `question`, `answer`, e `timestamp`. (api gateway)	V
TU-195	Verificare che `StoricoMessageAdapter.postStorico()` costruisca un oggetto `Chat` dal `ProvChat` con i campi `Message` annidati (`content` + `timestamp`). (api gateway)	V
TU-196	Verificare che `StoricoMessageAdapter.postUpdate()` invochi `HistoryService.sendMessage()` con un `LastUpdateCMD` contenente la data corretta. (api gateway)	V
TU-197	Verificare che `StoricoMessageAdapter.getLastUpdate()` trasformi la risposta RabbitMQ in un `LastUpdateCMD`. (api gateway)	V
TU-198	Verificare che `StoricoMessageAdapter.getStorico()` lanci un errore se la risposta RabbitMQ non contiene `question.content` o `answer.content`. (api gateway)	V
TU-199	Verificare che `GetRispostaService.execute()` lanci un errore se `ChatBotPort.getRisposta()` restituisce un oggetto senza `timestamp`. (api gateway)	V
TU-200	Verificare che la classe ReqAnswerCmd inizializzi correttamente i suoi campi privati e li esponga attraverso i getter appropriati, preservando i riferimenti agli oggetti originali. (chatbot)	V
TU-201	Verificare che l'entità Chat memorizzi correttamente la domanda e la risposta e li restituisca attraverso i metodi getter, funzionando anche con stringhe vuote. (chatbot)	V
TU-202	Verificare che l'entità Metadata memorizzi correttamente i valori di origin, type e originID e li esponga attraverso getter e accesso diretto alle proprietà. (chatbot)	V
TU-203	Verificare che l'entità Information mantenga i riferimenti alla stringa di content e all'oggetto Metadata durante la sua creazione, esponendo i valori attraverso getter appropriati. (chatbot)	V

Tabella 12: Stato dei test di unità

3.3) Test di integrazione

I test di accettazione sono finalizzati a garantire che il prodotto soddisfi i ***requisiti***$_G$ utente come specificati nel ***capitolato***$_G$. Essi vengono eseguiti in presenza del committente e dimostrano la conformità del prodotto alle aspettative attraverso l'esecuzione dei casi di prova previsti nel capitolato. Il superamento positivo di tali test durante il collaudo finale generalmente conduce al rilascio definitivo del prodotto.

Codice	Descrizione	Stato
TI-01	Verificare che il flusso fetchChatHistory nel controller FetchHistoryController richiami correttamente il servizio, l'adapter e il repository mockato, restituendo una lista di ChatDTO correttamente mappata a partire da entità ChatEntity. (chat history)	V
TI-02	Verificare che il flusso di inserimento chat (in db) nel controller ChatConsumer richiami correttamente il servizio, l'adapter e il repository mockato, e restituisca un ChatDTO costruito a partire da una ChatEntity. (chat history)	V
TI-03	Verificare che il controller InsertLastUpdateController, attraverso il service e l'adapter, invochi correttamente il metodo insertLastRetrieval del repository con la data contenuta nel LastUpdateDTO e restituisca true. (chat history)	V
TI-04	Verificare che il controller FetchLastUpdateController, attraverso il service e l'adapter, invochi il metodo fetchLastUpdate del repository, e restituisca un LastUpdateDTO contenente il valore corretto. (chat history)	V
TI-05	Verificare l'interazione tra il componente Header e il hook useTheme per il cambio di tema. (frontend)	V
TI-06	Verificare l'interazione tra il Navbar e il componente per il cambio di tema. (frontend)	V
TI-07	Verificare che il componente ModeToggle interagisca correttamente con il hook useTheme. (frontend)	V
TI-08	Verificare l'integrazione del provider ChatProvider nel componente ChatWindow. (frontend)	V
TI-09	Verifica che sendMessage venga chiamato al submit del messaggio (incluso tasto Enter). (frontend)	V
TI-10	Verifica che il pulsante «Load more» e gli stati di errore e caricamento vengano correttamente gestiti. (frontend)	V
TI-11	Verifica che il ChatProvider integri correttamente l'adapter e gestisca la cronologia dei messaggi. (frontend)	V
TI-12	Verifica che i messaggi vengano caricati correttamente quando viene inviato un nuovo messaggio. (frontend)	V
TI-13	Verifica che fetchHistory invii la richiesta corretta all'API con i parametri giusti. (frontend)	V
TI-14	Verifica che fetchQuestion invii correttamente la domanda all'API. (frontend)	V
TI-15	Verifica che fetchHistory restituisca i dati corretti quando la risposta dell'API è positiva. (frontend)	V

TI-16	Verifica che fetchQuestion restituisca correttamente la risposta dall'API. (frontend)	V
TI-17	Verificare che il sistema possa recuperare e memorizzare i documenti di Confluence attraverso l'intero flusso (ConfluenceFetchAndStore.integration.spec.ts). (information vector db)	V
TI-18	Verificare che il sistema gestisca correttamente gli errori durante il recupero dei dati da Confluence (ConfluenceFetchAndStore.integration.spec.ts). (information vector db)	V
TI-19	Verificare che il sistema gestisca correttamente gli errori durante la memorizzazione dei dati di Confluence (ConfluenceFetchAndStore.integration.spec.ts). (information vector db)	V
TI-20	Verificare che il sistema possa recuperare e memorizzare le informazioni di GitHub con successo (GithubFetchAndStore.integration.spec.ts). (information vector db)	V
TI-21	Verificare che il sistema gestisca correttamente gli errori durante il recupero dei dati da GitHub (GithubFetchAndStore.integration.spec.ts). (information vector db)	V
TI-22	Verificare che il sistema gestisca correttamente gli errori durante il flusso di lavoro di GitHub (GithubFetchAndStore.integration.spec.ts). (information vector db)	V
TI-23	Verificare che il sistema possa recuperare e memorizzare i ticket di Jira attraverso l'intero flusso (JiraFetchAndStore.integration.spec.ts). (information vector db)	V
TI-24	Verificare che il sistema gestisca correttamente gli errori durante il recupero dei dati da Jira (JiraFetchAndStore.integration.spec.ts). (information vector db)	V
TI-25	Verificare che il sistema gestisca correttamente gli errori durante la memorizzazione dei dati di Jira (JiraFetchAndStore.integration.spec.ts). (information vector db)	V
TI-26	Verificare che il sistema possa recuperare le informazioni quando viene chiamato direttamente (RetrievalService.integration.spec.ts). (information vector db)	V
TI-27	Verificare che il sistema gestisca correttamente gli errori durante il recupero delle informazioni (RetrievalService.integration.spec.ts). (information vector db)	V
TI-28	Verificare che `GET /get-storico` restituisca 500 se `HistoryService` risponde con dati malformati. (api gateway)	V
TI-29	Verificare che `POST /get-risposta` restituisca 500 se il servizio RabbitMQ del chatbot non è raggiungibile. (api gateway)	V
TI-30	Verificare che una richiesta `GET /get-storico` invochi correttamente `GetStoricoService` → `StoricoMessageAdapter` → `HistoryService`, restituendo `ChatDTO[]` con i campi `question.content` e `answer.content`. (api gateway)	V
TI-31	Verificare che una richiesta `POST /get-risposta` invochi `GetRispostaService` → `MessageAdapter` (per il chatbot) →	V

	`StoricoMessageAdapter` (per il salvataggio), e restituisca un `ChatDTO` con la struttura annidata `{ question: { content, timestamp }, ... }`. (api gateway)	
TI-32	Verificare che il flusso di salvataggio data ultimo fetch (`postUpdate`) invochi `HistoryService` con la data corretta e restituisca `true` in caso di successo. (api gateway)	V
TI-33	Verificare che il flusso di recupero data ultimo fetch (`getLastUpdate`) restituisca un `LastUpdateCMD` con il campo `LastFetch` valorizzato. (api gateway)	V
TI-34	Verificare che il flusso di salvataggio storico (`postStorico`) invochi `HistoryService` con un `Chat` correttamente costruito e restituisca `true` in caso di successo. (api gateway)	V
TI-35	Verificare che ElaborazioneService richiami correttamente i metodi searchVectorDb del VectorDbPort e generateAnswer del LLMPort, passando i parametri appropriati e restituendo la risposta generata. (chatbot)	V
TI-36	Verificare che ChatController, quando riceve una chiamata getAnswer con un ReqAnswerDTO, crei correttamente un oggetto ReqAnswerCmd, invochi il metodo getAnswer del caso d'uso ElaborazioneUseCase e restituisca la risposta ottenuta. (chatbot)	V
TI-37	Verificare che VectorDbAdapter, quando invocato con una query, chiami correttamente il metodo sendMessage del VectorDbClient con i parametri corretti e trasformi la risposta in un array di oggetti Information. (chatbot)	V
TI-38	Verificare che GroqAdapter (tramite implementazione mock) filtri correttamente i documenti che supererebbero il limite di token, producendo log appropriati e restituendo una risposta basata sulle informazioni filtrate. (chatbot)	V
TI-39	Verificare che GroqAdapter gestisca correttamente il caso di array di informazioni vuoto, restituendo una risposta predefinita che indica l'assenza di informazioni rilevanti. (chatbot)	V
TI-40	Verificare che VectorDbAdapter propaghi correttamente gli errori provenienti dal client e gestisca i casi di risposta vuota, restituendo un array vuoto. (chatbot)	V
TI-41	Verificare che VectorDbAdapter possa elaborare le risposte JSON correttamente quando queste sono già state deserializzate. (chatbot)	V

Tabella 13: Stato dei test di integrazione

3.4) Test di sistema

I test di sistema sono una fase del processo di testing software che mira a verificare che il sistema soddisfi i *requisiti*$_G$ specificati nella sezione Requisiti di funzionalità del documento `Analisi dei Requisiti`. Questa fase di testing è condotta sul sistema nel suo complesso, dopo che i test di unità e di integrazione sono stati completati con successo. L'obiettivo principale dei test di sistema è assicurare che l'applicazione sia in grado di svolgere le sue funzioni nel contesto del suo ambiente operativo.

Codice	Descrizione	Stato	Requisito
TS-01	Verificare che l'utenta riesca ad accedere all'applicazione senza autenticazione	V	RF-001
TS-02	Verificare che l'utente possa visualizzare correttamente lo storico della chat	V	RF-002
TS-03	Verificare che l'utente visualizzi un messaggio nel caso in cui non ci siano messaggi nello storico	V	RF-003
TS-04	Verificare che l'utente visualizzi un messaggio di errore nel caso in cui il sistema non sia riuscito a recuperare correttamente lo storico	V	RF-004
TS-05	Verificare che l'utente visualizzi un messaggio di errore nel caso in cui il sistema non riesca a connettersi	V	RF-005
TS-06	Verificare che l'utente visualizzi un messaggio di errore nel caso in cui il backend non risulti disponibile	V	RF-006
TS-07	Verificare che l'utente possa visualizzare per ogni messaggio il suo contenuto, data, orario di invio e mittente	V	RF-007
TS-08	Verificare che, attraverso l'interfaccia utente, l'utente sia in grado di porre una domanda in linguaggio naturale	V	RF-008
TS-09	Verificare che l'utente riesca ad inviare la domanda scritta attraverso la *User Interface*$_G$ al sistema	V	RF-009
TS-10	Verificare che, nel caso in cui ci sia stato un errore durante la generazione della risposta, l'utente visualizzi un messaggio di errore	V	RF-010
TS-11	Verificare che il sistema notifichi all'utente un messaggio di errore nel caso in cui la risposta non venga generata perché supera la lunghezza massima consentita	V	RF-011
TS-12	Verificare che il sistema notifichi all'utente un messaggio di errore nel caso in cui la domanda superi la lunghezza massima consentita	V	RF-012
TS-13	Verificare che il sistema elabori correttamente la domanda dell'utente, generando una risposta attinente e appropriata	V	RF-013
TS-14	Verificare che il sistema riesca a reperire tutte le informazioni necessarie da *GitHub*$_G$ (nome della repository, la sua descrizione, informazioni sui ticket, commit...)	V	RF-014
TS-15	Verificare che il sistema riesca a reperire tutte le informazioni necessarie da *Confluence*$_G$ (id della pagina, il titolo, lo status...)	V	RF-015

TS-16	Verificare che il sistema riesca a reperire tutte le informazioni necessarie da *Jira*$_G$ (nome di un *ticket*$_G$, il suo assegnatario, stato, ticket collegati…)	V	RF-016
TS-17	Verificare che il sistema riesca gestire correttamente domande fuori contesto, generando una risposta attinente e appropriata	V	RF-017
TS-18	Verificare che, nel caso in cui ci sia stato un errore durante la generazione della risposta, l'utente visualizzi un messaggio di errore	V	RF-018
TS-19	Verificare che il sistema informi l'utente se la risposta supera la lunghezza massima consentita	V	RF-019
TS-20	Verificare che il sistema fornisca correttamente la data e l'orario dell'ultimo aggiornamento dei dati utilizzati	V	RF-020
TS-21	Verificare che ogni 24 ore il sistema aggiorni i dati contenuti nei documenti provenienti da GitHub, Confluence e Jira	V	RF-021

Tabella 14: Stato dei test di sistema

3.5) Test di accettazione

I test di accettazione sono finalizzati a garantire che il prodotto soddisfi i ***requisiti***$_G$ utente come specificati nel capitolato. Essi vengono eseguiti in presenza del committente e dimostrano la conformità del prodotto alle aspettative attraverso l'esecuzione dei casi di prova previsti nel capitolato. Il superamento positivo di tali test durante il collaudo finale generalmente conduce al rilascio definitivo del prodotto.

Codice	Descrizione	Stato	Fonte
TA-01	Verificare che l'utente possa visualizzare lo storico della chat	V	UC1
TA-02	Verificare che per ogni messaggio l'utente possa visualizzare: contenuto, data, orario, mittente	V	UC1.4, UC1.4.1, UC1.4.2, UC1.4.3
TA-03	Verificare che l'utente possa inserire ed inviare attraverso l'interfaccia utente una nuova domanda	V	UC2
TA-04	Verificare che la domanda scritta da un utente venga inviata correttamente al sistema attraverso la ***User Interface***$_G$	V	UC3
TA-05	Verificare che il sistema generi una risposta appropriata dopo aver elaborato correttamente la domanda dell'utente	V	UC4
TA-06	Verificare che il sistema recuperi tutte le informazioni necessarie da ***GitHub***$_G$	V	UC4
TA-07	Verificare che il sistema recuperi tutte le informazioni necessarie da ***Confluence***$_G$	V	UC4
TA-08	Verificare che il sistema recuperi tutte le informazioni necessarie da ***Jira***$_G$	V	UC4
TA-09	Verificare che il sistema fornisca data e orario dell'ultimo aggiornamento dei dati utilizzati	V	UC4.4

Tabella 15: Stato dei test di accettazione

4) Cruscotto di valutazione della qualità

4.1) `M-PRC-EAC` - Estimated at Completion

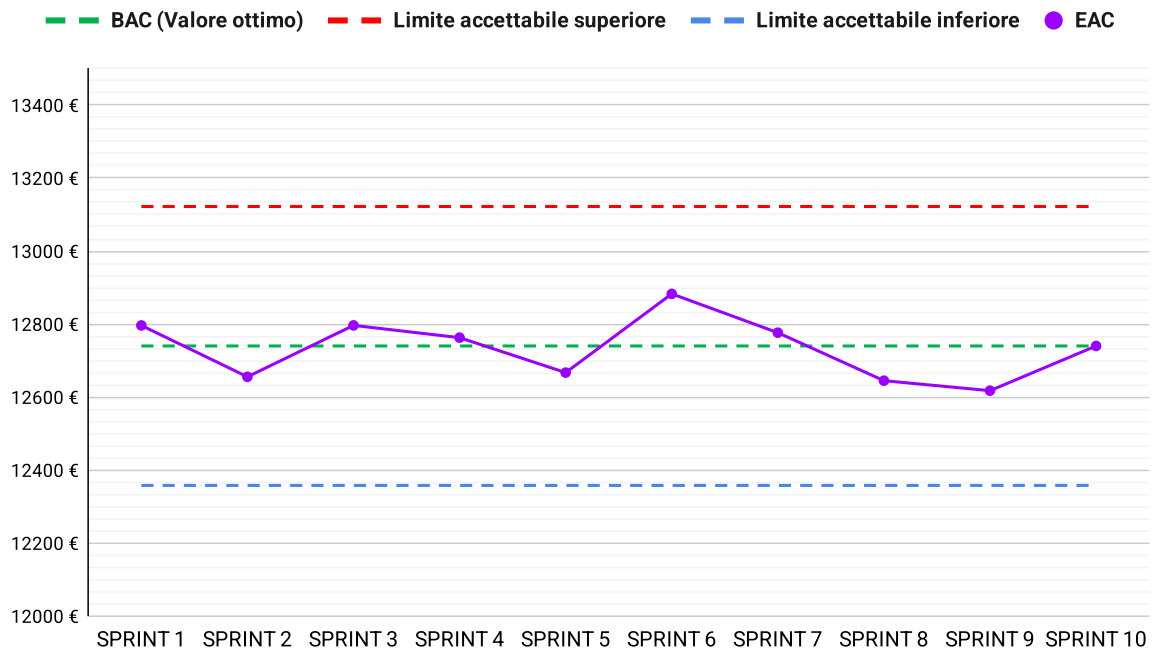


Figura 1: Proiezione della stima del costo totale nei vari periodi di progetto.

RTB: Il grafico illustra come varia l'EAC (**Estimate at Completion**) in relazione ai limiti accettabili (superiore e inferiore) e al valore ottimale, rappresentato dal BAC (**Budget at Completion**), durante i cinque sprint del progetto. L'EAC rappresenta una previsione aggiornata dei costi complessivi per portare a termine il progetto, stimata sulla base dei dati raccolti durante l'esecuzione. Viene utilizzato per identificare eventuali scostamenti rispetto al BAC, che corrisponde al budget inizialmente pianificato per il progetto. Osservando il grafico ne emerge che in seguito al secondo periodo del progetto l'EAC è sceso sotto il valore ottimale BAC, questo indica un'efficienza migliorata nell'uso delle risorse grazie a una riduzione dei costi ed a un'accelerazione nei tempi di completamento. Durante il terzo e quarto sprint invece si può notare un aumento del EAC a causa di difficoltà operative e tecniche che hanno richiesto al team di concentrarsi più attivamente ad attività di formazione e supporto piuttosto che all'avanzamento vero e proprio del progetto. Infine nel quinto ed ultimo periodo la stima è rimasta pressochè costante avvicinandosi al limite accettabile inferiore denotando una risoluzione efficace alle lacune riscontrate durante i periodi precedenti portando ad una gestione più rigorosa delle risorse.

PB: A partire dallo *Sprint*$_G$ 6, l'EAC mostra un leggero aumento rispetto al periodo precedente, avvicinandosi al BAC, a indicare una previsione di spesa leggermente superiore, ma comunque entro i limiti accettabili. Tuttavia, già nello *Sprint*$_G$ 7 si osserva una nuova flessione, seguita da un calo più marcato nello *Sprint*$_G$ 8, in cui si registra il valore minimo dell'intero intervallo considerato. Questa diminuzione potrebbe essere legata a un'ottimizzazione dei processi o a una riduzione delle ore previste in alcuni ruoli. Lo *Sprint*$_G$ 9 segna un ulteriore lieve ribasso, per poi chiudere con un incremento moderato nello *Sprint*$_G$ 10. Nonostante queste variazioni, l'EAC rimane sempre contenuto entro i limiti stabiliti, segno di una buona gestione economica nella fase finale del progetto.

4.2) `M-PRC-EV` - Earned Value & `M-PRC-PV` - Planned Value

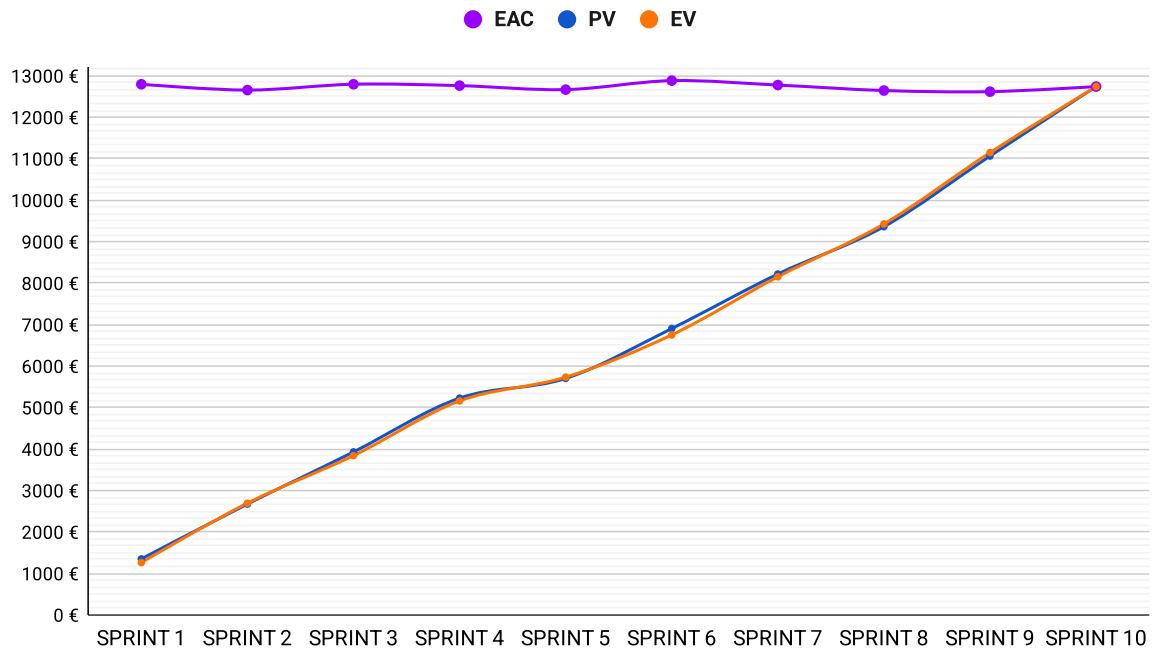


Figura 2: Proiezione dell'EV e del PV nei vari periodi di progetto.

RTB: Il grafico evidenzia la relazione tra il piano originale (PV), i progressi effettivi (EV) e le previsioni di costo complessive aggiornate (EAC), permettendo di valutare l'efficacia e l'efficienza della gestione del progetto durante i vari sprint. Osservando il grafico ne emerge una chiara corrispondenza tra la curva del Valore Guadagnato (**Earned Value**) e quella del Valore Pianificato (**Planned Value**). Questo allineamento indica che il lavoro completato è conforme alla pianificazione, suggerendo un progresso coerente e in linea con gli obiettivi del progetto.

PB: A partire dallo *Sprint*$_G$ 6, si osserva una prosecuzione dell'andamento parallelo tra il Valore Pianificato (**PV**) e il Valore Guadagnato (**EV**), segno che le attività procedono in linea con quanto previsto. Questo trend positivo si mantiene costante anche negli *sprint*$_G$ successivi, con una perfetta sovrapposizione tra le due curve fino allo *Sprint*$_G$ 10, dove il progetto si conclude con il raggiungimento del valore complessivo previsto. Per quanto riguarda l'**EAC**, la stima dei costi aggiornati resta sostanzialmente stabile, con leggere variazioni che non compromettono la coerenza generale. La convergenza di **PV**, **EV** ed **EAC** verso il medesimo valore finale testimonia un'elevata efficacia nella pianificazione e nel controllo, confermando che il progetto è stato completato rispettando sia i tempi che i costi prefissati.

4.3) `M-PRC-AC` - Actual Cost & `M-PRC-ETC` - Estimate to Complete

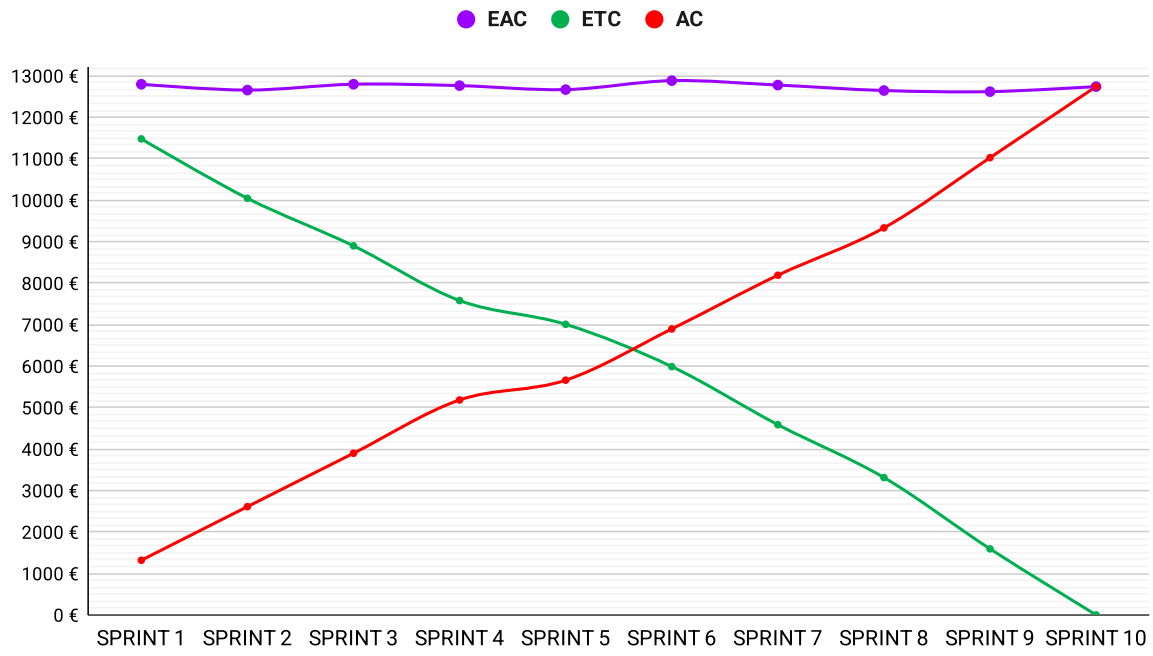


Figura 3: Proiezione dell'AC e dell'ETC nei vari periodi di progetto.

RTB: Il grafico rappresenta l'**Estimate to Complete** (ETC), ovvero la stima dei costi rimanenti necessari per completare il progetto, e l'**Actual Cost** (AC), che indica i costi reali sostenuti per il lavoro svolto fino a quel momento. Si nota che l'ETC diminuisce progressivamente nel tempo, mentre l'AC cresce in modo proporzionale al ritmo con cui si riduce l'ETC.

PB: Osservando il grafico, si nota un andamento complessivamente lineare e regolare dell'**AC**, che cresce in modo costante durante tutti gli *sprint*$_G$, senza evidenti fluttuazioni o discontinuità. Questo suggerisce una distribuzione omogenea del carico di lavoro e un'efficace gestione delle risorse. Parallelamente, l'**ETC** decresce gradualmente in modo proporzionale, riflettendo l'avanzamento costante del progetto verso il completamento. Al termine dell'ultimo sprint, corrispondente alla candidatura per la revisione PB, l'**AC** raggiunge l'**EAC**, mentre l'**ETC** si assesta su un valore prossimo allo zero. Questo risultato conferma che le attività sono state eseguite secondo quanto pianificato, con una previsione di spesa pressoché coincidente con quella effettiva e senza la necessità di revisioni straordinarie o interventi correttivi.

4.4) `M-PRC-CV` - Cost Variance & `M-PRC-SV` - Schedule Variance

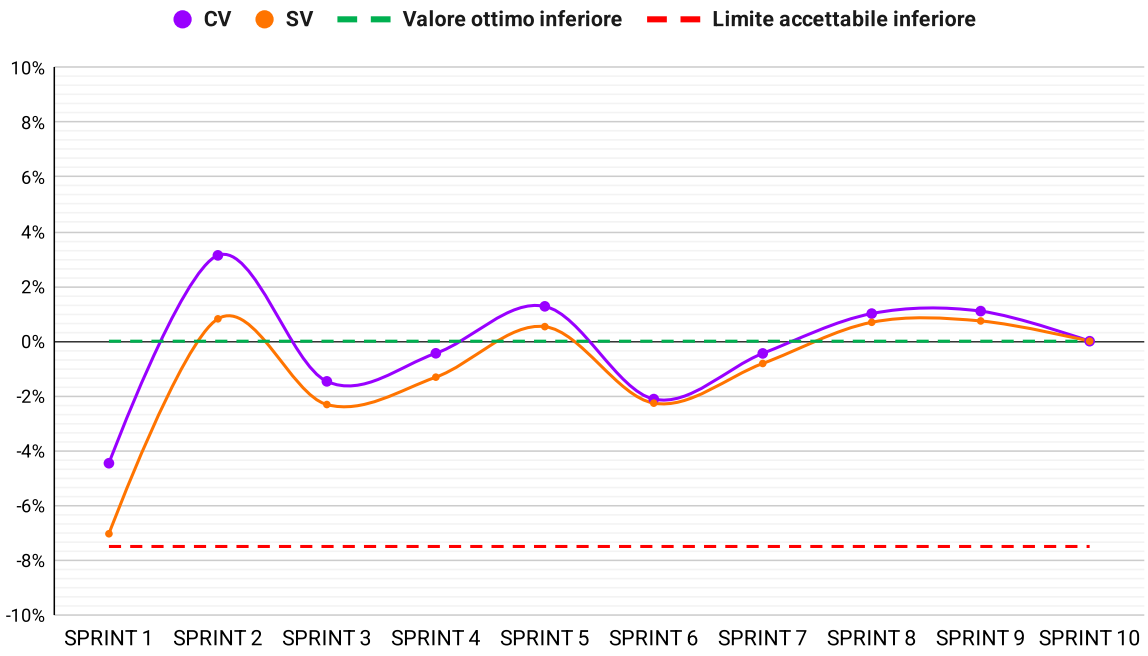


Figura 4: Proiezione della CV e della SV nei vari periodi di progetto.

RTB: Il grafico evidenzia la **Cost Variance** (CV), rappresentante la differenza tra il valore guadagnato (EV) e i costi sostenuti (AC) in percentuale, e la **Schedule Variance** (SV), indicando la differenza tra il valore guadagnato (EV) e il valore pianificato (PV) in percentuale.
Durante il primo sprint entrambi gli indicatori risultano negativi: i costi superano il valore realizzato e il progetto accumula ritardi rispetto alla pianificazione. Nel secondo sprint il Controllo dei Costi (CV) si attesta in positivo, segnalando una gestione più efficiente, mentre lo Scarto dei Tempi (SV) recupera leggermente pur rimanendo in negativo. Durante il terzo sprint lo SV peggiora ulteriormente, evidenziando un ritardo crescente, e sebbene il CV rimanga positivo, si registra una sua flessione, a indicare un incremento delle spese rispetto ai precedenti sprint. La svolta avviene nel quarto sprint, in cui entrambe le misure si avvicinano ai parametri ideali, a testimonianza di un miglioramento nella gestione sia del budget che delle tempistiche. Infine, nel quinto sprint il trend positivo si consolida: il controllo dei costi continua a rafforzarsi e, per la prima volta, il ritardo accumulato si trasforma in un avanzo temporale, indicando non solo il recupero delle criticità iniziali, ma anche un progresso oltre il piano stabilito.

PB: Guardando il grafico a partire dallo *Sprint*$_G$ 6, si osserva come la **CV** (**Cost Variance**) si mantenga sempre vicina allo 0%, indicando un'elevata coerenza tra i costi sostenuti e il valore effettivamente guadagnato. Questo andamento regolare suggerisce una gestione economica estremamente accurata nella fase finale del progetto. Per quanto riguarda la **SV** (**Schedule Variance**), anche in questo caso le variazioni risultano contenute: dopo un lieve calo iniziale allo *Sprint*$_G$ 6, la curva risale progressivamente, stabilizzandosi su valori prossimi allo zero fino alla conclusione. Questi dati confermano un controllo efficace della pianificazione e dell'avanzamento: il gruppo ha infatti rispettato tempi e attività con grande precisione nella fase che va dalla RTB fino alla **PB**, mantenendo entrambe le variabili entro i limiti ottimali e senza scostamenti significativi.

4.5) `M-PRC-RSI` - Requirements stability index

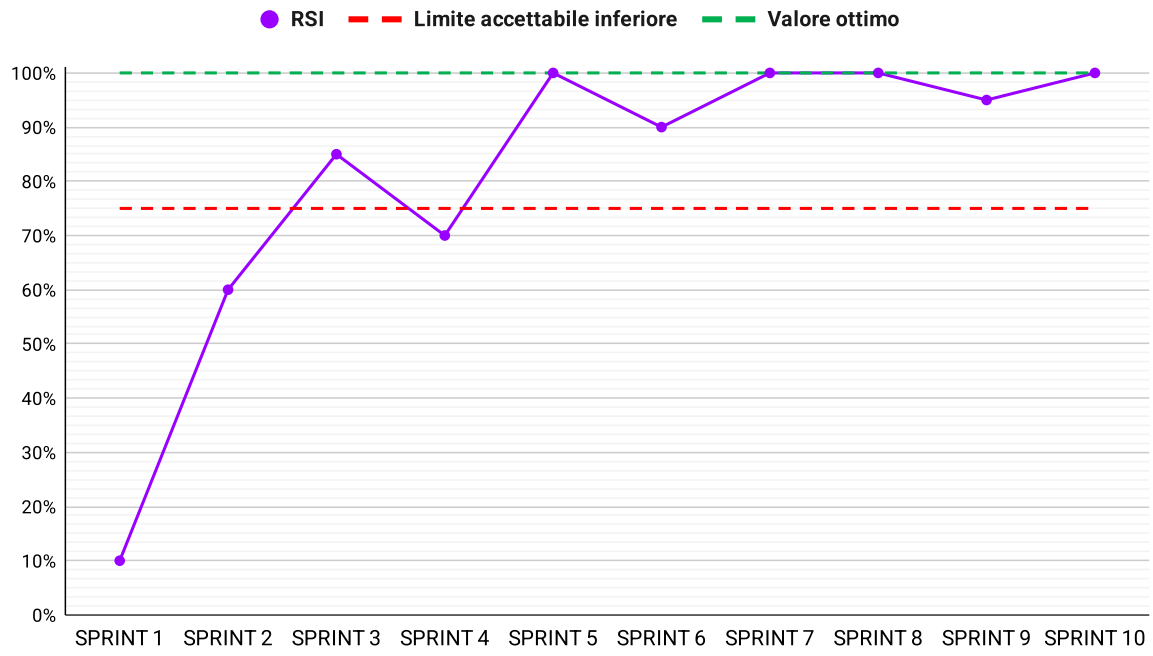


Figura 5: Proiezione del RSI nei vari periodi di progetto.

`RTB`: Il grafico illustra la dinamica della metrica **Requirements stability index** (`RSI`), volta a valutare la stabilità dei requisiti del progetto nel corso del tempo. Emerge chiaramente una rapida crescita alla fine del secondo periodo, coincidente con l'avvio dell'analisi dei requisiti da parte del gruppo. Inoltre, si nota un ulteriore aumento alla fine del terzo periodo, indicativo di modifiche e/o aggiornamenti nell'analisi dei requisiti che sono andati a diminuire. Il parametro poi è diminuito alla fine del quarto periodo per via di modifiche importanti, necessarie a raggiungere un livello di dettaglio dei requisiti ancora maggiore. Infine si nota che nel quinto ed ultimo periodo i requisiti sono migliorati fino ad arrivare ad un valore `RSI` pari al `100%`.

PB: Dal grafico si può osservare come, in seguito a un piccolo aggiustamento dei requisiti avvenuto durante la revisione **`RTB`** nell *Sprint*$_G$ 6, i requisiti siano rimasti sostanzialmente stabili a partire dallo *Sprint*$_G$ 6, mantenendo un **`RSI`** prossimo al `100%`. Questo dato riflette una fase di progetto ben consolidata, caratterizzata da una forte coerenza nei requisiti, che ha comportato vantaggi significativi quali maggiore stabilità del `Piano di Progetto`, risparmio di tempo e risorse, chiarezza negli obiettivi e riduzione del rischio di errori. Va tuttavia segnalato che, in chiusura dello *Sprint*$_G$ 9, è stato necessario intervenire per correggere alcuni requisiti risultati poco precisi. Tale intervento, sebbene limitato, non ha compromesso la stabilità generale, ma dimostra un'attenta attività di controllo e validazione che ha ulteriormente rafforzato la qualità del lavoro svolto. Una solida `Analisi dei Requisiti` iniziale si è dunque rivelata determinante per il buon esito del progetto.

4.6) `M-PRC-GLP` - Indice Gulpease

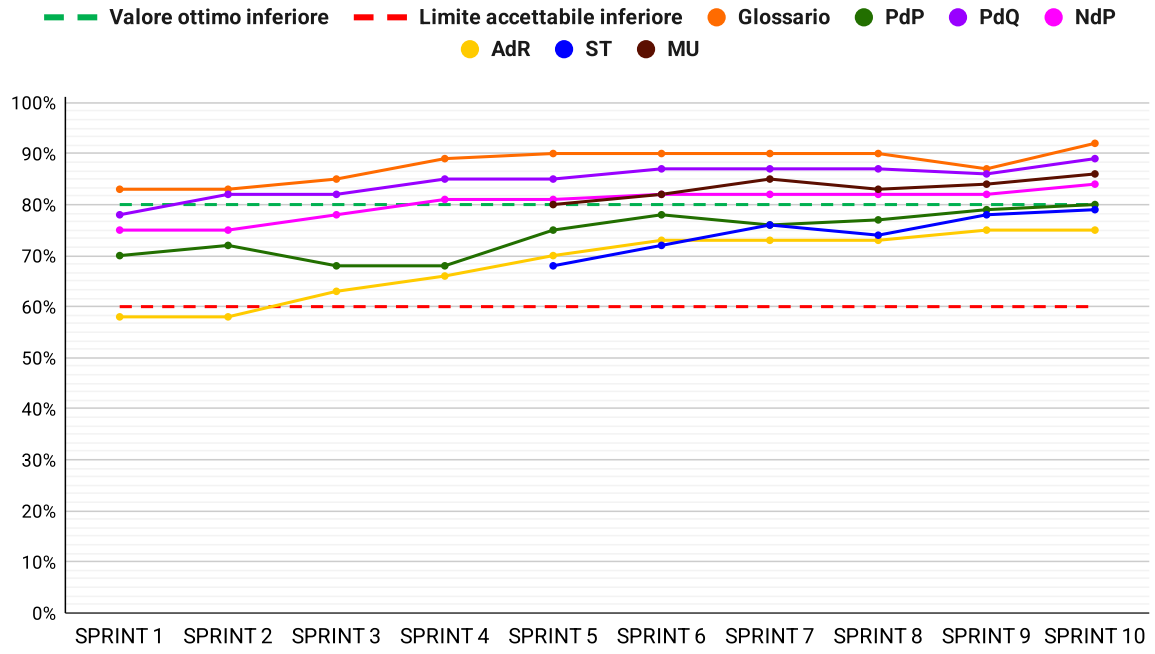


Figura 6: Proiezione dell'indice Gulpease per ogni documento (RTB) nei vari periodi di progetto.

RTB: Dall'osservazione del grafico si evidenzia come, per la maggior parte dei documenti, l'indice tenda ad aumentare o a rimanere stabile nel tempo. L'unica eccezione è il `Piano di Progetto`, che si discosta da questa tendenza a causa delle significative modifiche apportate al suo contenuto nel corso dei periodi considerati. Va inoltre notato che l'`Analisi dei Requisiti` è l'unico documento che inizia al di sotto del limite inferiore accettabile, un fenomeno riconducibile alla natura specifica degli argomenti trattati e al linguaggio utilizzato. Nonostante ciò, tutti i documenti risultano comprensibili anche per chi possiede un livello di istruzione pari alla licenza media.

PB: Osservando il grafico si può notare come, nei periodi successivi alla revisione **RTB**, i documenti già esistenti abbiano mantenuto un indice di Gulpease pressoché stabile, in linea con i valori precedenti. Questo evidenzia una costanza nella qualità redazionale e nella comprensibilità dei testi, senza variazioni rilevanti nella complessità sintattica o nella struttura linguistica. A questi si sono aggiunti il `Manuale Utente` e la `Specifica Tecnica`. Il primo presenta un indice più elevato, risultando quindi più accessibile, mentre il secondo ha un valore inferiore, coerente con la natura tecnica e dettagliata del suo contenuto. Tale differenza è attesa e perfettamente giustificata. In conclusione, possiamo affermare che tutti i documenti prodotti risultano leggibili e comprensibili da parte di utenti con almeno il titolo di licenza media, garantendo così l'accessibilità della documentazione anche a persone non esperte.

4.7) `M-PRC-CO` - Correttezza Ortografica

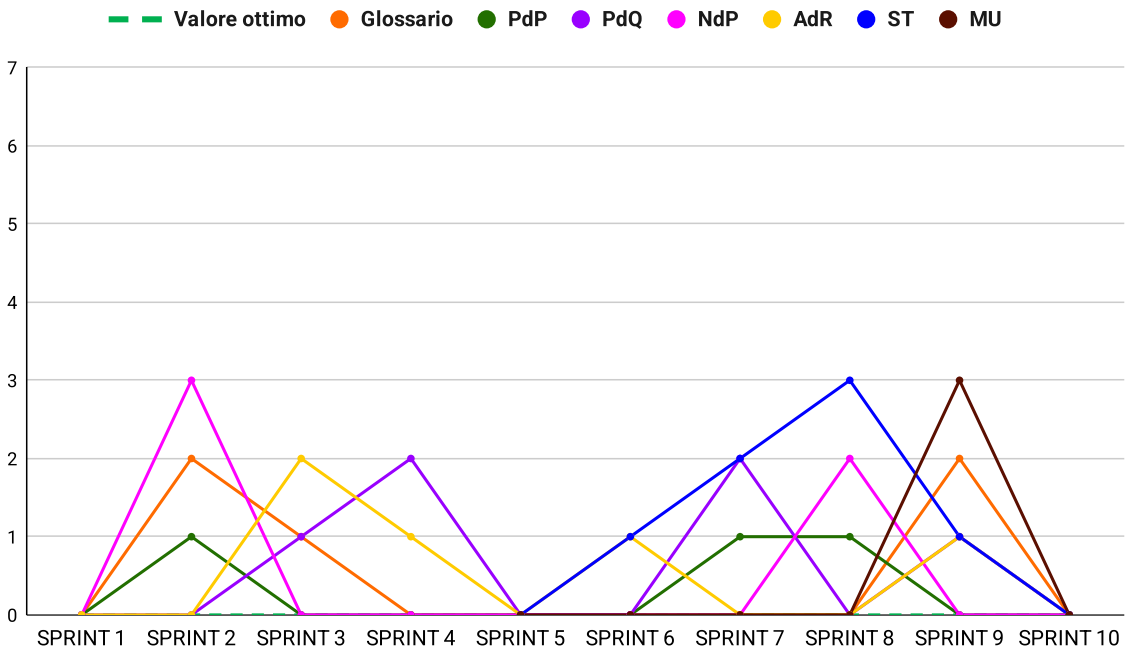


Figura 7: Proiezione della correttezza ortografica nei vari periodi di progetto.

RTB: Dal grafico si nota un impegno costante da parte del gruppo nel mantenere i documenti privi di errori ortografici. Sebbene alcuni errori siano inevitabilmente sfuggiti, la maggior parte dei documenti è rimasta senza errori per la maggior parte del tempo, raggiungendo un livello ottimale di accuratezza nell'ultimo periodo.

PB: Il grafico mostra come il numero di errori ortografici sia rimasto contenuto anche nella seconda parte del progetto, dimostrando una cura costante nella revisione dei testi. Sebbene lo zero assoluto di errori sia stato raggiunto su tutti i documenti solo nell'ultimo periodo, questo risultato evidenzia un progressivo miglioramento e una crescente attenzione alla qualità formale della documentazione. Il traguardo finale riflette un impegno continuo nel rilevare e correggere anche i più piccoli refusi, pur riconoscendo che, nel corso del lavoro, qualche imperfezione può essere sfuggita.

4.8) `M-PRC-QMS` - Quality Metrics Satisfied

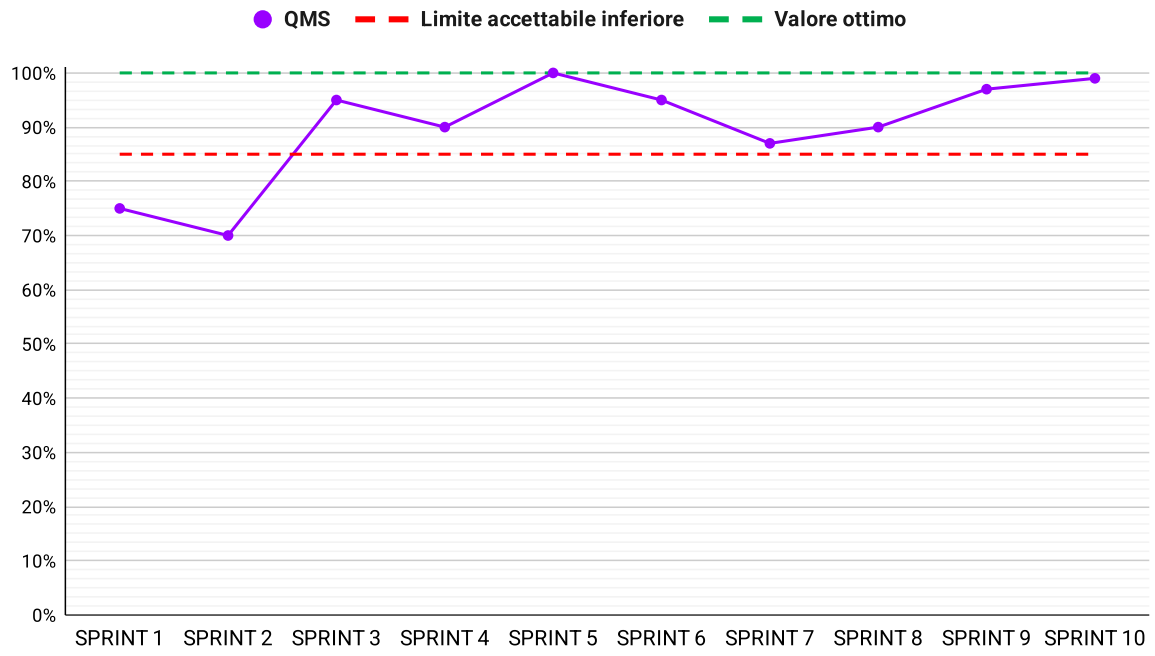


Figura 8: Proiezione della percentuale di metriche di qualità soddisfatte nei vari periodi di progetto.

RTB: Il grafico evidenzia che, nei primi periodi, alcune metriche di qualità definite dal gruppo non hanno raggiunto i valori desiderati, principalmente a causa dell'inesperienza iniziale dei membri. Tuttavia, grazie all'apprendimento dagli errori commessi, si è osservato un costante miglioramento, che ha portato a raggiungere valori accettabili e infine il livello ottimale (`100%`) al termine del quinto periodo. Questo risultato riflette un progresso significativo nel nostro approccio lavorativo e nei risultati qualitativi ottenuti.

PB: Il grafico evidenzia come, in seguito al superamento della revisione RTB, il valore della metrica si sia mantenuto costantemente al di sopra della soglia accettabile, a testimonianza di un livello qualitativo complessivamente positivo. Nelle fasi iniziali di questa seconda parte del progetto si osserva un leggero calo, coerente con la consapevolezza che il pieno soddisfacimento delle metriche di qualità sarebbe stato raggiunto solo in prossimità della chiusura del lavoro. Nell'ultimo periodo, si registra un chiaro miglioramento: quasi tutte le metriche hanno raggiunto valori pienamente accettabili, e in alcuni casi anche ottimali. Questo risultato conferma il costante impegno del gruppo nel perfezionare la qualità del prodotto, e rappresenta un segnale tangibile dell'efficacia delle strategie correttive adottate nel tempo.

4.9) `M-PRC-NCR` - Non-Calculated Risk

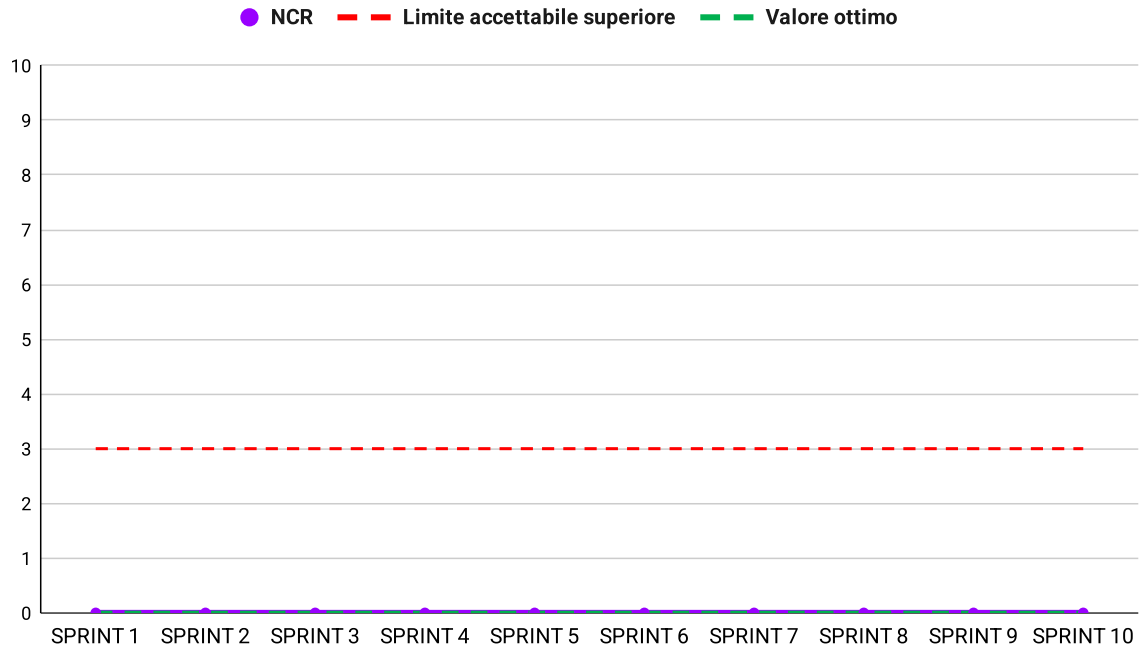


Figura 9: Proiezione rischi non identificati nei vari periodi di progetto.

RTB: Il grafico evidenzia come il gruppo abbia mantenuto un'ottima capacità di previsione dei rischi per l'intera durata del periodo considerato, senza che emergessero situazioni inattese. Sebbene la metrica non garantisca una futura assenza assoluta di rischi non previsti, indica che nessun rischio non anticipato si è verificato, sottolineando la precisione e l'efficacia nella gestione dei rischi da parte del gruppo.

PB: Anche negli ultimi cinque *sprint*$_G$, il grafico conferma l'elevata capacità del gruppo nel prevedere e gestire i rischi. Non si sono verificate situazioni impreviste, e la metrica ha mantenuto valori stabili, a riprova della solidità del processo di analisi e monitoraggio dei rischi adottato. Questo andamento costante riflette una pianificazione accurata e un controllo efficace, che hanno contribuito a mantenere il progetto su binari sicuri fino alla sua conclusione.

4.10) `M-PRC-TE` - Temporal Efficiency

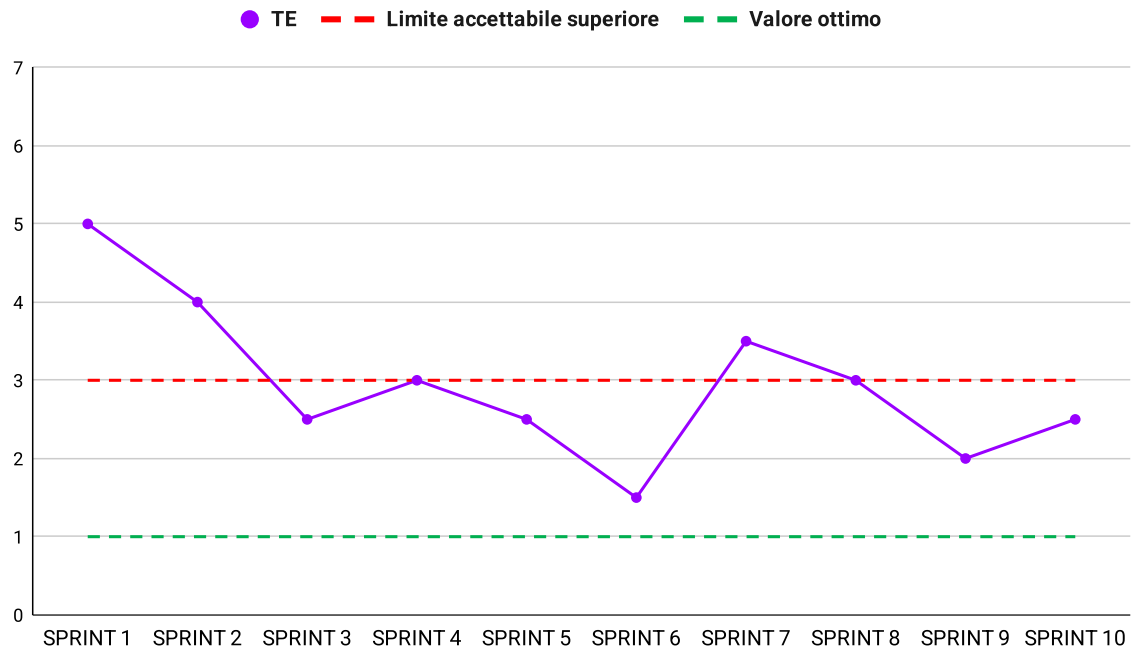


Figura 10: Proiezione dell'efficienza temporale nei vari periodi di progetto.

RTB: Il grafico mostra l'andamento della metrica sull'efficienza temporale nel corso dei vari periodi. Nei primi due periodi, la metrica si posiziona oltre il limite superiore accettabile, per poi stabilizzarsi entro valori adeguati a partire dal terzo periodo. Questo comportamento riflette il processo di adattamento del gruppo alle nuove tecnologie, ambienti e linguaggi richiesti, oltre che alle pratiche di gestione del progetto. Nel tempo, si osserva un netto miglioramento, con una riduzione del tempo necessario per raggiungere i risultati desiderati, segno di una maggiore esperienza acquisita dai membri del gruppo.

PB: Nella seconda parte del progetto, durante lo $Sprint_G$ 6, abbiamo ottenuto un buon miglioramento della metrica, avvicinandoci a una corrispondenza tra le ore produttive e quelle di orologio, anche se non perfetta. Questo risultato è stato possibile grazie al fatto che ci siamo concentrati su compiti che già conoscevamo bene, come il perfezionamento dei requisiti del progetto e dei test di sistema, attività che ci permettevano di lavorare in modo più efficiente. Nello $Sprint_G$ 7, la metrica è aumentata, superando il limite accettabile superiore, a causa dell'introduzione di nuove attività, come la progettazione del prodotto, con cui non avevamo esperienza. Questo ha portato ad un aumento delle ore di orologio rispetto alle ore produttive. Nel periodo successivo, durante lo $Sprint_G$ 8, abbiamo registrato un miglioramento, scendendo a un valore di 3. Da lì in poi, la metrica è continuata a migliorare, fino a stabilizzarsi intorno a `2:30`, rientrando così nei limiti accettabili.

4.11) `M-PRC-PTCP` - Passed Test Cases Percentage

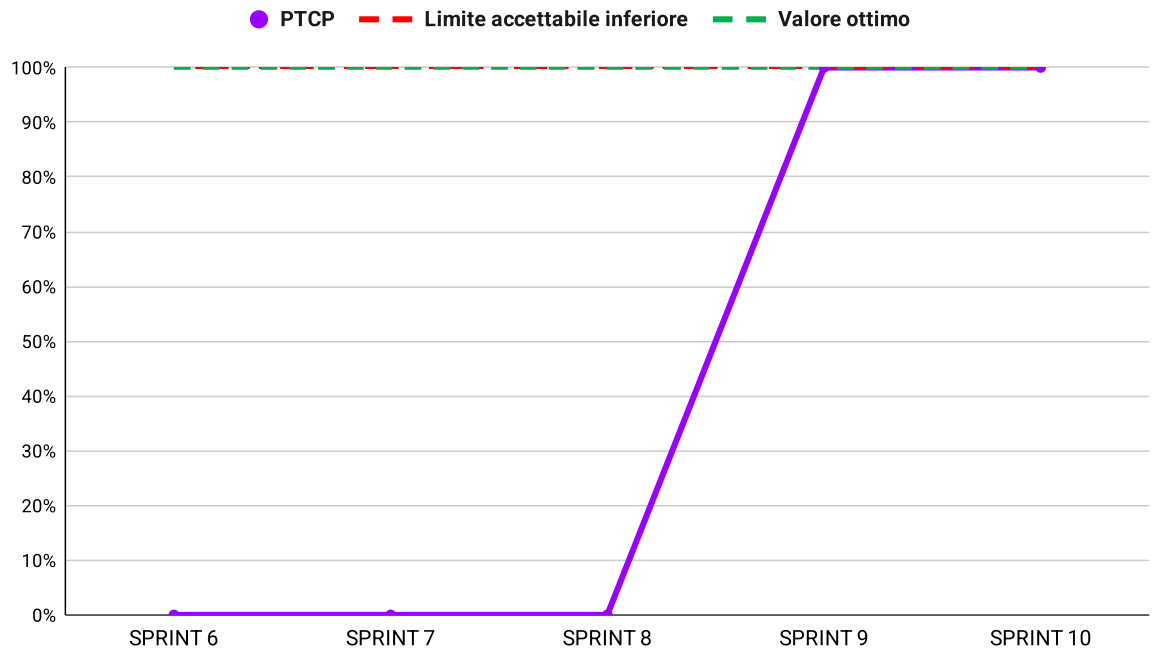


Figura 11: Proiezione della percentuale di test terminati con successo nei vari periodi di progetto.

PB: L'analisi del grafico mostra che, fin dall'introduzione dei primi test, avvenuta all'inizio dello *Sprint*$_G$ 8, questi sono stati sempre completamente superati. È importante sottolineare che non esiste alcuna differenza tra il limite minimo accettabile e il valore ottimale per questa metrica, poiché è fondamentale che il prodotto superi con successo tutti i test a cui viene sottoposto. Questo evidenzia l'importanza di mantenere costantemente alti standard di performance e qualità durante tutto il processo di sviluppo.

4.12) `M-PRD-CRV` - Copertura dei requisiti obbligatori (vincolanti)

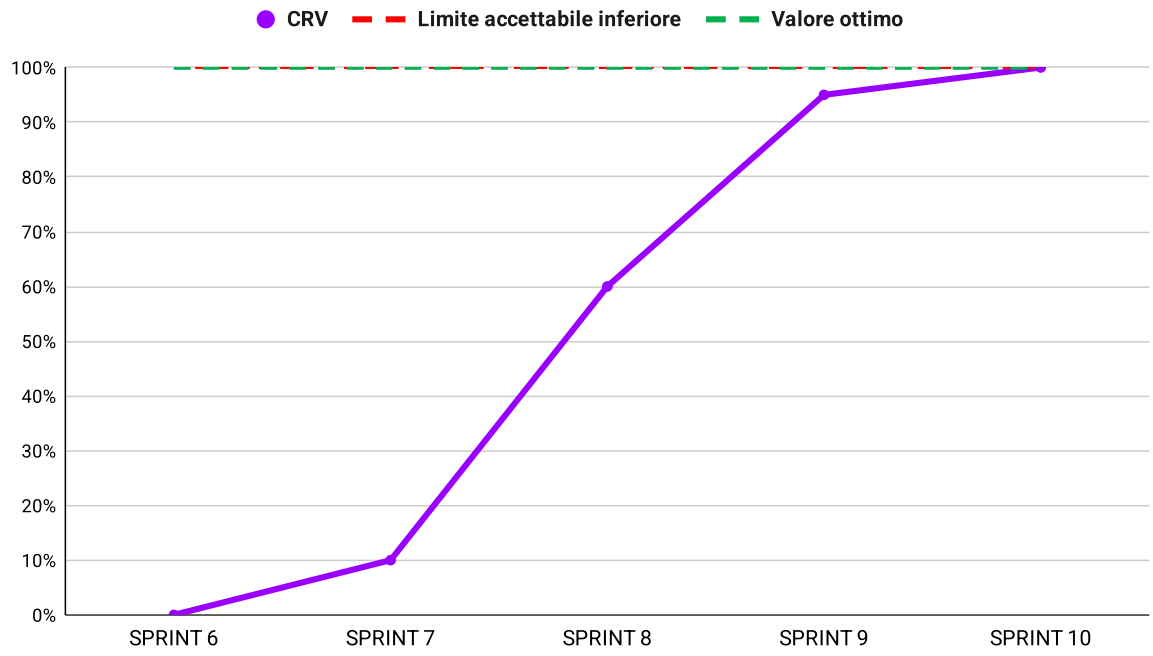


Figura 12: Proiezione della copertura dei requisiti obbligatori (vincolanti) nei vari periodi di progetto.

PB: Osservando il grafico si nota come la copertura dei requisiti obbligatori sia iniziata gradualmente, con i primi completamenti registrati a partire dallo *Sprint*$_G$ 7. Il lavoro su questi requisiti è poi proseguito in modo più intenso nello Sprint successivo, dove si è raggiunto un buon livello di avanzamento. La fase conclusiva del progetto ha visto il completamento quasi totale dei requisiti, portando alla loro piena copertura entro l'ultimo sprint. Questo andamento riflette la scelta del gruppo di concentrarsi inizialmente sulla progettazione dei moduli più critici, per poi procedere con la loro implementazione in modo strutturato e progressivo. La priorità attribuita a questa categoria di requisiti ha permesso di rispettare pienamente gli obiettivi prefissati, assicurandone la completa realizzazione entro i tempi stabiliti.

4.13) `M-PRD-CRD` - Copertura dei requisiti desiderabili

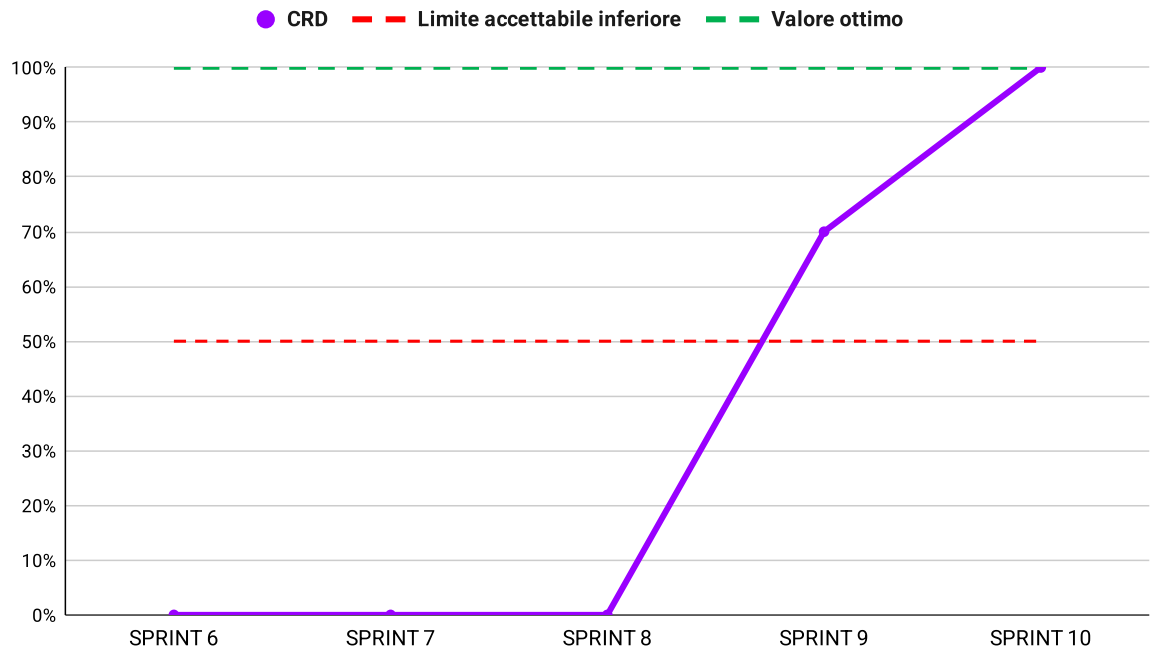


Figura 13: Proiezione della copertura dei requisiti desiderabili nei vari periodi di progetto.

PB: Dall'analisi del grafico emerge che la copertura dei *requisiti*$_G$ desiderabili ha subito un'accelerazione solo nelle fasi finali del progetto. Nelle prime iterazioni, infatti, non è stato possibile dedicare tempo a queste funzionalità, poiché le risorse sono state completamente assorbite dal completamento dei *requisiti*$_G$ obbligatori e dalla definizione dell'architettura del sistema. Solo a partire dagli ultimi *sprint*$_G$ il gruppo è riuscito a concentrarsi su questi aspetti. Questa scelta è stata condivisa con l'azienda, che ha ritenuto opportuno privilegiare obiettivi ritenuti strategicamente più rilevanti per il progetto. Di conseguenza, l'integrazione dei *requisiti*$_G$ desiderabili è avvenuta in modo mirato e ponderato, senza compromettere la qualità complessiva del prodotto.

4.14) `M-PRD-CRO` - Copertura dei requisiti opzionali

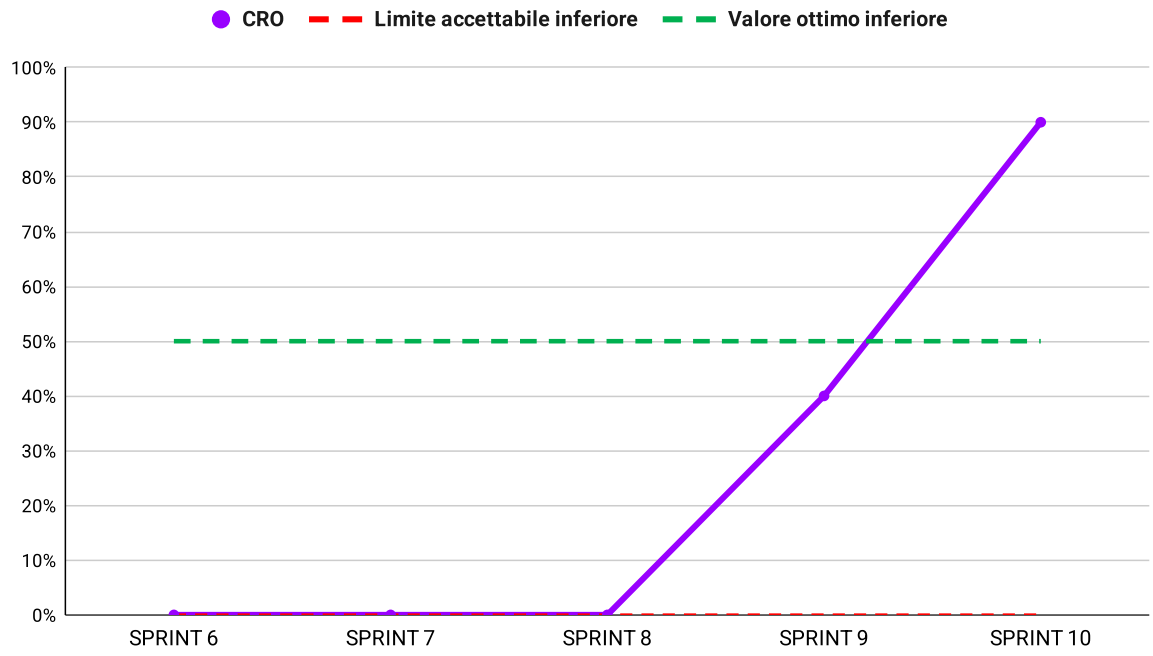


Figura 14: Proiezione della copertura dei requisiti opzionali nei vari periodi di progetto.

PB: Dal grafico si può osservare come la copertura dei *requisiti*$_G$ opzionali sia stata posticipata fino alle fasi conclusive del progetto. Nei primi *sprint*$_G$ successivi alla revisione RTB non è stato possibile dedicare risorse a queste funzionalità, poiché l'attenzione è stata concentrata principalmente sui *requisiti*$_G$ obbligatori e desiderabili. Solo a partire dallo *Sprint*$_G$ 9 è iniziato un lavoro concreto su questa categoria, che ha trovato il suo completamento quasi totale nell'ultimo *sprint*$_G$. La decisione di procedere comunque con l'implementazione di diversi *requisiti*$_G$ opzionali, pur non essendo strettamente necessari, è stata motivata dalla volontà del gruppo di consegnare un prodotto quanto più completo possibile. Questo approccio ci ha permesso di superare il valore ottimale inferiore per la metrica, migliorando ulteriormente la qualità percepita del sistema finale.

4.15) `M-PRD-CC` - Code coverage

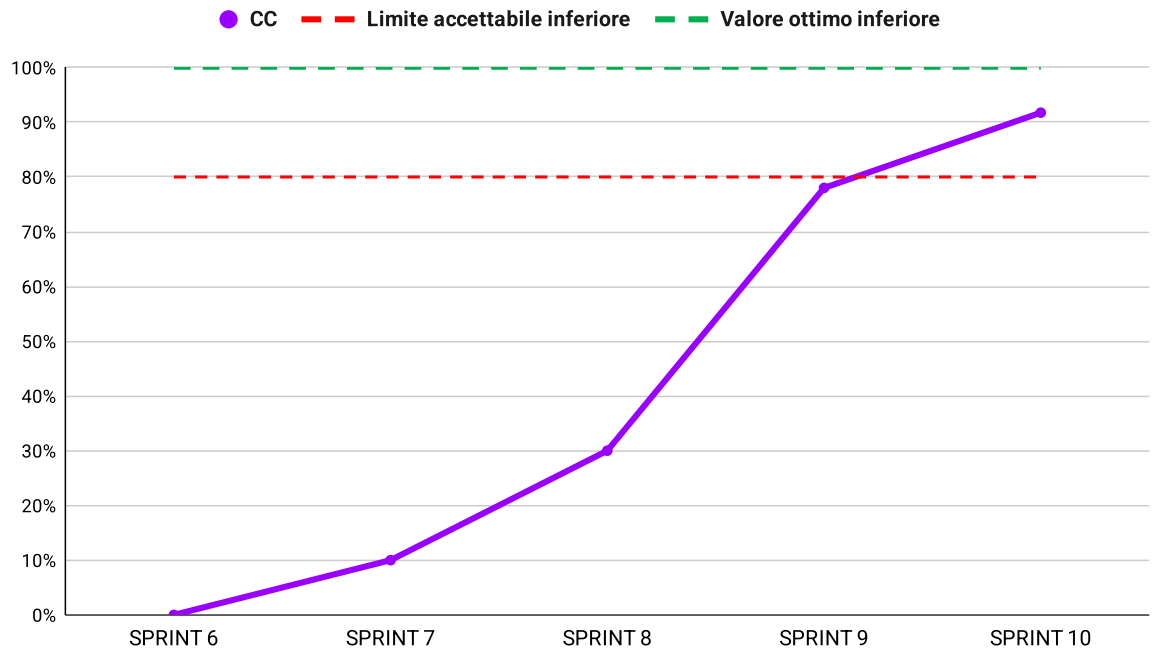


Figura 15: Proiezione della code coverage nei vari periodi di progetto.

PB: Analizzando il grafico, si osserva come la **`code coverage`** abbia registrato una crescita costante a partire dallo *Sprint*$_G$ 7, momento in cui abbiamo dato avvio all'attività di testing, in parallelo allo sviluppo del prodotto. Grazie a una pianificazione attenta e a un progressivo affinamento della strategia di test, siamo riusciti a dedicare sempre più risorse a questa attività, fino a raggiungere, nell'ultimo *sprint*$_G$, un'ottima copertura del codice pari al `91,75%`, superiore al valore minimo richiesto. Riteniamo che questo risultato testimoni l'efficacia del nostro approccio e l'attenzione costante alla qualità del *software*$_G$ lungo tutto il processo di sviluppo.

4.16) `M-PRD-BC` - Branch coverage

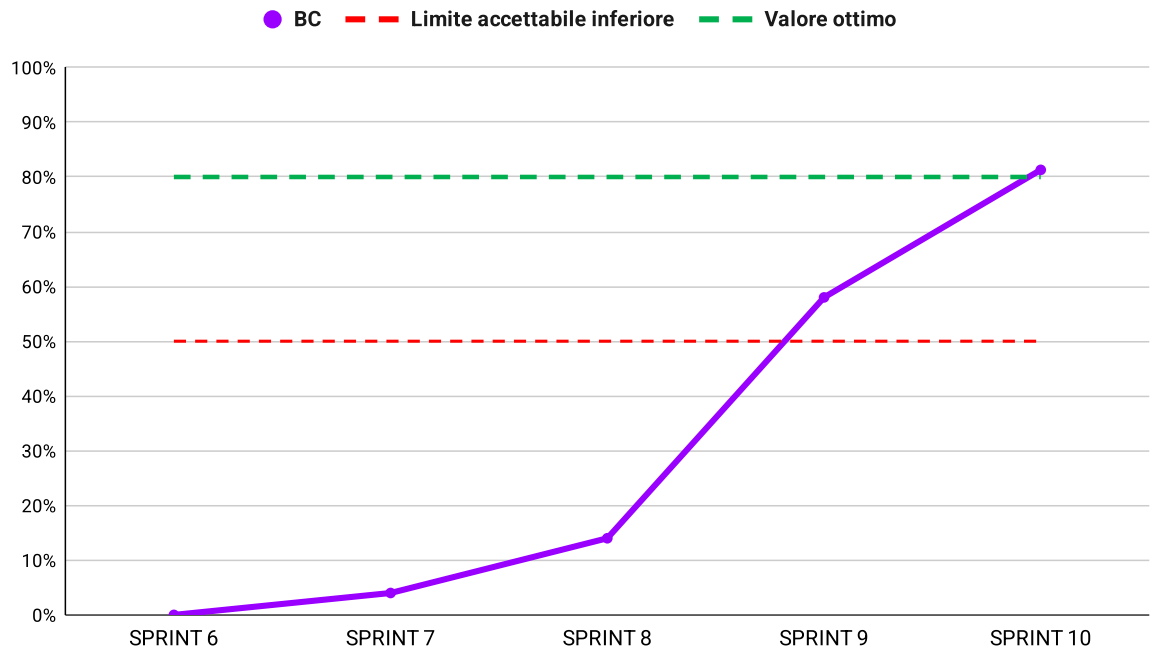


Figura 16: Proiezione della branch coverage nei vari periodi di progetto.

`PB`: Dall'analisi del grafico si osserva come, durante lo *Sprint 6*$_G$, la **`branch coverage`** fosse prossima allo zero, per poi crescere gradualmente e superare il limite accettabile tra lo *Sprint*$_G$ 8 e lo *Sprint*$_G$ 9. Nel corso dello Sprint 10, il valore ha infine raggiunto una soglia prossima all'obiettivo ottimale di circa `80%`. Questo andamento suggerisce che l'introduzione dei test abbia dato risultati tangibili fin dalle prime fasi, consentendo di coprire progressivamente un numero sempre maggiore di diramazioni del flusso d'esecuzione del codice. Il superamento del limite minimo prima e l'avvicinamento al valore ottimale poi evidenziano come la strategia di testing sia stata efficace nel garantire un progressivo aumento della copertura e, di conseguenza, nell'assicurare solidità e affidabilità crescenti al *software*$_G$.

4.17) `M-PRD-FD` - Failure density

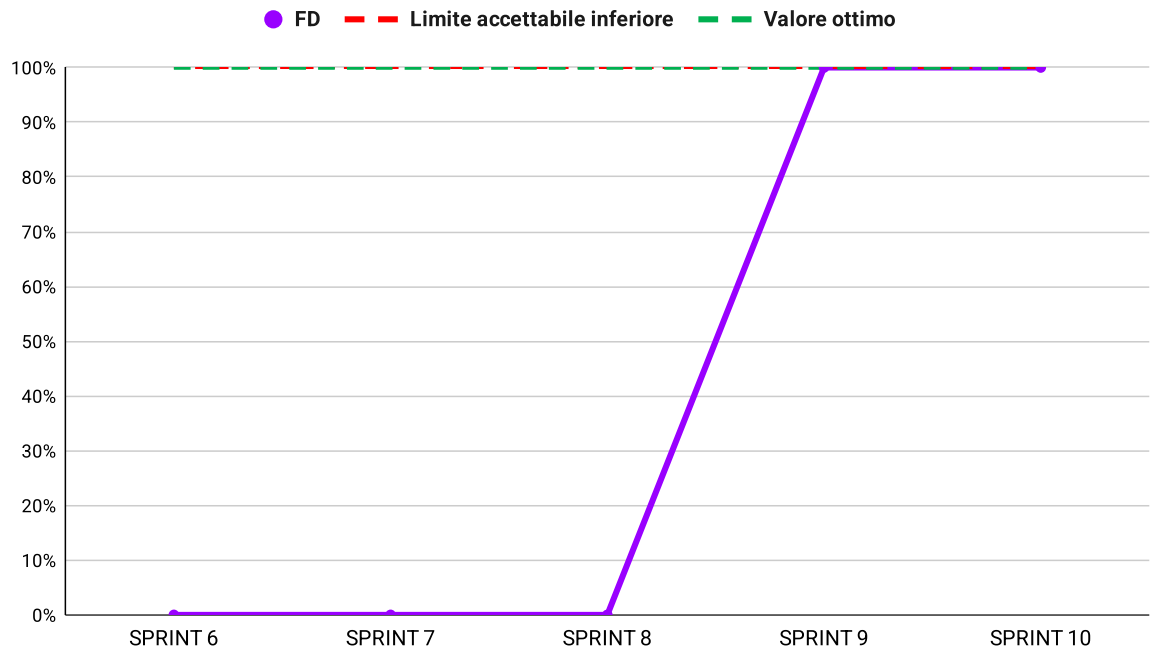


Figura 17: Proiezione della failure density nei vari periodi di progetto.

PB: Dal momento dell'introduzione dei primi test, avvenuta all'inizio dello *Sprint*$_G$ 8, il grafico evidenzia che tutti i difetti del prodotto da noi individuati sono stati rilevati con successo. Questo risultato è particolarmente significativo, poiché dimostra l'efficacia del processo di testing adottato. Riuscire a trovare e correggere i difetti nelle fasi iniziali consente infatti di garantire una maggiore qualità del prodotto finale, riducendo il rischio di errori e contribuendo a una maggiore soddisfazione del cliente. Inoltre, l'individuazione tempestiva dei problemi permette di risparmiare tempo e risorse, evitando che i difetti si accumulino e diventino più complessi da risolvere nelle fasi successive dello sviluppo. Naturalmente, è importante precisare che aver identificato tutti i difetti fin dall'inizio del testing non significa necessariamente che il prodotto sia completamente privo di errori. Anche con un processo di testing accurato, è sempre possibile che alcune problematiche sfuggano al controllo o che ne emergano di nuove a seguito di modifiche o interazioni complesse all'interno del sistema.

4.18) `M-PRD-CS` - Code smell

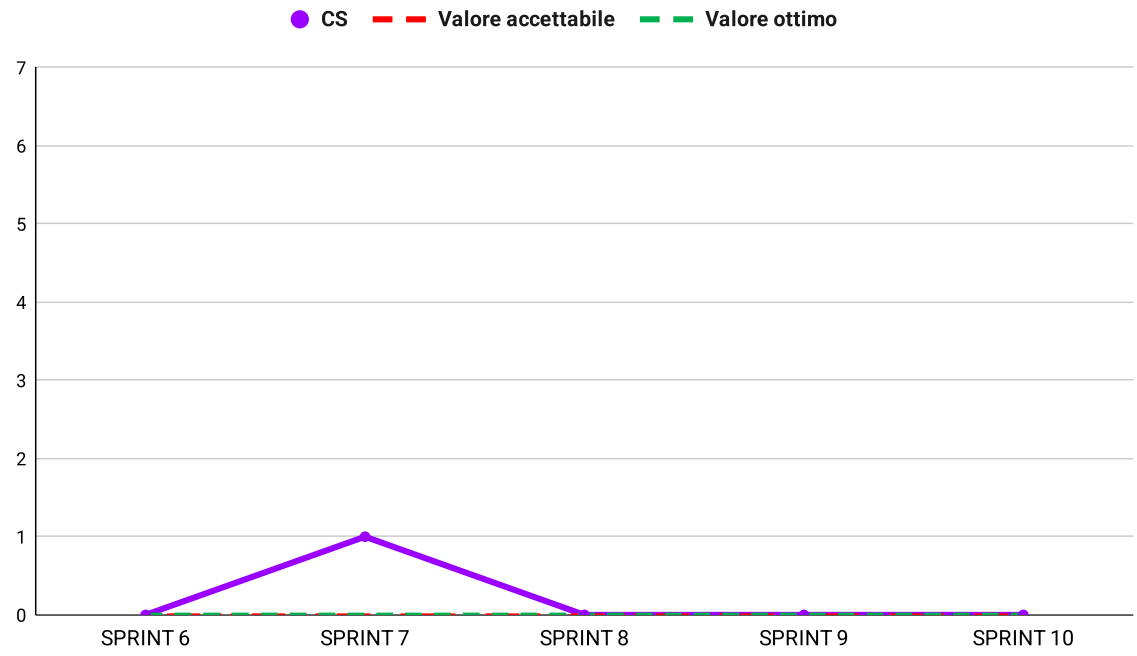


Figura 18: Proiezione del numero di code smell nei vari periodi di progetto.

PB: Il grafico mostra l'andamento del numero di **`code smell`** rilevati durante il progetto, intesi come potenziali segnali di debolezza nella progettazione o nel codice che potrebbero richiedere interventi correttivi. Durante lo *Sprint*$_G$ 7 è emerso un unico **`code smell`** in fase di progettazione, che abbiamo affrontato e risolto prima di procedere con lo sviluppo, anche grazie al confronto con il Prof. Riccardo Cardin. Negli altri *Sprint*$_G$ il numero di **`code smell`** è rimasto sempre pari a zero. Questo andamento evidenzia come l'attenzione posta nelle fasi iniziali ci abbia permesso di mantenere alta la qualità del codice, prevenendo la comparsa di ulteriori criticità e garantendo una maggiore facilità di manutenzione e stabilità del sistema nel tempo.