ONION

SOFTWARE

softwareonion@gmail.com

Piano di qualifica

Informazioni sul documento

Versione	v4.0.0
Data approvazione	2019-07-12
Responsabili	Matteo Lotto
Redattori	Federico Brian
	Federico Omodei
Verificatori	Nicola Pastore
Stato	Approvato
Lista distribuzione	*Imola Informatica S.P.A*
	Onion Software
	prof. Tullio Vardanega
	prof. Riccardo Cardin

Il documento descrive le operazione di verifica, test e validazione eseguite dal gruppo Onion Software durante la realizzazione del progetto Butterfly.

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
v4.0.0	2019-07-12	Matteo Lotto	Responsabile	Documento approvato per il rilascio
v3.1.0	2019-07-11	Nicola Pastore	Verificatore	Verifica superata
v3.0.1	2019-07-10	Federico Omodei	Amministratore	Rettifica documento
v3.0.0	2019-06-08	Linpeng Zhang	Responsabile	Documento approvato per il rilascio
v2.1.0	2019-06-07	Nicola Pastore	Verificatore	Verifica superata
v2.0.3	2019-06-07	Matteo Lotto	Progettista	Modifica e integrazione §A
v2.0.2	2019-05-30	Matteo Lotto	Progettista	Aggiunta test di unità
v2.0.1	2019-05-25	Federico Brian	Progettista	Rettifica documento post RP
v2.0.0	2019-05-09	Nicola Pastore	Responsabile	Documento approvato per il rilascio
v1.2.0	2019-05-09	Alessio Lazzaron	Verificatore	Verifica superata
v1.1.1	2019-05-09	Matteo Lotto	Amministratore	Aggiunta test d'integrazione
v1.1.0	2019-05-08	Alessio Lazzaron	Verificatore	Verifica superata
v1.0.3	2019-05-06	Matteo Lotto	Amministratore	Modifica e integrazione §A
v1.0.2	2019-05-04	Federico Brian	Amministratore	Rettifica documento secondo indicazioni del committente
v1.0.1	2019-05-02	Matteo Lotto	Amministratore	Aggiunta test d'integrazione
v1.0.0	2019-04-10	Federico Brian	Responsabile	Documento approvato per il rilascio
v0.4.0	2019-04-10	Alessio Lazzaron	Verificatore	Verifica superata
v0.3.1	2019-04-10	Nicola Zorzo	Verificatore	Stesura §A.1
v0.3.0	2019-04-08	Linpeng Zhang	Verificatore	Verifica superata del paragrafo §4
v0.2.2	2019-04-07	Nicola Pastore	Amministratore	Stesura §4.1
v0.2.1	2019-04-07	Matteo Lotto	Amministratore	Stesura §4.2
v0.2.0	2019-04-06	Alessio Lazzaron	Verificatore	Verifica superata del paragrafo §3
v0.1.2	2019-04-06	Matteo Lotto	Amministratore	Modifica del paragrafo §3.2
v0.1.1	2019-04-05	Matteo Lotto	Amministratore	Stesura §3.1, §3.2 e §3.3
v0.1.0	2019-04-04	Nicola Zorzo	Verificatore	Verifica superata dei paragrafi §1 e §2
v0.0.3	2019-04-04	Matteo Lotto	Amministratore	Stesura §2
v0.0.2	2019-04-01	Nicola Pastore	Amministratore	Stesura §1
v0.0.1	2019-03-28	Nicola Pastore	Amministratore	Creazione del documento LaTeX, definizione scheletro

Indice

1 Introduzione **1**
- 1.1 Premessa . . . 1
- 1.2 Scopo del documento . . . 1
- 1.3 Scopo del prodotto . . . 1
- 1.4 Note esplicative . . . 1
- 1.5 Riferimenti . . . 1
  - 1.5.1 Riferimenti normativi . . . 1
  - 1.5.2 Riferimenti informativi . . . 1

2 Qualità del processo **3**
- 2.1 Scopo . . . 3
- 2.2 Processo di pianificazione . . . 3
  - 2.2.1 Obiettivi . . . 3
- 2.3 Processo di verifica del software . . . 3
  - 2.3.1 Obiettivi . . . 3

3 Qualità del prodotto **5**
- 3.1 Scopo . . . 5
- 3.2 Produzione della documentazione . . . 5
  - 3.2.1 Obiettivi . . . 5
- 3.3 Produzione del software . . . 5
  - 3.3.1 Obiettivi . . . 5

4 Specifica dei test **7**
- 4.1 Test di accettazione . . . 7
- 4.2 Test di sistema . . . 12
- 4.3 Test di integrazione . . . 29
- 4.4 Test di unità . . . 31

A Resoconto dell'attività di verifica **46**
- A.1 Revisione dei Requisiti . . . 46
  - A.1.1 Analisi sui documenti . . . 46
    - A.1.1.1 MPDD01: Indice Gulpease . . . 46
  - A.1.2 Analisi sui processi . . . 47
    - A.1.2.1 MPR01: Schedule Variance . . . 47
    - A.1.2.2 MPR02: Budget Variance . . . 48
- A.2 Revisione di Progetto . . . 49
  - A.2.1 Analisi sui documenti . . . 49
    - A.2.1.1 MPDD01: Indice Gulpease . . . 49
  - A.2.2 Analisi sui processi . . . 50
    - A.2.2.1 MPR01: Schedule Variance . . . 50
    - A.2.2.2 MPR02: Budget Variance . . . 51
- A.3 Revisione di Qualifica . . . 52
  - A.3.1 Considerazioni sulle metriche utilizzate . . . 52
  - A.3.2 Analisi sui documenti . . . 53
    - A.3.2.1 MPDD01: Indice Gulpease . . . 53
  - A.3.3 Analisi sui processi . . . 54
    - A.3.3.1 MPR01: Schedule Variance . . . 54
    - A.3.3.2 MPR02: Budget Variance . . . 54
    - A.3.3.3 MPRS01: Code Coverage . . . 55
  - A.3.4 Analisi sul prodotto . . . 57
    - A.3.4.1 MPDS01: Copertura requisiti obbligatori . . . 57
    - A.3.4.2 MPDS02: Copertura requisiti accettati . . . 57

A.3.4.3 MPDS09: Accoppiamento di componenti . . . 58
A.3.4.4 MPDS10: Interfaccia utente auto-esplicativa . . . 58
A.3.4.5 MPDS06: Densità errori . . . 59
A.3.4.6 MPDS07: Capacità di analisi degli errori . . . 59
A.3.4.7 MPDS08: Efficienza delle modifiche . . . 60
A.3.4.8 MPDS12: Duplicazione del codice . . . 60
A.3.5 Analisi sui test . . . 61
A.3.5.1 TS01: Test passati . . . 61
A.3.5.2 TS02: Test falliti . . . 61
A.3.5.3 TS03: Test eseguiti in rapporto ai requisiti . . . 62
A.4 Revisione di Accettazione . . . 63
A.4.1 Considerazioni sulle metriche utilizzate . . . 63
A.4.2 Analisi sui documenti . . . 64
A.4.2.1 MPDD01: Indice Gulpease . . . 64
A.4.3 Analisi sui processi . . . 65
A.4.3.1 MPR01: Schedule Variance . . . 65
A.4.3.2 MPR02: Budget Variance . . . 65
A.4.3.3 MPRS01: Code Coverage . . . 66
A.4.4 Analisi sul prodotto . . . 68
A.4.4.1 MPDS01: Copertura requisiti obbligatori . . . 68
A.4.4.2 MPDS02: Copertura requisiti accettati . . . 68
A.4.4.3 MPDS09: Accoppiamento di componenti . . . 69
A.4.4.4 MPDS10: Interfaccia utente auto-esplicativa . . . 69
A.4.4.5 MPDS06: Densità errori . . . 70
A.4.4.6 MPDS07: Capacità di analisi degli errori . . . 70
A.4.4.7 MPDS08: Efficienza delle modifiche . . . 71
A.4.4.8 MPDS12: Duplicazione del codice . . . 71
A.4.4.9 MPDS03: Accuratezza rispetto alle attese . . . 72
A.4.4.10 MPDS04: Tempo medio di risposta . . . 72
A.4.4.11 MPDS11: Comprensibilità dei messaggi d'errore . . . 73
A.4.5 Analisi sui test . . . 73
A.4.5.1 TS01: Test passati . . . 73
A.4.5.2 TS02: Test falliti . . . 74
A.4.5.3 TS03: Test eseguiti in rapporto ai requisiti . . . 74

Elenco delle figure

2 Esito schedule variance prima del RR . . . 48
3 Esiti del Budget Variance prima del RR . . . 48
5 Esito schedule variance prima del RP . . . 51
6 Esiti del Budget Variance prima del RP . . . 51
8 Esito schedule variance prima del RQ . . . 54
9 Esiti del Budget Variance prima del RQ . . . 55
10 Esito code coverage . . . 55
12 Esito copertura requisiti . . . 57
13 Esito copertura requisiti accettati . . . 57
14 Esito accoppiamento componenti . . . 58
15 Esito interfaccia auto-esplicativa . . . 58
16 Esito densità errori . . . 59
17 Esito analisi errori . . . 59
18 Esito efficienza modifiche . . . 60
19 Esito efficienza modifiche . . . 60
20 Test passati . . . 61
21 Test falliti . . . 61

22 Test eseguiti in rapporto ai requisiti . . . 62
24 Esito schedule variance prima del RA . . . 65
25 Esiti del Budget Variance prima del RA . . . 66
26 Esito code coverage . . . 66
28 Esito copertura requisiti . . . 68
29 Esito copertura requisiti accettati . . . 68
30 Esito accoppiamento componenti . . . 69
31 Esito interfaccia auto-esplicativa . . . 69
32 Esito densità errori . . . 70
33 Esito analisi errori . . . 70
34 Esito efficienza modifiche . . . 71
35 Esito efficienza modifiche . . . 71
36 Esito accuratezza rispetto alle attese . . . 72
37 Esito tempo medio di risposta . . . 72
38 Esito comprensibilità dei messaggi . . . 73
39 Test passati . . . 73
40 Test falliti . . . 74
41 Test eseguiti in rapporto ai requisiti . . . 74

Elenco delle tabelle

1 Parametri per le metriche dei processi di pianificazione . . . 3
2 Parametri per le metriche per il processo di verifica del software . . . 4
3 Parametri per le metriche della documentazione . . . 5
4 Parametri per le metriche per la produzione del software . . . 6
5 Parametri per le metriche dei test . . . 7
6 Tabella dei test di accettazione . . . 7
7 Tabella dei test di sistema . . . 12
8 Tabella dei test di integrazione . . . 29
9 Tabella dei test di integrazione . . . 31
10 Esiti indici di Gulpease . . . 46
11 Esiti indici dello Schedule Variance . . . 47
12 Esiti indice del Budget Variance . . . 48
13 Esiti indici di Gulpease . . . 49
14 Esiti indici dello Schedule Variance . . . 50
15 Esiti indice del Budget Variance . . . 51
16 Esiti indici di Gulpease . . . 53
17 Esiti indici dello Schedule Variance . . . 54
18 Esiti indice del Budget Variance . . . 54
19 Esiti indici di Gulpease . . . 64
20 Esiti indici dello Schedule Variance . . . 65
21 Esiti indice del Budget Variance . . . 65

1 Introduzione

1.1 Premessa

Poiché molti contenuti del documento *piano di qualifica* sono mutevoli, si prevede di lavorare sul documento per l'intera durata del progetto. In particolare, alcune parti saranno prodotte in fasi temporali differenti e per garantire la qualità del documento il gruppo ha deciso di adottare un modello incrementale. Ne segue che i contenuti, nelle fasi iniziali, potrebbero essere incompleti, e prossimi a significative aggiunte o modifiche nel tempo.

1.2 Scopo del documento

Il fine del documento è elencare le tecniche utilizzate per la verifica e la validazione dei prodotti e dei processi. Il sistema di verifica descritto successivamente sarà applicato durante l'intero svolgimento del progetto e, pertanto, il documento sarà costantemente aggiornato. Ciò è vantaggioso perché:

- consente di individuare gli errori e risanarli in modo tempestivo, minimizzando l'uso di risorse;
- permette di monitorare costantemente la qualità del prodotto in maniera quantificabile.

1.3 Scopo del prodotto

Il proponente richiede di far fronte alla poca flessibilità dei molteplici strumenti utilizzati per i processi di *continuous integration*$_G$ e *continuous delivery*$_G$ in una realtà enterprise di grandi dimensioni. Tali strumenti forniscono già dei meccanismi di segnalazione di messaggi ad eventuali problematiche riscontrate nel processo, però ognuno di essi ne ha uno proprio e, spesso, con scarse possibilità di configurazione. Da questi fattori deriva la necessità per l'utente di interfacciarsi con molteplici *dashboard*$_G$, ognuna con la propria struttura ed interfaccia, gravando sull'accessibilità. La finalità del prodotto è dunque di accentrare e rendere standard queste segnalazioni, oltre a permetterne una gestione automatizzata e personalizzata.

1.4 Note esplicative

Al fine di evitare possibili ambiguità relative al linguaggio utilizzato nei documenti formali, viene fornito il *glossario v2.0.0*. In questo documento vengono definiti e descritti tutti i termini con un significato specifico. Per facilitarne la comprensione, questi termini saranno contrassegnati, alla prima occorrenza, da una 'G' a pedice e da uno stile corsivo.

1.5 Riferimenti

1.5.1 Riferimenti normativi

- **Nome di progetto**: *Norme di progetto v2.0.0*

1.5.2 Riferimenti informativi

- **Capitolato d'appalto C1 - Butterfly, monitor per processi CI/CD**: https://www.math.unipd.it/~tullio/IS-1/2018/Progetto/C1.pdf;
- **Slide L13 del corso Ingegneria del Software - Qualità del software**: https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L13.pdf;
- **Slide L14 del corso Ingegneria del Software - Qualità di processo**: https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L14.pdf;

- **Slide L16 del corso Ingegneria del Software - Verifica e validazione: introduzione**:
  https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L16.pdf;
- **Slide L17 del corso Ingegneria del Software - Verifica e validazione: analisi statica**:
  https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L17.pdf;
- **Slide L18 del corso Ingegneria del Software - Verifica e validazione: analisi dinamica**:
  https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L18.pdf;
- **Standard ISO/IEC 9126**:
  https://it.wikipedia.org/wiki/ISO/IEC_9126;
- **ISO/IEC 12207:1995**:
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
- **Indice di Gulpease**:
  https://it.wikipedia.org/wiki/Indice_Gulpease
- **Schedule Variance**
  https://www.smartsheet.com/hacking-pmp-how-calculate-schedule-variance

2 Qualità del processo

2.1 Scopo

Per garantire qualità al prodotto è fondamentale perseguire la qualità dei processi che lo definiscono. Per raggiungere tale scopo, è stato deciso di seguire il principio di miglioramento continuo $PDCA_{\boldsymbol{G}}$ e di adottare lo standard ISO/IEC 15504 denominato SPICE (Software Process Improvement and Capability Determination), opportunamente enunciati nel documento *norme di progetto*.

2.2 Processo di pianificazione

Il processo ha il fine di raggruppare i piani di sviluppo per il progetto, comprendenti la descrizione delle attività, dei compiti da svolgere e della loro suddivisione, inoltre è da considerare anche la scelta del modello di ciclo di vita del progetto e la pianificazione temporale del lavoro e il corrispettivo costo di produzione.

2.2.1 Obiettivi

Durante lo sviluppo del progetto si dovrà porre particolare attenzione a rispettare i seguenti parametri:

- **budget**: al fine di non avere scarti eccessivi con il costo preventivato, si devono utilizzare le metriche prestabilite;
- **formazione**: ogni membro del gruppo deve avere almeno un livello base di preparazione per l'esecuzione dei task assegnati, per evitare ritardi nella produzione;
- **standard**: definire lo standard di processo ogni qualvolta sia possibile, per facilitare l'incremento continuativo;
- **calendario**: garantire una pianificazione adatta ai compiti da svolgere, per massimizzare l'efficienza della produzione.

Le metriche prestabilite nelle *norme di progetto* sono:

Metrica	Nome	Valore di accettazione	Valore ideale
MPR01	Schedule Variance	≥ -5 giorni	≥ 0 giorni
MPR02	Budget Variance	≤ 5.4	≤ 4.6

Tabella 1: Parametri per le metriche dei processi di pianificazione

2.3 Processo di verifica del software

Il processo punta a verificare se qualsiasi elemento del sistema soddisfa completamente i requisiti ad esso assegnati.

2.3.1 Obiettivi

È necessario che il codice venga costantemente verificato:

- **commenti**: ogni segmento di codice deve essere adeguatamente commentato per essere ritenuto verificabile;
- **bug**: controllare la presenza di bug prima dell'utilizzo del codice.

Per verificare il codice vengono utilizzate le metriche prestabilite nelle *norme di progetto*, ossia:

Metrica	Nome	Valore di accettazione	Valore ideale
MPRS01	Code Coverage	$\geq 80\%$	100%
MPRS02	Function coverage	$\geq 98\%$	100%
MPRS03	Statement coverage	$\geq 97\%$	100%
MPRS04	$Branch_{\boldsymbol{G}}$ coverage	$\geq 95\%$	100%
MPRS05	Condition coverage	$\geq 95\%$	100%
MPRS06	Lines coverage	$\geq 99\%$	100%

Tabella 2: Parametri per le metriche per il processo di verifica del software

3 Qualità del prodotto

3.1 Scopo

Il gruppo ha deciso di utilizzare lo standard ISO/IEC 9126 come riferimento informativo per quanto riguarda la qualità del prodotto. Questo standard definisce i requisiti che un prodotto deve soddisfare per essere considerato di qualità.
Sono state utilizzate metriche adeguate al fine di garantire qualità per il prodotto, descritte nelle *norme di progetto*. In questa sezione vengo enunciati i parametri che quantificano il grado di raggiungimento qualitativo desiderato.

3.2 Produzione della documentazione

I documenti prodotti dal gruppo Onion Software dovranno essere leggibili, comprensibili e corretti dal punto di vista ortografico, sintattico, logico e semantico.

3.2.1 Obiettivi

I documenti prodotti da Onion Software dovranno rispettare:

- una facile comprensione per persone con un minimo grado di diploma di scuola superiore di secondo grado;
- una correttezza grammaticale e coesione.

Le metriche da misurare, definite nelle *norme di progetto*, sono le seguenti: Le metriche prestabilite nelle *norme di progetto* sono:

Metrica	Nome	Valore di accettazione	Valore ideale
MPDD01	*Indice Gulpease*$_G$	≥ 35	≥ 40
MPDD02	Errori ortografici	≤ 4	$= 0$

Tabella 3: Parametri per le metriche della documentazione

3.3 Produzione del software

L'obbiettivo finale del progetto, e il conseguente prodotto che viene consegnato e utilizzato dal cliente è il prodotto software. Di conseguenza è di grande importanza garantire un buon livello di qualità.

3.3.1 Obiettivi

Il software finale dovrà essere sviluppato puntando a rispettare:

- i principi *SOLID*$_G$ per garantire un programma mantenibile e flessibile;
- un facile e rapido utilizzo del software per gli utenti.

Gli obiettivi da seguire per mantenere la qualità del prodotto software sono i seguenti:

Metrica	Nome	Valore di accettazione	Valore ideale
MPDS01	Copertura requisiti obbligatori	100%	100%
MPDS02	Copertura requisiti accettati	$\geq 60\%$	100%
MPDS03	Accuratezza rispetto alle attese	$\geq 60\%$	$\geq 80\%$
MPDS04	Tempo medio di risposta	$\leq 5s$	$\leq 1s$
MPDS05	Adeguatezza del tempo di risposta	$\geq 40\%$	$\geq 50\%$
MPDS06	Densità errori	$\leq 30\%$	$\leq 10\%$
MPDS07	Capacità di analisi degli errori	$\geq 40\%$	$\geq 50\%$
MPDS08	Efficienza delle modifiche	$\geq 80\%$	100%
MPDS09	Accoppiamento di componenti	≤ 20	≤ 10
MPDS10	Interfaccia utente auto-esplicativa	$\geq 60\%$	$\geq 80\%$
MPDS11	Comprensibilità dei messaggi d'errore	$\geq 70\%$	$\geq 90\%$
MPDS12	Duplicazione del codice	$\leq 20\%$	$\leq 0\%$

Tabella 4: Parametri per le metriche per la produzione del software

4 Specifica dei test

Per garantire una gestione efficacie dell'analisi dinamica è fondamentale stabilire delle misurazioni sull'esecuzione del programma. Va considerato che questa sezione verrà assolutamente ampliata o modificata in quanto si ritiene prematuro affrontare l'argomento di metriche di test; di conseguenza sono elencate di seguito le metriche ritenute essenziali definite nelle *norme di progetto*.

Metrica	Nome	Valore accettazione	Valore ideale
TS01	percentuale di test passati	80%	100%
TS02	percentuale di test falliti	20%	0%
TS03	totalità dei test eseguiti in rapporto ai requisiti	100%	100%

Tabella 5: Parametri per le metriche dei test

Per permettere al lettore la comprensione dello stato di avanzamento dei vari test viene riportata di seguito una legenda, nella quale sono descritte in dettaglio le sigle utilizzate, questa legenda è presente anche nel documento *norme di progetto* ma per comodità viene riscritta anche in questa sezione. Vengono utilizzate le seguenti sigle:

- **IP**: indica che il test è implementato;
- **NIP**: indica che il test non è implementato;
- **V**: indica che il test è stato verificato e quindi soddisfa la richiesta;
- **NV**: indica che il test non è stato verificato.

4.1 Test di accettazione

I test di accettazione devono dimostrare che il software sviluppato soddisfi i requisiti del capitolato concordati con il proponente; essi vengono eseguiti durante il collaudo finale. Si compongono essenzialmente dei test di sistema però eseguiti in sede aziendale.

Tabella 6: Tabella dei test di accettazione

Codice	Requisito	Descrizione	Stato
TAOF01	RFO1 RFO1.1 RFO1.1.1 RFO1.1.2 RFO1.1.3 RFO1.1.4 RFO1.1.5 RFO1.1.6 RFO1.1.7	l'utente inserisce un nuovo contatto. Tale operazione richiede necessariamente email, nome, cognome. Inoltre si inseriscono l'username associato a Telegram, l'email associata a Slack, le preferenze sulla ricezione delle notifiche ed i giorni di indisponibilità.	NIP

– *continua a pagina successiva*

Tabella 6– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TAOF02	RFO1 RFO1.2 RFO1.2.1 RFO1.2.2 RFO1.2.3 RFO1.2.4 RFO1.2.5 RFO1.2.6 RFO1.2.7	l'utente modifica i dati di un contatto presente nel gestore personale. In particolare, può modificare: email, nome, cognome, username associato a Telegram, email associata a Slack, preferenze sulla ricezione delle notifiche e giorni di indisponibilità.	NIP
TAOF03	RFO1 RFO1.3	l'utente cancella un contatto presente nel gestore personale.	NIP
TAOF04	RFO2 RFO2.1 RFO2.1.1 RFO2.1.2 RFO2.1.3	l'utente visualizza la lista dei contatti esistenti presenti nel gestore personale. Per ognuno di essi visualizza email identificativa, nome e cognome.	NIP
TAOF05	RFO3 RFO3.1 RFO3.2 RFO3.3 RFO3.4 RFO3.5 RFO3.6 RFO3.7	l'utente effettua la lettura dei dati di un contatto presente nel gestore personale visualizzando email identificativa, nome, cognome, username associato a Telegram, email associata a Slack, preferenze sulla ricezione delle notifiche e giorni di indisponibilità.	NIP
TAOF06	RFO4 RFO4.1 RFO4.1.1 RFO4.1.2	l'utente inserisce nel sistema un nuovo progetto fornendo obbligatoriamente l'url della repository di tale progetto e il nome.	NIP
TAOF07	RFO4 RFO4.2 RFO4.2.1	l'utente modifica i dati riguardanti un progetto presente nel gestore personale. In particolare, può modificare il nome.	NIP
TAOF08	RFO4 RFO4.3	l'utente rimuove un progetto memorizzato in precedenza nel gestore personale.	NIP
TAOF09	RFO5 RFO5.1 RFO5.1.1 RFO5.1.2	l'utente visualizza la lista di tutti i progetti presenti nel gestore personale. Per ognuno di essi visualizza nome e url.	NIP
TAOF10	RFO6 RFO6.1 RFO6.2 RFO6.3	l'utente effettua la lettura dei dati di un singolo progetto presente nel gestore personale, visualizzando nome, url e numero di contatti iscritti al progetto.	NIP
TAOF11	RFO7 RFO7.1 RFO7.1.1 RFO7.1.2 RFO7.1.3 RFO7.1.4	l'utente iscrive un utente esistente ad uno specifico progetto specificando l'email del contatto da inserire, l'url del progetto e la priorità associata allo specifico progetto per quel contatto. Opzionalmente può fornire una lista di keyword associate al progetto.	NIP

– *continua a pagina successiva*

Tabella 6– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TAOF12	RFO7 RFO7.2 RFO7.2.1 RFO7.2.2 RFO7.2.3 RFO7.2.4	l'utente modifica l'iscrizione di un contatto ad un progetto. In particolare può cambiare: l'email del contatto, l'url del progetto, il livello di priorità e la lista di keyword.	NIP
TAOF13	RFO7 RFO7.3	l'utente cancella l'iscrizione ad un progetto di un contatto presente nel gestore personale.	NIP
TAOF14	RFO8 RFO8.1 RFO8.1.1 RFO8.1.2 RFO8.1.3 RFO8.1.4	l'utente deve poter vedere la lista di tutte le iscrizione esistenti nel gestore personale. Per ognuna di esse visualizza il nome ed il cognome del contatto, il nome del progetto associato e la priorità dell'iscrizione.	NIP
TAOF15	RFO9 RFO9.1 RFO9.2 RFO9.3 RFO9.4 RFO9.5 RFO9.6	l'utente effettua la lettura dei dati di una singola iscrizione nel gestore personale visualizzando email, nome e cognome del contatto, nome del progetto, grado di priorità e lista di keyword.	NIP
TAOF16	RFO10	l'utente, utilizzando una keyword, ricerca informazioni specifiche riguardanti uno o più contatti e li ordina sotto forma di lista.	NIP
TAOF17	RFO11	l'utente, utilizzando una keyword, ricerca informazioni specifiche riguardanti uno o più progetti e li riordina sotto forma di lista.	NIP
TAOF18	RFO12	l'utente, utilizzando una keyword, ricerca informazioni specifiche riguardanti uno o più iscrizioni e le riordina sotto forma di lista.	NIP
TAOF19	RFO13 RFO18 RFO19 RFO20 RFO33 RFO34 RFO35 RFO36 RFO37 RFO38 RFO39 RFO56 RFO57 RFO58 RFO59 RFO60	Il sistema deve mandare un messaggio di errore quando, durante l'inserimento di un nuovo contatto, i dati inseriti sono errati.	NIP

– *continua a pagina successiva*

Tabella 6– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TAOF20	RFO14 RFO21 RFO22 RFO23 RFO40 RFO41 RFO42 RFO43 RFO44 RFO45 RFO46 RFO61 RFO62 RFO63 RFO64 RFO65	Il sistema deve mandare un messaggio di errore quando, durante la modifica di un contatto esistente, i dati inseriti sono errati.	NIP
TAOF21	RFO15 RFO24 RFO25 RFO47 RFO48 RFO66 RFO67 RFO77	Il sistema deve mandare un messaggio di errore quando, durante l'inserimento di un nuovo progetto, i dati inseriti sono errati.	NIP
TAOF22	RFO26 RFO49 RFO68	Il sistema deve mandare un messaggio di errore quando, durante la modifica di un progetto esistente, i dati inseriti sono errati.	NIP
TAOF23	RFO16 RFO27 RFO28 RFO29 RFO50 RFO51 RFO52 RFO69 RFO70 RFO73 RFO75	Il sistema deve mandare un messaggio di errore quando, durante l'inserimento di una nuova iscrizione, i dati inseriti sono errati.	NIP
TAOF24	RFO17 RFO30 RFO31 RFO32 RFO53 RFO54 RFO55 RFO71 RFO72 RFO74 RFO76	Il sistema deve mandare un messaggio di errore quando, durante la modifica di una iscrizione esistente, i dati inseriti sono errati.	NIP

– *continua a pagina successiva*

Tabella 6– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TAOF25	RFO78 RFO79 RFO88	GitLab invia una notifica a GitLabProducer segnalando che è avvenuta una operazione di push.	NIP
TAOF26	RFO78 RFO80 RFO88	GitLab invia una notifica a GitLabProducer segnalando che è stata creata una nuova issue.	NIP
TAOF27	RFO78 RFO81 RFO88	GitLab invia una notifica a GitLabProducer segnalando che è stata modificata una issue.	NIP
TAOF28	RFO82 RFO83 RFO89	Redmine invia una notifica a RedmineProducer segnalando che è stata creata una nuova issue.	NIP
TAOF29	RFO82 RFO84 RFO89	Redmine invia una notifica a RedmineProducer segnalando che è stata modificata una issue.	NIP
TAOF30	RFO82 RFO85 RFO89	Redmine invia una notifica a RedmineProducer segnalando che è stata inserita una priorità per una issue.	NIP
TAOF31	RFO86 RFO89	Redmine invia una notifica a RedmineProducer segnalando che è stata modificata la priorità per una issue.	NIP
TAOF32	RFO87 RFO90	SonarQube invia una notifica al componente SonarQubeProducer elaborandola e inviando le informazioni opportune al broker.	NIP
TAOF33	RFO91	il componente TelegramConsumer invia un pacchetto contenente le informazioni correlate ad un evento ad uno o più contatti Telegram.	NIP
TAOF34	RFO92	il componente EmailConsumer invia un pacchetto contenente le informazioni correlate ad un evento ad uno o più indirizzi email.	NIP
TAOF35	RFO93	il componente SlackConsumer invia un pacchetto contenente le informazioni correlate ad un evento ad uno o più contatti Slack.	NIP

4.2 Test di sistema

Sono impiegati per garantire il corretto funzionamento di tutte le componenti del sistema nel loro insieme.

Tabella 7: Tabella dei test di sistema

Codice	Requisito	Descrizione	Stato
TSOF01	RFO1 RFO1.1 RFO1.1.1 RFO1.1.2 RFO1.1.3 RFO1.1.4 RFO1.1.5 RFO1.1.6 RFO1.1.7	• **Obiettivo**: l'utente inserisce un nuovo contatto. Tale operazione richiede necessariamente email, nome, cognome. Opzionalmente si possono inserire il numero associato a Telegram, l'email associata a Slack, le preferenze sulla ricezione delle notifiche ed i giorni di indisponibilità. • **Procedura**: – l'utente inserisce un nome per il nuovo contatto; – l'utente inserisce un cognome per il nuovo contatto; – l'utente inserisce un indirizzo email per il nuovo contatto; – l'utente può anche inserire numero cellulare associato Telegram, email associata a Slack, preferenze sulla ricezione di modifiche e giorni d'inattività; – l'utente invia le informazioni inserite al sistema; – il sistema le accetta.	V
TSOF02	RFO1 RFO1.2 RFO1.2.1 RFO1.2.2 RFO1.2.3 RFO1.2.4 RFO1.2.5 RFO1.2.6 RFO1.2.7	• **Obiettivo**: l'utente modifica i dati di un contatto presente nel gestore personale. • **Procedura**: – l'utente può modificare i seguenti campi dati del contatto: ∗ nome; ∗ cognome; ∗ email; ∗ preferenze; ∗ giorni di inattività; ∗ numero cellulare associato a Telegram; ∗ indirizzo email associato a Slack; – l'utente invia le informazioni modificate al sistema; – il sistema le accetta.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF03	RFO1 RFO1.3	• **Obiettivo**: l'utente cancella un contatto presente nel gestore personale. • **Procedura**: – l'utente invia al sistema il contatto da cancellare; – il sistema lo cancella.	V
TSOF04	RFO2 RFO2.1 RFO2.1.1 RFO2.1.2 RFO2.1.3	• **Obiettivo**: l'utente visualizza, anche in modo personalizzato, la lista dei contatti esistenti presenti nel gestore personale. • **Procedura**: – il sistema restituisce nome, cognome e email dei contatti. – l'utente visualizza la lista dei contatti;	V
TSOF05	RFO3 RFO3.1 RFO3.2 RFO3.3 RFO3.4 RFO3.5 RFO3.6 RFO3.7	• **Obiettivo**: l'utente effettua la lettura dei dati di un utente presente nel gestore personale. • **Procedura**: – il sistema restituisce le seguenti informazioni del contatto: ∗ nome; ∗ cognome; ∗ email; ∗ preferenze(se presenti); ∗ giorni di inattività(se presenti); ∗ numero cellulare associato a Telegram(se presente); ∗ indirizzo email associato a Slack(se presente); – l'utente visualizza i dettagli del contatto.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF06	RFO4 RFO4.1 RFO4.1.1 RFO4.1.2	• **Obiettivo**: l'utente inserisce nel sistema un nuovo progetto fornendo obbligatoriamente l'url della repository di tale progetto e il nome. • **Procedura**: – l'utente inserisce un indirizzo url di un progetto Redmine o Gitlab esistente per il nuovo progetto; – l'utente inserisce un nome per il nuovo progetto; – l'utente invia al sistema il nuovo progetto; – il sistema lo accetta.	V
TSOF07	RFO4 RFO4.2 RFO4.2.1	• **Obiettivo**: l'utente modifica i dati riguardanti un progetto presente nel gestore personale. • **Procedura**: – l'utente modifica il nome del progetto; – l'utente invia al sistema il nome modificato o meno; – il sistema lo accetta.	V
TSOF08	RFO4 RFO4.3	• **Obiettivo**: l'utente rimuove un progetto memorizzato in precedenza nel gestore personale. • **Procedura**: – l'utente invia al sistema il progetto memorizzato da cancellare; – il sistema lo cancella.	V
TSOF09	RFO5 RFO5.1 RFO5.1.1 RFO5.1.2	• **Obiettivo**: l'utente visualizza la lista di tutti i progetti presenti nel gestore personale anche in modo personalizzato. • **Procedura**: – il sistema restituisce nome e servizio associato dei progetti; – l'utente visualizza la lista dei progetti.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF10	RFO6 RFO6.1 RFO6.2 RFO6.3	• **Obiettivo**: l'utente visualizza le informazioni specifiche di un singolo progetto presente nel gestore personale. • **Procedura**: – il sistema restituisce le seguenti informazioni del progetto: ∗ nome; ∗ servizio associato; ∗ url; ∗ numero totale dei contatti iscritti al progetto. – l'utente visualizza il progetto.	V
TSOF11	RFO7 RFO7.1 RFO7.1.1 RFO7.1.2 RFO7.1.3 RFO7.1.4	• **Obiettivo**: l'utente iscrive un utente esistente ad uno specifico progetto specificando l'email del contatto da inserire, l'url del progetto e la priorità associata allo specifico progetto per quel contatto. Opzionalmente può fornire una lista di keyword associate al progetto. • **Procedura**: – l'utente inserisce un indirizzo email di un contatto esistente per la nuova iscrizione; – l'utente inserisce un indirizzo url di un progetto esistente e a cui l'utente non sia già collegato con un altra iscrizione; – l'utente inserisce una priorità per la nuova iscrizione; – l'utente invia le informazioni al sistema; – il sistema le accetta.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF12	RFO7 RFO7.2 RFO7.2.1 RFO7.2.2 RFO7.2.3 RFO7.2.4	• **Obiettivo**: l'utente modifica l'iscrizione di un contatto ad un progetto cambiando l'email del contatto, l'url del progetto, il livello di priorità e la lista di keyword. • **Procedura**: – l'utente può modificare le seguenti informazioni: ∗ l'indirizzo email; ∗ l'indirizzo url; ∗ il livello di priorità; ∗ la lista di keyword. – l'utente invia le informazioni al sistema; – il sistema le accetta.	V
TSOF13	RFO7 RFO7.3	• **Obiettivo**: l'utente cancella l'iscrizione ad un progetto di un contatto presente nel gestore personale. • **Procedura**: – l'utente invia al sistema l'iscrizione da cancellare; – il sistema lo cancella.	V
TSOF14	RFO8 RFO8.1 RFO8.1.1 RFO8.1.2 RFO8.1.3 RFO8.1.4	• **Obiettivo**: l'utente deve poter vedere la lista, anche personalizzata, di tutte le iscrizione esistenti nel gestore personale e visualizzando per ognuna il nome del contatto, il progetto associato e la priorità dell'iscrizione. • **Procedura**: – il sistema restituisce una lista di iscrizioni diversificate da: ∗ nome iscritto; ∗ cognome iscritto; ∗ nome progetto dell'iscritto; ∗ priorità iscrizione. – l'utente visualizza la lista delle iscrizioni.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF15	RFO9 RFO9.1 RFO9.2 RFO9.3 RFO9.4 RFO9.5 RFO9.6	• **Obiettivo**: l'utente visualizza le informazioni dettagliate di una singola iscrizione nel gestore personale. • **Procedura**: – il sistema restituisce le seguenti informazioni degli iscritti: ∗ nome; ∗ cognome; ∗ email; ∗ nome del progetto; ∗ priorità iscrizione; ∗ lista keyword. – l'utente visualizza le informazioni dell'iscritto.	V
TSOF16	RFO10	• **Obiettivo**: l'utente, utilizzando una keyword, ricerca informazioni specifiche riguardanti uno o più contatti e li ordina sotto forma di lista. • **Procedura**: – l'utente inserisce una keyword per filtrare i contatti; – il sistema restituisce la lista dei contatti filtrati.	V
TSOF17	RFO11	• **Obiettivo**: l'utente, utilizzando una keyword, ricerca informazioni specifiche riguardanti uno o più progetti e li riordina sotto forma di lista. • **Procedura**: – l'utente inserisce una keyword per filtrare i progetti; – il sistema restituisce la lista dei progetti filtrati.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF18	RFO12	• **Obiettivo**: l'utente, utilizzando una keyword, ricerca informazioni specifiche riguardanti uno o più iscrizioni e le riordina sotto forma di lista. • **Procedura**: – l'utente inserisce una keyword per filtrare le iscrizioni; – il sistema restituisce la lista delle iscrizioni filtrate.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF19	RFO13 RFO18 RFO19 RFO20 RFO33 RFO34 RFO35 RFO36 RFO37 RFO38 RFO39 RFO56 RFO57 RFO58 RFO59 RFO60	• **Obiettivo**: Il sistema deve mandare un messaggio di errore quando, durante l'inserimento di un nuovo contatto, i dati inseriti sono errati. • **Procedura**: – l'utente inserisce le informazioni richieste in modo errato, gli errori che può compiere sono: ∗ inserire un email già in uso; ∗ non viene inserito un nome; ∗ non viene inserito un cognome; ∗ non viene inserito un indirizzo email; ∗ inserire un nome di persona non corretto rispetto al dominio; ∗ inserire un cognome di persona non corretto rispetto al dominio; ∗ inserire un indirizzo email non corretto rispetto al dominio; ∗ inserire un numero di telefono non corretto rispetto al dominio; ∗ inserire un indirizzo email associato a Slack non corretto rispetto al dominio; ∗ inserire delle preferenze sulla ricezione delle notifiche non corrette rispetto al dominio; ∗ inserire delle date di indisponibilità non corrette rispetto al dominio; ∗ inserire un indirizzo email troppo lungo; ∗ inserire un nome troppo lungo; ∗ inserire un cognome troppo lungo; ∗ inserire un numero di telefono troppo lungo; ∗ inserire un indirizzo email collegato a Slack troppo lungo. – l'utente invia al sistema i dati inseriti; – il sistema restituisce il segnale di errore in questione.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF20	RFO14 RFO21 RFO22 RFO23 RFO40 RFO41 RFO42 RFO43 RFO44 RFO45 RFO46 RFO61 RFO62 RFO63 RFO64 RFO65	• **Obiettivo**: Il sistema deve mandare un messaggio di errore quando, durante la modifica di un contatto esistente, i dati inseriti sono errati. • **Procedura**: – l'utente inserisce le informazioni richieste in modo errato, gli errori che può compiere sono: ∗ inserire un email già in uso; ∗ inserire stringa vuota come nuovo nome; ∗ inserire stringa vuota come nuovo cognome; ∗ inserire stringa vuota come nuovo un indirizzo email; ∗ inserire un nome di persona non corretto rispetto al dominio; ∗ inserire un cognome di persona non corretto rispetto al dominio; ∗ inserire un indirizzo email non corretto rispetto al dominio; ∗ inserire un numero di telefono non corretto rispetto al dominio; ∗ inserire un indirizzo email associato a Slack non corretto rispetto al dominio; ∗ inserire delle preferenze sulla ricezione delle notifiche non corrette rispetto al dominio; ∗ vengono inserite delle date di indisponibilità non corrette rispetto al dominio; ∗ inserire un indirizzo email troppo lungo; ∗ inserire un nome troppo lungo; ∗ inserire un cognome troppo lungo; ∗ inserire un numero di telefono troppo lungo; ∗ inserire un indirizzo email collegato a Slack troppo lungo. – l'utente invia al sistema i dati modificati; – il sistema restituisce il segnale di errore in questione.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF21	RFO15 RFO24 RFO25 RFO47 RFO48 RFO66 RFO67 RFO77	• **Obiettivo**: Il sistema deve mandare un messaggio di errore quando, durante l'inserimento di un nuovo progetto, i dati inseriti sono errati. • **Procedura**: – l'utente inserisce le informazioni richieste in modo errato, gli errori che può compiere sono: ∗ inserire un url attribuito a un progetto aggiunto in precedenza; ∗ non viene inserito un indirizzo url; ∗ non viene inserito il nome di tale progetto; ∗ inserire un indirizzo url non corretto rispetto al dominio; ∗ inserire un nome non corretto rispetto al dominio; ∗ inserire un indirizzo url troppo lungo; ∗ inserire un nome di progetto troppo lungo; ∗ inserire un indirizzo url non valido. – l'utente invia al sistema i dati inseriti; – il sistema restituisce il segnale di errore in questione.	V
TSOF22	RFO26 RFO49 RFO68	• **Obiettivo**: Il sistema deve mandare un messaggio di errore quando, durante la modifica di un progetto esistente, i dati inseriti sono errati. • **Procedura**: – l'utente inserisce le informazioni richieste in modo errato, gli errori che può compiere sono: ∗ inserire stringa vuota come nuovo nome del progetto; ∗ inserire un nome non corretto rispetto al dominio; ∗ inserire un nome di progetto troppo lungo; – l'utente invia al sistema i dati modificati; – il sistema restituisce il segnale di errore in questione.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF23	RFO16 RFO27 RFO28 RFO29 RFO50 RFO51 RFO52 RFO69 RFO70 RFO73 RFO75	• **Obiettivo**: Il sistema deve mandare un messaggio di errore quando, durante l'inserimento di una nuova iscrizione, i dati inseriti sono errati. • **Procedura**: – l'utente inserisce le informazioni richieste in modo errato, gli errori che può compiere sono: ∗ inserire una coppia di url ed email che identificano un'iscrizione già esistente; ∗ non viene inserito un indirizzo email; ∗ non viene inserito un indirizzo url; ∗ non viene inserito un grado di priorità; ∗ inserire un indirizzo url non corretto rispetto al dominio; ∗ inserire un indirizzo email non corretto rispetto al dominio; ∗ inserire una priorità associata alla iscrizione non corretta rispetto al dominio; ∗ inserire un indirizzo email troppo lungo; ∗ inserire un indirizzo url troppo lungo; ∗ inserire un indirizzo email non associato con alcun contatto; ∗ inserire un indirizzo url non associato ad alcun progetto. – l'utente invia al sistema i dati inseriti; – il sistema restituisce il segnale di errore in questione.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF24	RFO17 RFO30 RFO31 RFO32 RFO53 RFO54 RFO55 RFO71 RFO72 RFO74 RFO76	• **Obiettivo**: Il sistema deve mandare un messaggio di errore quando, durante la modifica di una iscrizione esistente, i dati inseriti sono errati. • **Procedura**: – l'utente modifica un'iscrizione esistente; – l'utente inserisce le informazioni richieste in modo errato, gli errori che può compiere sono: ∗ inserire una coppia di url ed email che identificano un'iscrizione già esistente; ∗ inserire una stringa vuota come nuovo indirizzo email del contatto associato all'iscrizione; ∗ inserire una stringa vuota come nuovo indirizzo url del progetto associato all'iscrizione; ∗ inserire una stringa vuota come nuovo grado di priorità associato all'iscrizione; ∗ inserire un indirizzo url non corretto rispetto al dominio; ∗ inserire un indirizzo email non corretto rispetto al dominio; ∗ inserire una priorità associata alla iscrizione non corretta rispetto al dominio; ∗ inserire un indirizzo email troppo lungo; ∗ inserire un indirizzo url troppo lungo; ∗ inserire un indirizzo email non associato con alcun contatto; ∗ inserire indirizzo url non associato ad alcun progetto. – l'utente invia al sistema i dati modificati; – il sistema restituisce il segnale di errore in questione.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF25	RFO78 RFO79 RFO88	• **Obiettivo**: GitLab invia una notifica a GitLabProducer segnalando che è avvenuta una operazione di push. • **Procedura**: – l'utente esegue una operazione di push sul progetto Gitlab; – GitLabProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di push.	V
TSOF26	RFO78 RFO80 RFO88	• **Obiettivo**: GitLab invia una notifica a GitLabProducer segnalando che è stata creata una nuova issue. • **Procedura**: – l'utente esegue una operazione di creazione di una nuova issue sul progetto Gitlab; – GitLabProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di creazione issue.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF27	RFO78 RFO81 RFO88	• **Obiettivo**: GitLab invia una notifica a GitLabProducer segnalando che è stata modificata una issue. • **Procedura**: – l'utente esegue una operazione di modifica di una issue sul progetto Gitlab; – GitLabProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di modifica issue.	V
TSOF28	RFO82 RFO83 RFO89	• **Obiettivo**: Redmine invia una notifica a RedmineProducer segnalando che è stata creata una nuova issue. • **Procedura**: – l'utente esegue una operazione di creazione di una nuova issue sul progetto Redmine; – RedmineProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di creazione issue.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF29	RFO82 RFO84 RFO89	• **Obiettivo**: Redmine invia una notifica a RedmineProducer segnalando che è stata modificata una issue. • **Procedura**: – l'utente esegue una operazione di modifica di una issue sul progetto Redmine; – RedmineProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di modifica issue.	V
TSOF30	RFO82 RFO85 RFO89	• **Obiettivo**: Redmine invia una notifica a RedmineProducer segnalando che è stata inserita una priorità per una issue. • **Procedura**: – l'utente esegue una operazione di inserimento di priorità per una issue sul progetto Redmine; – RedmineProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di inserimento priorità issue.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF31	RFO86 RFO89	• **Obiettivo**: Redmine invia una notifica a RedmineProducer segnalando che è stata modificata la priorità per una issue. • **Procedura**: – l'utente esegue una operazione di modifica di priorità per una issue sul progetto Redmine; – RedmineProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di modifica priorità di issue.	V
TSOF32	RFO87 RFO90	• **Obiettivo**: SonarQube invia una notifica al componente SonarQubeProducer elaborandola e inviando le informazioni opportune al broker. • **Procedura**: – l'utente esegue una operazione sul progetto SonarQube; – SonarQubeProducer invia la notifica al broker; – il broker invia la notifica al gestore; – il gestore elabora la notifica e la invia al broker; – broker la invia al consumer; – l'utente verifica l'arrivo della notifica di evento.	NIP
TSOF33	RFO91	• **Obiettivo**: il componente TelegramConsumer invia un pacchetto contenente le informazioni correlate ad un evento ad uno o più contatti Telegram. • **Procedura**: – l'utente esegue un azione sul progetto da causare una notifica; – l'utente verifica l'arrivo della corrispondente notifica su Telegram.	V

– *continua a pagina successiva*

Tabella7– *continuazione da pagina precedente*

Codice	Requisito	Descrizione	Stato
TSOF34	RFO92	• **Obiettivo**: il componente EmailConsumer invia un pacchetto contenente le informazioni correlate ad un evento ad uno o più indirizzi email. • **Procedura**: – l'utente esegue un azione sul progetto da causare una notifica; – l'utente verifica l'arrivo della corrispondente notifica sull'indirizzo email.	V
TSOF35	RFO93	• **Obiettivo**: il componente SlackConsumer invia un pacchetto contenente le informazioni correlate ad un evento ad uno o più contatti Slack. • **Procedura**: – l'utente esegue un azione sul progetto da causare una notifica; – l'utente verifica l'arrivo della corrispondente notifica su Slack.	V

4.3 Test di integrazione

Verranno qui presentati alcuni dei test d'integrazione, previsti per le varie componenti al fine di assicurarne il corretto funzionamento quando messe in relazione, altri test verranno aggiungi prossimamente secondo il modello a V.

Tabella 8: Tabella dei test di integrazione

Codice	Descrizione	Stato
TI01	Viene verificato che il sistema gestisca correttamente l'integrazione tra *GitLabProducer* e il broker *Kafka*.	V
TI02	Viene verificato che il *GitLabProducer* sia in grado di leggere correttamente gli eventi di *Gitlab* relativi alle issue.	V
TI03	Viene verificato che il *GitLabProducer* sia in grado di leggere correttamente gli eventi di *Gitlab* relativi alle push.	V
TI04	Viene verificato che il sistema gestisca correttamente l'integrazione tra *RedmineProducer* e il broker *Kafka*.	V
TI05	Viene verificato che il *RedmineProducer* sia in grado di leggere correttamente gli eventi di *Redmine* relativi alle creazione di una segnalazione.	V
TI06	Viene verificato che il sistema gestisca correttamente l'integrazione tra *SonarqubeProducer* e il broker *Kafka*.	NIP
TI07	Viene verificato che il *SonarqubeProducer* sia in grado di leggere correttamente l'evento relativo all'analisi statica con *Sonarqube*.	NIP
TI08	Viene verificato che il sistema gestisca correttamente l'integrazione tra *TelegramConsumer* e il broker *Kafka*.	V
TI09	Viene verificato che il sistema gestisca correttamente l'integrazione tra il *TelegramConsumer* e il servizio di comunicazione esterna *Telegram*.	V
TI10	Viene verificato che il sistema gestisca correttamente l'integrazione tra *EmailConsumer* e il broker *Kafka*.	V
TI11	Viene verificato che il sistema gestisca correttamente l'integrazione tra il *EmailConsumer* e il servizio di comunicazione esterna SMTP.	V
TI12	Viene verificato che il sistema accetti solo nomi di lunghezza minore o uguale a 50 caratteri.	V
TI13	Viene verificato che il sistema accetti solo cognomi di lunghezza minore o uguale a 50 caratteri.	V
TI14	Viene verificato che l'email inserita durante la creazione di un nuovo contatto rispetti il formato delle email.	V
TI15	Viene verificato che il sistema accetti email passata per la modifica del contatto, sia inserita nel formato valido.	V
TI16	Viene verificato che il sistema accetti solo nomi di lunghezza minore o uguale a 50 caratteri prima di modificare il contatto.	V
TI17	Viene verificato che il sistema accetti solo cognomi di lunghezza minore o uguale a 50 caratteri prima di modificare il contatto.	V

– *continua a pagina successiva*

Tabella8– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TI18	Viene verificato che l'email passata per la cancellazione di un contatto sia espressa nel formato valido.	V
TI19	Viene verificato che il sistema elimini tutte le iscrizioni ad eventi che il contatto eliminato ha effettuato.	V
TI20	Viene verificato che il sistema elimini tutte i riferimenti di contatto associati al contatto eliminato.	V
TI21	Viene verificato che il sistema, per l'inserimento di un progetto, accetti solo nomi di progetto di lunghezza minore o uguale a 50 caratteri.	V
TI22	Viene verificato che il corpo della richiesta REST per l'inserimento del progetto contenga i dati obbligatori ovvero nome e url di progetto.	V
TI23	Viene verificato che il sistema, per la ricerca di un progetto specifico, accetti solo nomi di progetto di lunghezza minore o uguale a 50 caratteri.	V
TI24	Viene verificato che il sistema, per la modifica di un progetto, accetti solo nomi di progetto di lunghezza minore o uguale a 50 caratteri.	V
TI25	Viene verificato che il sistema, per la cancellazione di un progetto, accetti solo nomi di progetto di lunghezza minore o uguale a 50 caratteri.	V
TI26	Viene verificato che il sistema elimini tutte le iscrizioni ad eventi che coinvolgono il progetto eliminato.	V
TI27	Viene verificato che il corpo della richiesta inviata all'endpoint REST di inserimento di un nuovo contatto contenga almeno i dati obbligatori ovvero email, nome e cognome.	V
TI28	Viene verificato che il corpo della richiesta inviata all'endpoint REST di modifica del contatto contenga almeno campi email, nome e cognome.	V
TI29	Viene verificato che il sistema, per la ricerca di un contatto specifico, accetti solo nomi o cognomi di lunghezza minore o uguale a 50 caratteri.	V

4.4 Test di unità

Hanno il fine di verificare il corretto funzionamento della parte più piccola autonoma del prodotto. I test in questione verranno sviluppati in seguito, secondo il modello a V.

Tabella 9: Tabella dei test di integrazione

Codice	Descrizione	Stato
TU001	Il metodo create() di KafkaConsumerFactory crea un KafkaConsumer con successo	V
TU002	Il metodo create() di KafkaConsumerFactory lancia una Exception quando, dopo 10 tentativi, continua a ricevere l'errore NoBrokersAvailable creando il KafkaConsumer.	V
TU003	Il metodo create di KafkaConsumerFactory chiude il programma se riceve un KeyboardInterrupt.	V
TU004	La classe base Consumer viene costruita assegnando il parametro consumer di tipo KafkaConsumer all'attributo. _consumer	V
TU005	Il metodo start() della classe base Consumer chiama il suo metodo consume() per ogni messaggio nel suo attributo. _consumer	V
TU006	Il metodo get_topic_list() della classe base Consumer ritorna il risultato del metodo topics() del suo parametro. _consumer	V
TU007	Il metodo get_subscription() della classe base Consumer ritorna il risultato del metodo subscription() del suo parametro. _consumer	V
TU008	Il metodo close() della classe base Consumer chiama il metodo close() del suo parametro. _consumer	V
TU009	La classe MailSender viene costruita assegnando il parametro server di tipo SMTP all' attributo _server, il parametro email di tipo stringa all' attributo _email, il parametro password di tipo stringa all' attributo _password, e avviando il parametro _server con i suoi metodi ehlo(), starttls() e login() con i parametri _email e _password.	V
TU010	Il metodo send() chiama i metodi sendmail() con parametri l'attributo _email e e i parametri recipient di tipo stringa e message di tipo stringa, e close() del suo attributo _server.	V
TU011	Il metodo getEmail() della classe MailSender ritorna il suo attributo _email.	V
TU012	La classe EmailConsumer viene costruita chiamando il costruttore della classe base con il parametro consumer di tipo KafkaConsumer, e assegnando il parametro sender di tipo MailSender all'attributo _sender.	V
TU013	Il metodo consume() della classe EmailConsumer chiama il metodo send() dell'attributo _sender per ogni recipient ritornato dal metodo extract_recipient_list().	V
TU014	Il metodo extract_recipient_list() della classe EmailConsumer ritorna il risultato della funzione pop() con parametro di tipo stringa "recipients" sul parametro configs di tipo map.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU015	Il metodo _decorate() della classe EmailConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a una nuova issue di Gitlab.	V
TU016	Il metodo _decorate() della classe EmailConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a un nuovo push di Gitlab.	V
TU017	Il metodo _decorate() della classe EmailConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a una nuova issue di Redmine.	V
TU018	La funzione start_email_consumer() crea un MailSender, chiama il metodo create() di KafkaConsumerFactory, chiama il costruttore di EmailConsumer con i due oggetti creati come parametri, esegue il suo metodo start() e in seguito esegue il suo metodo close() con successo.	V
TU019	La funzione start_email_consumer() crea un MailSender, chiama il metodo create() di KafkaConsumerFactory che fallisce lanciando una Exception, e il programma si chiude stampando il messaggio di errore.	V
TU020	La funzione start_email_consumer() crea un MailSender, chiama il metodo create() di KafkaConsumerFactory, chiama il costruttore di EmailConsumer con i due oggetti creati come parametri, esegue il suo metodo start(), ma durante la sua esecuzione viene lanciato l'eccezione KeyboardInterrupt, e il programma si chiude.	V
TU021	La classe SlackBot viene creata assegnando il parametro client di tipo SlackClient all'attributo _client, il parametro mongo_db di tipo Mongo all'attributo _mongo, e l'attributo _isListening di tipo booleano a falso.	V
TU022	Il metodo send_message() della classe SlackBot chiama il metodo api_call() del suo attributo _client con i parametri "chat.postMessage", recipient di tipo stringa e message di tipo stringa	V
TU023	Il metodo start() della classe SlackBot chiama il metodo rtm_connect() dell'attributo _client, prende il risultato del metodo api_call() dell'attributo _client e per ogni elemento user di tipo map ottenuto, se contiene la chiave "email", viene chiamato il metodo update_slack_id() dell'attributo _mongo, viene impostata isListening a falso e chiamato il metodo listen() con successo.	V
TU024	Il metodo listen() della classe SlackBot, fintanto che l'attributo _isListening é vero, per ogni elemento event ritornato dal metodo rtm_read() dell'attributo _client, chiama il metodo _parse_event(), e attende 1 secondo.	V
TU025	Il metodo close() della classe SlackBot imposta l'attributo _isListening a falso.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU026	Il metodo _parse_event() della classe SlackBot chiama il metodo api_call() dell'attributo _client e per ogni elemento ritornato che possiede i giusti attributi, chiama il metodo update_slack_id() dell'attributo _mongo con successo.	V
TU027	La classe SlackConsumer viene creata chiamando il costruttore della classe base con il parametro consumer di tipo KafkaConsumer, e assegna il parametro bot di tipo SlackBot all'attributo _bot.	V
TU028	Il metodo consume() della classe SlackConsumer chiama il metodo send_message() dell'attributo _bot per ogni recipient ritornato dal metodo extract_recipient_list().	V
TU029	Il metodo _decorate() della classe SlackConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a una nuova issue di Gitlab.	V
TU030	Il metodo _decorate() della classe SlackConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a un nuovo push di Gitlab.	V
TU031	Il metodo _decorate() della classe SlackConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a una nuova issue di Redmine.	V
TU032	Il metodo extract_recipient_list() della classe SlackConsumer ritorna il risultato della funzione pop() con parametro di tipo stringa "recipients" sul parametro configs di tipo map.	V
TU033	La funzione start_slack_consumer() crea un Mongo, uno SlackBot con il Mongo creato, chiama il metodo create() di KafkaConsumerFactory, chiama il costruttore di SlackConsumer con i due oggetti creati come parametri, chiama il metodo start() della classe Thread ed esegue il suo metodo start(). In seguito esegue il metodo close() dello SlackConsumer e SlackBot con successo.	V
TU034	La classe TelegramBot viene creata assegnando il parametro bot di tipo TeleBot all'attributo _bot, il parametro mongo_db di tipo Mongo all'attributo _mongo, e l'attributo.	V
TU035	Il metodo set_bot() della classe TelegramBot chiama il proprio metodo close() e assegna all'attributo _bot il valore del parametro bot.	V
TU036	Il metodo get_bot() della classe TelegramBot ritorna l'attributo _bot.	V
TU037	Il metodo close() della classe TelegramBot chiama il metodo stop_polling() e stop_bot() dell'attributo _bot.	V
TU038	Il metodo start() della classe TelegramBot chiama il metodo polling dell'attributo _bot.	V
TU039	Il metodo sendMessage() della classe TelegramBot chiama il metodo send_message() dell'attributo _bot con i parametri recipient di tipi intero e message di tipo stringa	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU040	La classe TelegramConsumer viene creata chiamando il costruttore della classe base con il parametro consumer di tipo KafkaConsumer, e assegna il parametro bot di tipo TelegramBot all'attributo _bot.	V
TU041	Il metodo _decorate() della classe TelegramConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a una nuova issue di Gitlab.	V
TU042	Il metodo _decorate() della classe TelegramConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a un nuovo push di Gitlab.	V
TU043	Il metodo _decorate() della classe TelegramConsumer ritorna un testo per rappresentare le informazioni ricevute come parametro data, relative a una nuova issue di Redmine.	V
TU044	Il metodo consume() della classe TelegramConsumer chiama il metodo send_message() dell'attributo _bot per ogni recipient ritornato dal metodo extract_recipient_list().	V
TU045	Il metodo extract_recipient_list() della classe TelegramConsumer ritorna il risultato della funzione pop() con parametro di tipo stringa "recipients" sul parametro configs di tipo map.	V
TU046	La funzione start_telegram_consumer() crea un Mongo, un TelegramBot con il Mongo creato, chiama il metodo create() di KafkaConsumerFactory, chiama il costruttore di SlackConsumer con i due oggetti creati come parametri, chiama il metodo start() della classe Thread ed esegue il suo metodo start(). In seguito esegue il metodo close() dello SlackConsumer e SlackBot con successo.	V
TU047	La funzione start_telegram_consumer() crea un Mongo, un TelegramBot con il Mongo creato, chiama il metodo create() di KafkaConsumerFactory che fallisce lanciando una Exception, e il programma si chiude stampando il messaggio di errore.	V
TU048	La funzione start_telegram_consumer() crea un Mongo, un TelegramBot con il Mongo creato, chiama il metodo create() di KafkaConsumerFactory, chiama il costruttore di SlackConsumer con i due oggetti creati come parametri, chiama il metodo start() della classe Thread ed esegue il suo metodo start(), ma durante la loro esecuzione viene lanciato l'eccezione KeyboardInterrupt, e il programma si chiude.	V
TU049	Il metodo _yes_no_to_bool() della classe Terminal ritorna il valore di vero o falso del parametro answer che contiene una chiave di _yes_no_dict	V
TU050	Il metodo _yes_no_to_bool() della classe Terminal ritorna il valore di falso con il parametro answer che non contiene una chiave di _yes_no_dict	V
TU051	La classe Terminal viene creata correttamente attribuendo il valore del parametro mongo_db di tipo Mongo all'attributo _mongo	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU052	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "exit", stampa End Terminal e finisce.	V
TU053	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "insert contact", chiama il suo metodo insert_contact() , e in seguito si chiude ricevuto l'input "exit".	V
TU054	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "update contact", chiama il suo metodo update_contact() , e in seguito si chiude ricevuto l'input "exit".	V
TU055	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "delete contact", chiama il suo metodo delete_contact() , e in seguito si chiude ricevuto l'input "exit".	V
TU056	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "find contact", chiama il suo metodo find_contact() , e in seguito si chiude ricevuto l'input "exit".	V
TU057	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "list contact", chiama il suo metodo list_contact(), e in seguito si chiude ricevuto l'input "exit".	V
TU058	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "insert project", chiama il suo metodo insert_project(), e in seguito si chiude ricevuto l'input "exit".	V
TU059	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "update project", chiama il suo metodo update_project(), e in seguito si chiude ricevuto l'input "exit".	V
TU060	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "delete project", chiama il suo metodo delete_project(), e in seguito si chiude ricevuto l'input "exit".	V
TU061	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "find project", chiama il suo metodo find_project(), e in seguito si chiude ricevuto l'input "exit".	V
TU062	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "list project", chiama il suo metodo list_project(), e in seguito si chiude ricevuto l'input "exit".	V
TU063	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "insert subscription", chiama il suo metodo insert_subscription() , e in seguito si chiude ricevuto l'input "exit".	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU064	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "update subscription", chiama il suo metodo update_subscription(), e in seguito si chiude ricevuto l'input "exit".	V
TU065	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "delete subscription", chiama il suo metodo delete_subscription(), e in seguito si chiude ricevuto l'input "exit".	V
TU066	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "find subscription", chiama il suo metodo find_subscription(), e in seguito si chiude ricevuto l'input "exit".	V
TU067	il metodo show() della classe Terminal stampa "Start terminal" e "Enter command: " , e ricevuto in input la stringa "list subscription", chiama il suo metodo list_subscription(), e in seguito si chiude ricevuto l'input "exit".	V
TU068	Il metodo print_query_no_result() della classe Terminal stampa "The query didn't find any result".	V
TU069	Il metodo print_record_inserted() della classe Terminal stampa "Record inserted, ID:" seguito dalla stringa del parametro record_id.	V
TU070	Il metodo print_record_updated() della classe Terminal stampa "Record updated".	V
TU071	Il metodo print_record_deleted() della classe Terminal stampa "Record deleted".	V
TU072	Il metodo print_record_not_found() della classe Terminal stampa "Record not found".	V
TU073	Il metodo print_error() della classe Terminal stampa "Error."seguito dalla stringa del parametro 'e' di tipo Exception.	V
TU074	Il metodo insert_contact() della classe Terminal chiede un input di 10 dati, e in seguito chiama il metodo insert_contact() dell'attributo _mongo con parametri i dati inseriti prima, e chiama il metodo print_record_inserted() con il dato ritornato.	V
TU075	Il metodo insert_contact() della classe Terminal chiede un input di 10 dati, e in seguito chiama il metodo insert_contact() dell'attributo _mongo con parametri i dati inseriti prima, che lancia una Exception, e chiama il metodo print_error().	V
TU076	Il metodo update_contact() della classe Terminal chiede un input di 10 dati, e in seguito chiama il metodo update_contact() dell'attributo _mongo con parametri i dati inseriti prima, e chiama il metodo print_record_inserted() con il dato ritornato.	V
TU077	Il metodo update_contact() della classe Terminal chiede un input di 10 dati, e in seguito chiama il metodo update_contact() dell'attributo _mongo con parametri i dati inseriti prima, che lancia una Exception, e chiama il metodo print_error().	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU078	Il metodo delete_contact() della classe Terminal chiede un input e chiama il metodo find_contact_by_email() dell'attributo _mongo. Trovato un match chiama il proprio metodo ask_for_confirmation(), e una volta confermato chiama il metodo delete_contact() dell'attributo_mongo e il proprio metodo print_record_deleted().	V
TU079	Il metodo delete_contact() della classe Terminal chiede un input e chiama il metodo find_contact_by_email() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_record_not_found().	V
TU080	Il metodo find_contacts() della classe Terminal chiede un input e chiama il metodo find_contacts() dell'attributo _mongo. Trovato almeno un match stampa i risultati.	V
TU081	Il metodo find_contacts() della classe Terminal chiede un input e chiama il metodo find_contacts() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_query_no_result().	V
TU082	Il metodo list_contacts() della classe Terminal chiama il metodo list_contacts() dell'attributo _mongo. Trovato almeno un match stampa i risultati.	V
TU083	Il metodo list_contacts() della classe Terminal chiama il metodo list_contacts() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_query_no_result().	V
TU084	Il metodo insert_project() della classe Terminal chiede un input di 2 dati, e in seguito chiama il metodo insert_project() dell'attributo _mongo con parametri i dati inseriti prima, e chiama il metodo print_record_inserted() con il dato ritornato.	V
TU085	Il metodo insert_project() della classe Terminal chiede un input di 2 dati, e in seguito chiama il metodo insert_project() dell'attributo _mongo con parametri i dati inseriti prima, che lancia una Exception, e chiama il metodo print_error().	V
TU086	Il metodo update_project() della classe Terminal chiede un input di 2 dati, e in seguito chiama il metodo update_project() dell'attributo _mongo con parametri i dati inseriti prima, e chiama il metodo print_record_inserted() con il dato ritornato.	V
TU087	Il metodo update_project() della classe Terminal chiede un input di 2 dati, e in seguito chiama il metodo update_project() dell'attributo _mongo con parametri i dati inseriti prima, che lancia una Exception, e chiama il metodo print_error().	V
TU088	Il metodo delete_project() della classe Terminal chiede un input e chiama il metodo find_project_by_url() dell'attributo _mongo. Trovato un match chiama il proprio metodo ask_for_confirmation(), e una volta confermato chiama il metodo delete_project() dell'attributo_mongo e il proprio metodo print_record_deleted().	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU089	Il metodo delete_project() della classe Terminal chiede un input e chiama il metodo find_project_by_url() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_record_not_found().	V
TU090	Il metodo find_projects() della classe Terminal chiede un input e chiama il metodo find_projects() dell'attributo _mongo. Trovato almeno un match stampa i risultati.	V
TU091	Il metodo find_projects() della classe Terminal chiede un input e chiama il metodo find_projects() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_query_no_result().	V
TU092	Il metodo list_projects() della classe Terminal chiama il metodo list_projects() dell'attributo _mongo. Trovato almeno un match stampa i risultati.	V
TU093	Il metodo list_projects() della classe Terminal chiama il metodo list_projects() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_query_no_result().	V
TU094	Il metodo insert_subscription() della classe Terminal chiede un input di 4 dati, e in seguito chiama il metodo insert_subscription() dell'attributo _mongo con parametri i dati inseriti prima, e chiama il metodo print_record_inserted() con il dato ritornato.	V
TU095	Il metodo insert_subscription() della classe Terminal chiede un input di 10 dati, e in seguito chiama il metodo insert_subscription() dell'attributo _mongo con parametri i dati inseriti prima, che lancia una Exception, e chiama il metodo print_error().	V
TU096	Il metodo update_subscription() della classe Terminal chiede un input di 2 dati, e in seguito chiama il metodo update_subscription() dell'attributo _mongo con parametri i dati inseriti prima, e chiama il metodo print_record_inserted() con il dato ritornato.	V
TU097	Il metodo update_subscription() della classe Terminal chiede un input di 2 dati, e in seguito chiama il metodo update_subscription() dell'attributo _mongo con parametri i dati inseriti prima, che lancia una Exception, e chiama il metodo print_error().	V
TU098	Il metodo delete_subscription() della classe Terminal chiede 2 input e chiama il metodo find_subscription_by_email_url() dell'attributo _mongo. Trovato un match chiama il proprio metodo ask_for_confirmation(), e una volta confermato chiama il metodo delete_subscription() dell'attributo_mongo e il proprio metodo print_record_deleted().	V
TU099	Il metodo delete_subscription() della classe Terminal chiede 2 input e chiama il metodo find_subscription_by_email_url() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_record_not_found().	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU100	Il metodo find_subscription() della classe Terminal chiede un input e chiama il metodo find_subscription() dell'attributo _mongo. Trovato almeno un match stampa i risultati.	V
TU101	Il metodo find_subscription() della classe Terminal chiede un input e chiama il metodo find_subscription() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_query_no_result().	V
TU102	Il metodo list_subscription() della classe Terminal chiama il metodo list_subscription() dell'attributo _mongo. Trovato almeno un match stampa i risultati.	V
TU103	Il metodo list_subscription() della classe Terminal chiama il metodo list_subscription() dell'attributo _mongo. Non trovato un match chiama il proprio metodo print_query_no_result().	V
TU104	Il metodo ask_for_confirmation() della classe Terminal prende un input, e lo passa al metodo print_query_no_result(), per ritornare poi il valore che riceve.	V
TU105	Il metodo start_terminal() crea un oggetto di tipo Mongo con i suoi parametri, e in seguito chiama il costruttore di Terminal con l'oggetto di tipo Mongo creato, e chiama il suo metodo view().	V
TU106	la classe PersonalManager viene creata assegnando il parametro mongo di tipo Mongo all'attributo _mongo, e il parametro producer di tipo Producer all'attributo _producer.	V
TU107	Il metodo process() della classe PersonalManager chiama il metodo get_interested_recipients() con il parametro data, poi ritornando un risultato non vuoto, chiama il metodo _add_preferences(), e produce() dell'attributo _producer.	V
TU108	Il metodo stop() della classe PersonalManager chiama il metodo close() dell'attributo _producer.	V
TU109	Il metodo _add_preferences() della classe PersonalManager aggiunge la chiave "recipients" dal primo parametro data di tipo map al secondo parametro recipients di tipo map.	V
TU110	La classe DispatcherProducer viene creata correttamente, chiamando il costruttore della classe base Producer.	V
TU111	Il metodo produce() della classe DispatcherProducer chiama il metodo extract_topic_list(), e per ogni valore ritornato dalla chiamata del metodo keys() del risultato, esegue il suo metodo send() e il metodo pop() del parametro data di tipo map	V
TU112	Il metodo extract_topic_list() della classe DispatcherProducer esegue il metodo pop() del parametro data di tipo map	V
TU113	La classe DispatcherConsumer viene creata assegnando il parametro processor di tipo Processor all'attributo _processor, e chiama il costruttore della classe base Consumer.	V
TU114	Il metodo consume() della classe DispatcherConsumer chiama il metodo process() dell'attributo _processor con suo parametro data.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU115	Il metodo close() della classe DispatcherConsumer chiama il metodo close() della classe base e il metodo stop() dell'attributo _processor.	V
TU116	La funzione create_dispatcher_producer() chiama il metodo create() di KafkaProducerFactory e ritorna un DispatcherProducer costruito con il valore ritornato da prima con successo.	V
TU117	La funzione create_dispatcher_producer() chiama il metodo create() di KafkaProducerFactory, che lancia un Exception, stampa il messaggio di errore ed chiude il programma.	V
TU118	La funzione create_personal_manager() costruisce un oggetto di tipo Mongo con i parametri ottenuti dal parametro configs di tipo map, e ritorna un PersonalManager costruito con l'oggetto creato.	V
TU119	La funzione create_dispatcher_consumer() chiama il metodo create() di KafkaProducerFactory e ritorna un DispatcherConsumer costruito con il valore ritornato da prima con successo.	V
TU120	La funzione create_dispatcher_consumer() chiama il metodo create() di KafkaProducerFactory, che lancia un Exception, stampa il messaggio di errore ed chiude il programma.	V
TU121	La funzione start_manager() chiama la funzione create_dispatcher_consumer() e invoca i metodi start() e close() dell'oggetto ritornato con successo.	V
TU122	La funzione start_manager() chiama la funzione create_dispatcher_consumer() e invoca i metodi start(), e close() dell'oggetto ritornato, ma ricevendo un KeyboardInterrupt, chiude il programma.	V
TU123	La classe Mongo viene creata chiamando la funziona connect() con i suoi parametri.	V
TU124	Il metodo first_element_if_exists() della classe Mongo ritorna il primo risultato del parametro query_result	V
TU125	Il metodo first_element_if_exists() della classe Mongo ritorna None con il parametro query_result vuoto	V
TU126	Il metodo insert_contact() della classe Mongo costruisce un Telegram, uno Slack, un Preferences e un Holidays, e costruisce un Contact con gli oggetti creati in precedenza. In seguito chiama il metodo save() dell'oggetto di tipo Contact costruito, e lo ritorna.	V
TU127	Il metodo insert_contact() della classe Mongo costruisce un Telegram, uno Slack, un Preferences e un Holidays ma riceve delle date per l'oggetto Holidays incoerenti tra loro (una vacanza che comincia dopo la sua conclusione), e lancia un'eccezione.	V
TU128	Il metodo update_contact() della classe Mongo chiama il metodo find_contact_by_email() della classe con un suo parametro, e ricevuto un oggetto non vuoto, vi assegna come attributi gli altri parametri ricevuti.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU129	Il metodo update_contact() della classe Mongo chiama il metodo find_contact_by_email() della classe con un suo parametro, e ricevuto un oggetto non vuoto, vi assegna come attributi gli altri parametri ricevuti, ma riceve delle date per l'oggetto Holidays incoerenti tra loro (una vacanza che comincia dopo la sua conclusione), e lancia un'eccezione.	V
TU130	Il metodo find_contact_by_email() della classe Mongo chiama il metodo objects() della classe Contact, e ritorna il risultato del metodo first_element_if_exists() passando come parametro l'oggetto ottenuto dalla precedente chiamata	V
TU131	Il metodo delete_contact() della classe Mongo chiama il metodo delete() del parametro contact di tipo Contact ricevuto.	V
TU132	Il metodo find_contacts() chiama il metodo compile delle espressioni regolari per formattare il parametro keyword ricevuto. In seguito chiama il metodo objects() dell'oggetto Contact, e ritorna la lista dei risultati filtrati.	V
TU133	Il metodo list_contacts() della classe Mongo ritorna il risultato del metodo objects() della classe Contacts.	V
TU134	Il metodo insert_project() della classe Mongo ritorna un oggetto di tipo Project, costruendolo con i suoi parametri, e chiamando il metodo save().	V
TU135	Il metodo update_project della classe Mongo chiama il metodo find_project_by_url(), e con un risultato non vuoto, vi assegna come attributi i parametri ricevuti, ritornando l'oggetto modificato.	V
TU136	Il metodo find_project_by_url() della classe Mongo chiama il metodo objects della classe Project, e ritorna il risultato del metodo first_element_if_exists() della classe, con parametro l'oggetto precedente.	V
TU137	Il metodo delete_project() della classe Mongo chiama il metodo delete() del parametro project di tipo Project ricevuto.	V
TU138	Il metodo find_projects() della classe Mongo chiama il metodo compile delle espressioni regolari per formattare il parametro keyword ricevuto. In seguito chiama il metodo objects() della classe Project, e ritorna la lista dei risultati filtrati.	V
TU139	Il metodo list_projects() della classe Mongo ritorna il risultato del metodo objects() della classe Projects.	V
TU140	Il metodo insert_subscription() della classe Mongo chiama i metodi objects() delle classi Project e Contact, chiamando il metodo di classe first_element_if_exists() su entrambi, e entrambi i risultati non nulli, crea un Subscription usandoli come parametri, e chiama il metodo save() del nuovo oggetto, ritornandolo.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU141	Il metodo insert_subscription() della classe Mongo chiama i metodi objects() delle classi Project e Contact, chiamando il metodo di classe first_element_if_exists() su entrambi, con uno dei risultati ritornato nullo, lancia una Exception con stringa "Contatto o progetto inesistenti".	V
TU142	Il metodo update_subscription della classe Mongo chiama il metodo find_subscription_by_email_url(), e con un risultato non vuoto, vi assegna come attributi i parametri ricevuti, ritornando l'oggetto modificato.	V
TU143	Il metodo delete_subscription() della classe Mongo chiama il metodo delete() del parametro subscription di tipo Subscription ricevuto.	V
TU144	Il metodo find_subscription_by_email_url() della classe Mongo chiama i metodi objects della classe Project e Contacts, chiamando il metodo di classe first_element_if_exists() su entrambi, e li usa come parametri nel chiamare il metodo objects() della classe Subscription, ritorna il risultato dei metodo first_element_if_exists() della classe, con parametro l'oggetto precedente.	V
TU145	Il metodo find_subscriptions() della classe Mongo chiama il metodo compile delle espressioni regolari per formattare il parametro keyword ricevuto. In seguito chiama il metodo objects() della classe Subscription, e ritorna la lista dei risultati filtrati.	V
TU146	Il metodo list_subscriptions() della classe Mongo ritorna il risultato del metodo objects() della classe Subscription.	V
TU147	Il metodo update_telegram_id() della classe Mongo chiama il metodo objects() della classe Contact, passando come parametro il risultato al metodo di classe first_element_if_exists(), che ritorna un risultato non vuoto. In seguito assegna il parametro telegram_id all'oggetto tornato, e chama il suo metodo save().	V
TU148	Il metodo update_telegram_id() della classe Mongo chiama il metodo objects() della classe Contact, passando come parametro il risultato al metodo di classe first_element_if_exists(), che ritorna un risultato vuoto. Il metodo lancia una Exception con stringa.	V
TU149	Il metodo update_slack_id() della classe Mongo chiama il metodo objects() della classe Contact, passando come parametro il risultato al metodo di classe first_element_if_exists(), che ritorna un risultato non vuoto. In seguito assegna il parametro slack_id all'oggetto tornato, e chama il suo metodo save().	V
TU150	Il metodo update_slack_id() della classe Mongo chiama il metodo objects() della classe Contact, passando come parametro il risultato al metodo di classe first_element_if_exists(), che ritorna un risultato vuoto. Il metodo lancia una Exception con stringa.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU151	Il metodo get_interested_recipients() della classe Mongo chiama il metodo find_project_by_url() e usa il risultato come parametro per il metodo objects() della classe Subscription, che ritorna un oggetto non vuoto. Il metodo racchiude i dati dell'oggetto in una map e per ogni oggetto del subscription_query_result() chiama il metodo di classe add_recipients_to_dict().	V
TU152	Il metodo get_interested_recipients() della classe Mongo chiama il metodo find_project_by_url() e usa il risultato come parametro per il metodo objects() della classe Subscription, che ritorna un oggetto vuoto. Il metodo ritorna None.	V
TU153	Il metodo add_recipients_to_dict() chiama il metodo objects() della classe Contact, passando come parametro il risultato al metodo di classe first_element_if_exists() e aggiunge i corrispettivi dati dell'oggetto ritornato al parametro recipients di tipo map.	V
TU154	Il metodo create() di KafkaProducerFactory crea un KafkaProducer con successo	V.
TU155	Il metodo create() di KafkaProducerFactory lancia una Exception quando, dopo 10 tentativi, continua a ricevere l'errore NoBrokersAvailable creando il KafkaProducer.	V
TU156	Il metodo create di KafkaProducerFactory chiude il programma se riceve un KeyboardInterrupt.	V
TU157	La classe base Producer viene costruita assegnando il parametro producer di tipo KafkaProducer all'attributo. _consumer	V
TU158	Il metodo send() della classe base Producer chiama il metodo send() del suo attributo _producer con i suoi parametri. _consumer	V
TU159	Il metodo close() della classe base Producer chiama il metodo close() dell'attributo _producer. _consumer	V
TU160	La classe FlaskServer viene costruita assegnando il parametro flask di tipo Flask all'attributo _app, e il parametro producer di tipo Producer all'attributo _producer.	V
TU161	Il metodo start() della classe FlaskServer chiama il metodo run() del suo attributo _app con i suoi parametri.	V
TU162	Il metodo _shutdown_app() della classe FlaskServer chiama il metodo request.environ.get(), e chiamando in seguito la funzione ritornata con successo.	V
TU163	Il metodo _shutdown_app() della classe FlaskServer chiama il metodo request.environ.get(), che ritorna None e lancia un RuntimeError.	V
TU164	Il metodo close() della classe FlaskServer chiama il suo metodo _shutdown_app() e il metodo close() del suo attributo _producer.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU165	La classe GitlabProducer viene creata passando per parametro un KafkaProducer e una stringa con l'url della repository	V
TU166	Il metodo produce() della classe GitlabProducer fa una chiamata al metodo send() della classe base passando per parametro una stringa che identifica il producer e i dati provenienti da Gitlab. Tale metodo può lanciare una Exception e il programma si chiude stampando il messaggio di errore.	V
TU167	Il metodo _parse() della classe GitlabProducer ritorna un dizionario con i dati desiderati provenienti da Gitlab per una nuova issue.	V
TU168	Il metodo _parse() della classe GitlabProducer ritorna un dizionario con i dati desiderati provenienti da Gitlab per una issue modificata.	V
TU169	Il metodo _parse() della classe GitlabProducer ritorna un dizionario con i dati desiderati provenienti da Gitlab per un push effettuato.	V
TU170	Il metodo _parse() della classe GitlabProducer ritorna un dizionario con i dati desiderati provenienti da Gitlab per un nota aggiunta.	V
TU171	Il metodo _parse() della classe GitlabProducer ritorna un dizionario con nulla ricevendo dati incompatibili.	V
TU172	Il metodo _find_note_keys_commit() della classe GitlabProducer ritorna la lista delle parole chiavi legate al commit ricevuta come parametro.	V
TU173	Il metodo _find_note_keys_nocommit() della classe GitlabProducer ritorna la lista delle parole chiavi legate al commit ricevuta come parametro.	V
TU174	Il metodo _find_issue_keys() della classe GitlabProducer ritorna la lista delle parole chiavi legate alla issue ricevuta come parametro.	V
TU175	Il metodo _find_new_issue_keys() della classe GitlabProducer ritorna la lista delle parole chiavi legate alla nuova issue ricevuta come parametro.	V
TU176	Il metodo _find_modified_issue_keys() della classe GitlabProducer ritorna la lista delle parole chiavi legate alla issue modificata ricevuta come parametro.	V
TU177	Il metodo _find_push_keys() della classe GitlabProducer ritorna la lista delle parole chiavi legate al push modificata ricevuta come parametro.	V
TU178	Il metodo start_gitlab_producer() della classe GitlabProducer chiama il metodo create() di KafkaProducerFactory creando un oggetto di KafkaProducer, esegue il suo metodo start() e in seguito esegue il suo metodo close() con successo.	V

– *continua a pagina successiva*

Tabella9– *continuazione da pagina precedente*

Codice	Descrizione	Stato
TU179	Il metodo start_gitlab_producer() della classe GitlabProducer chiama il metodo create() di KafkaProducerFactory creando un oggetto di KafkaProducer che fallisce lanciando una Exception, e il programma si chiude stampando il messaggio di errore.	V
TU180	Il metodo start_gitlab_producer() della classe GitlabProducer chiama il metodo create() di KafkaProducerFactory creando un oggetto di KafkaProducer ma durante la sua esecuzione viene lanciato l'eccezione KeyboardInterrupt, e il programma si chiude.	V
TU181	La classe RedmineProducer viene creata passando per parametro un KafkaProducer e una stringa con l'indirizzo IP del server	V
TU182	Il metodo produce() della classe RedmineProducer fa una chiamata al metodo send() della classe base passando per parametro una stringa che identifica il producer e i dati provenienti da Redmine. Tale metodo può lanciare una Exception e il programma si chiude stampando il messaggio di errore.	V
TU183	Il metodo _parse() della classe RedmineProducer ritorna un dizionario con i dati desiderati provenienti da Redmine	V
TU184	Il metodo start_redmine_producer() della classe RedmineProducer chiama il metodo create() di KafkaProducerFactory creando un oggetto di KafkaProducer, esegue il suo metodo start() e in seguito esegue il suo metodo close() con successo.	V
TU185	La classe Application viene creata passando per parametro un oggetto di tipo Flask e uno di tipo Mongo.	V
TU186	Il metodo _shutdown_app() della classe Application chiama il metodo request.environ.get()che ritorna None e lancia un RuntimeError.	V

A Resoconto dell'attività di verifica

A.1 Revisione dei Requisiti

Nel periodo antecedente la consegna della Revisione dei Requisiti sono state effettuate le attività di verifica dei documenti e dei processi.
La documentazione è stata verificata secondo le procedure descritte nelle *norme di progetto*, nello specifico si è utilizzata la tecnica del *walkthrough* per una analisi statica del documento. Questa analisi ha fatto emergere alcuni errori che sono stati trattati nel seguente modo:

- Correzione degli errori rilevati;
- Inserimento degli errori più frequenti nella lista di controllo;
- Applicazione del ciclo PDCA in modo da ottimizzare l'applicazione del metodo di inspection e rendere più efficiente ed efficace il processo di verifica.

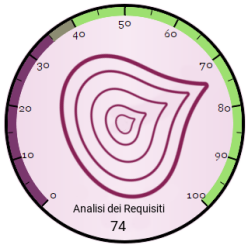
Infine si sono calcolate le metriche descritte nella sezione seguenti.
L'avanzamento dei processi è stato controllato e verificato applicando la procedura descritta nell'apposita sezione delle *norme di progetto*, quindi sono stati valutati e riportati i valori di Schedule Variance e Budget Variance. È stata in seguito applicata l'inspection utilizzando la lista di controllo stilata durante la veri!ca dei documenti precedentemente verificati, ponendo particolare attenzione ai casi d'uso. Corretti gli errori si è cercato di migliorare il processo che ha manifestato l'errore tramite cicli di PDCA e si è controllato che tutte le metriche fossero nel range di accettazione.

A.1.1 Analisi sui documenti

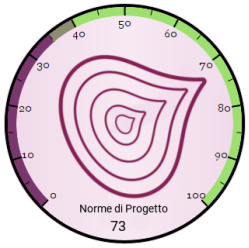
A.1.1.1 MPDD01: Indice Gulpease

Documento	Punteggio	Esito
Analisi dei requisiti v1.0.0	74	superato
Norme di progetto v1.0.0	73	superato
Piano di qualifica v1.0.0	75	superato
Piano di progetto v1.0.0	76	superato
Glossario v1.0.0	66	superato
Studio di fattibilità v1.0.0	73	superato

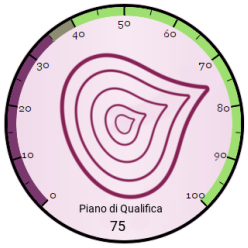
Tabella 10: Esiti indici di Gulpease



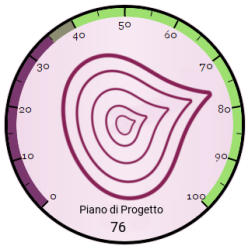
(a) Analisi dei requisiti v1.0.0



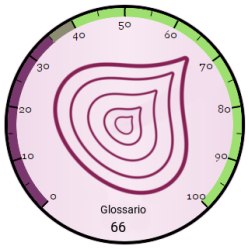
(b) Norme di progetto v1.0.0



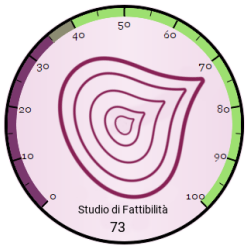
(c) Piano di qualifica v1.0.0



(d) Piano di progetto v1.0.0



(e) Glossario v1.0.0



(f) Studio di fattibilità v1.0.0

A.1.2 Analisi sui processi

A.1.2.1 MPR01: Schedule Variance

Attività	Punteggio	Valutazione
Analisi dei requisiti v1.0.0	−3	accettabile
Norme di progetto v1.0.0	−8	inaccettabile
Piano di qualifica v1.0.0	−5	accettabile
Piano di progetto v1.0.0	−1	accettabile
Glossario v1.0.0	−3	accettabile
Studio di fattibilità v1.0.0	−4	accettabile

Tabella 11: Esiti indici dello Schedule Variance

Considerando che, per ogni stesura, si è riscontrato una ritardo di produzione di almeno 3 giorni rispetto alle previsioni, per la prossima consegna verrano stabilite delle date di previsione più generose.

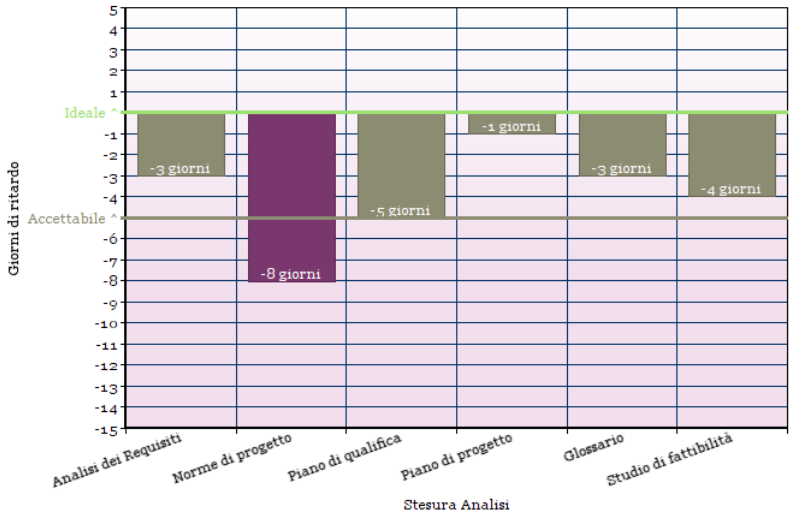


Figura 2: Esito schedule variance prima del RR

A.1.2.2 MPR02: Budget Variance

Attività	Punteggio	Esito
Analisi	2.954	ottimale
Consolidamento dei Requisiti	2.954	ottimale

Tabella 12: Esiti indice del Budget Variance

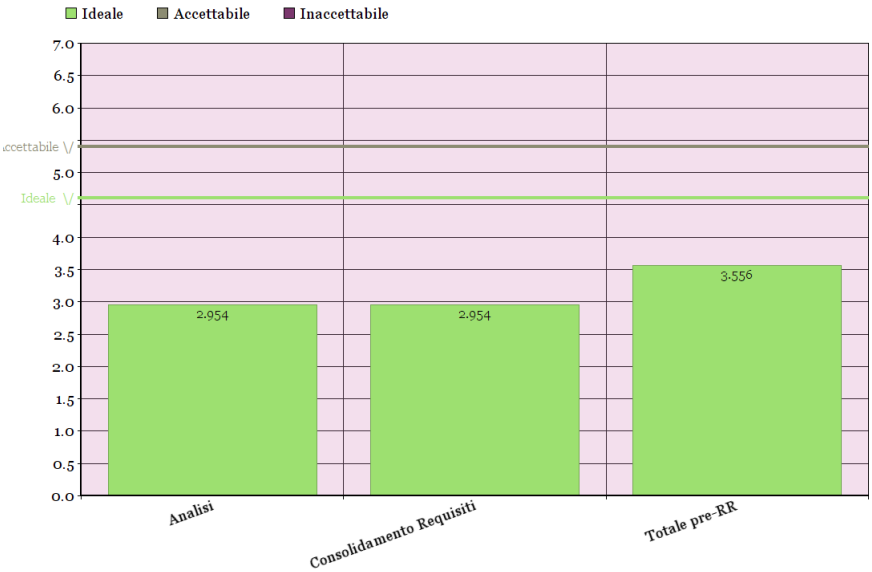


Figura 3: Esiti del Budget Variance prima del RR

A.2 Revisione di Progetto

Nel periodo precedente alla consegna per la Revisione di Progettazione, sono stati verificati i documenti redatti ponendo particolare attenzione alle sezioni aggiunte o modificate dei vari documenti rispetto alla loro precedente versione.
La documentazione è stata verificata secondo le procedure descritte nelle *norme di progetto*, nello specifico si è utilizzata la tecnica del *walkthrough* per una analisi statica del documento. Questa analisi ha fatto emergere alcuni errori che sono stati trattati nel seguente modo:

- Correzione degli errori rilevati;
- Inserimento degli errori più frequenti nella lista di controllo;
- Applicazione del ciclo PDCA in modo da ottimizzare l'applicazione del metodo di inspection e rendere più efficiente ed efficace il processo di verifica.

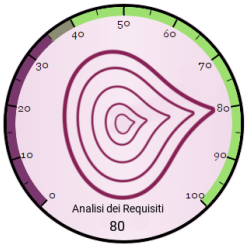
Si è poi passato al metodo *Inspection*, usando la lista di controllo stilata per controllare i documenti precedentemente verificati, ponendo maggiore attenzione agli errori più frequenti. Infine si sono calcolate le nuove metriche che sono descritte nella sezione seguente.
L'avanzamento dei processi è stato controllato e verificato applicando la procedura descritta nell'apposita sezione delle *norme di progetto*, quindi sono stati valutati i valori di Schedule Variance e Budget Variance e riportati qui di seguito.

A.2.1 Analisi sui documenti

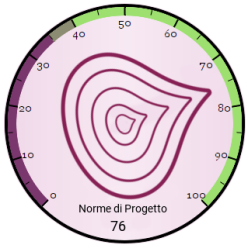
A.2.1.1 MPDD01: Indice Gulpease

Documento	Punteggio	Esito
Analisi dei requisiti v2.0.0	80	superato
Norme di progetto v2.0.0	76	superato
Piano di qualifica v2.0.0	66	superato
Piano di progetto v2.0.0	70	superato
Glossario v2.0.0	70	superato

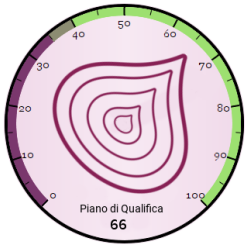
Tabella 13: Esiti indici di Gulpease



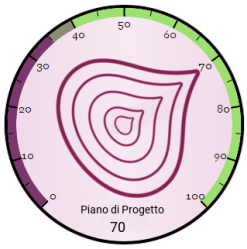
(a) Analisi dei requisiti v2.0.0



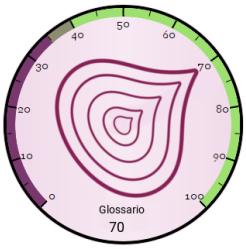
(b) Norme di progetto v2.0.0



(c) Piano di qualifica v2.0.0



(d) Piano di progetto v2.0.0



(e) Glossario v2.0.0

A.2.2 Analisi sui processi

A.2.2.1 MPR01: Schedule Variance

Attività	Punteggio	Valutazione
Analisi dei requisiti v2.0.0	0	Ideale
Norme di progetto v2.0.0	+2	Ideale
Piano di qualifica v2.0.0	+1	Ideale
Piano di progetto v2.0.0	+1	Ideale
Glossario v2.0.0	0	Ideale
Progettazione ad alto livello	+1	Ideale

Tabella 14: Esiti indici dello Schedule Variance

L'Esito del *schedule variance* risulta essere ideale mostrando che l'organizzazione del team è migliorata rispetto alla fase precedente.

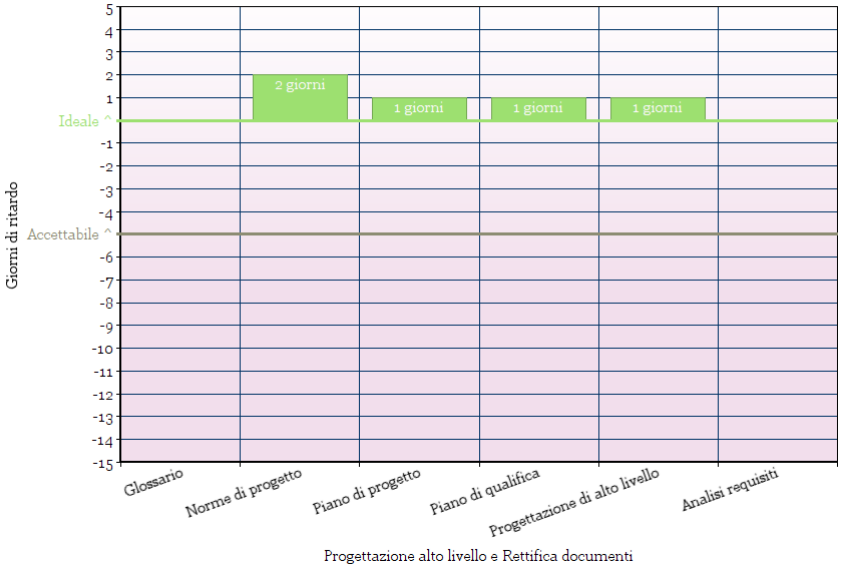


Figura 5: Esito schedule variance prima del RP

A.2.2.2 MPR02: Budget Variance

Attività	Punteggio	Esito
Progettazione architetturale	3.690	Ideale

Tabella 15: Esiti indice del Budget Variance

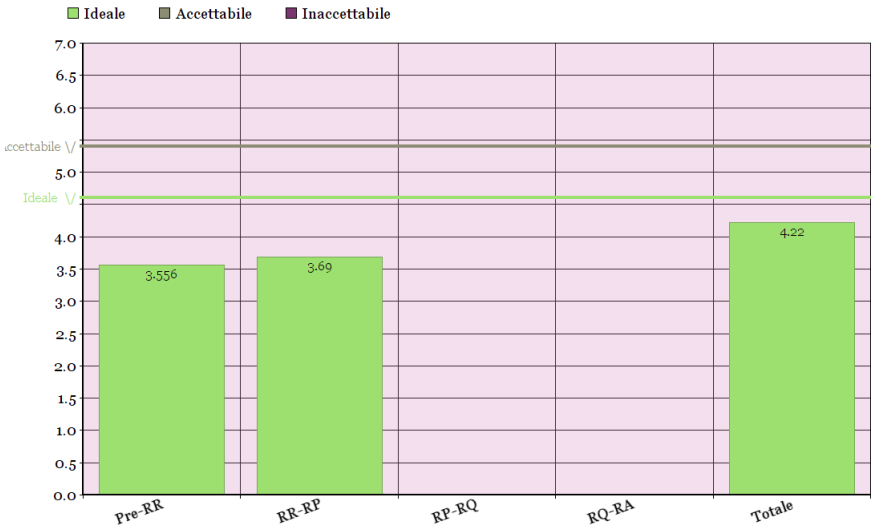


Figura 6: Esiti del Budget Variance prima del RP

A.3 Revisione di Qualifica

Nel periodo precedente alla consegna per la Revisione di Qualifica, sono stati verificati i documenti redatti ponendo particolare attenzione alle sezioni aggiunte o modificate dei vari documenti rispetto alla loro precedente versione.
La documentazione è stata verificata secondo le procedure descritte nelle *norme di progetto*, nello specifico si è utilizzata la tecnica del *walkthrough* per una analisi statica del documento. Questa analisi ha fatto emergere alcuni errori che sono stati trattati nel seguente modo:

- Correzione degli errori rilevati;
- Inserimento degli errori più frequenti nella lista di controllo;
- Applicazione del ciclo PDCA in modo da ottimizzare l'applicazione del metodo di inspection e rendere più efficiente ed efficace il processo di verifica.

Si è poi passato al metodo *Inspection*, usando la lista di controllo stilata per controllare i documenti precedentemente verificati, ponendo maggiore attenzione agli errori più frequenti. Infine si sono calcolate le nuove metriche che sono descritte nella sezione seguente.
L'avanzamento dei processi e del prodotto sono stati controllati e verificati applicando la procedura descritta nell'apposita sezione delle *norme di progetto*, quindi sono stati valutati i valori calcolati nelle metriche riportati in seguito.

A.3.1 Considerazioni sulle metriche utilizzate

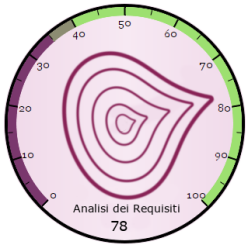
Va fatto notare che in questo periodo di lavoro ci siamo dedicati prettamente alla codifica e progettazione del software e solo in seguito allo sviluppo degli opportuni test, com'è previsto secondo il *modello a V*, di conseguenza alcune metriche calcolate sono da considerarsi poco attendibili in quanto la vera e propria attività di testing è prevista per il prossimo periodo di lavoro, ossia post RQ; in particolare le metriche riguardanti lo sviluppo del prodotto software (codificate con MDPS e MPRS secondo il documento *norme di progetto*) risultano per la maggior parte non accettabili, ma è da considerarsi regolare.

A.3.2 Analisi sui documenti

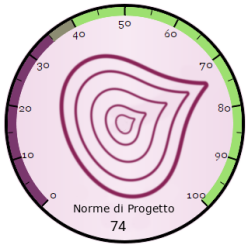
A.3.2.1 MPDD01: Indice Gulpease

Documento	Punteggio	Esito
Analisi dei requisiti v3.0.0	78	superato
Norme di progetto v3.0.0	74	superato
Piano di qualifica v3.0.0	68	superato
Piano di progetto v3.0.0	72	superato
Manuale sviluppatore v1.0.0	52	superato
Manuale utente v1.0.0	65	superato
Glossario v3.0.0	69	superato

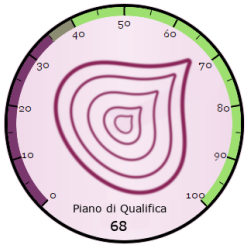
Tabella 16: Esiti indici di Gulpease



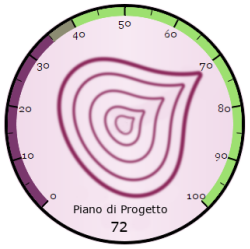
(a) Analisi dei requisiti v3.0.0



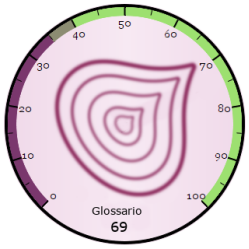
(b) Norme di progetto v3.0.0



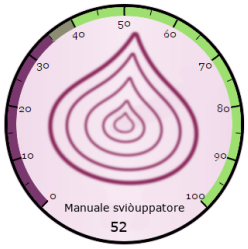
(c) Piano di qualifica v3.0.0



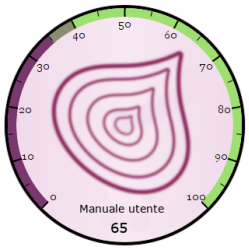
(d) Piano di progetto v3.0.0



(e) Glossario v3.0.0



(f) Manuale sviluppatore v1.0.0



(g) Manuale utente v1.0.0

A.3.3 Analisi sui processi

A.3.3.1 MPR01: Schedule Variance

Attività	Punteggio	Valutazione
Analisi dei requisiti v3.0.0	1	Ideale
Norme di progetto v3.0.0	0	Ideale
Piano di qualifica v3.0.0	0	Ideale
Piano di progetto v3.0.0	0	Ideale
Glossario v3.0.0	0	Ideale
Progettazione	-2	Accettabile
Codifica	2	Ideale
Manuale sviluppatore v1.0.0	0	Ideale
Manuale utente v1.0.0	0	Ideale

Tabella 17: Esiti indici dello Schedule Variance

L'Esito del *schedule variance* risulta essere secondo le aspettative.

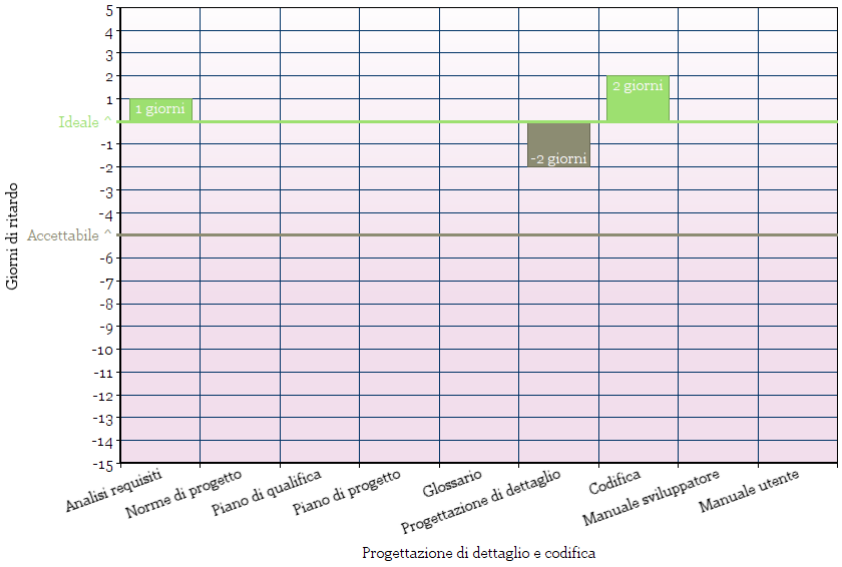


Figura 8: Esito schedule variance prima del RQ

A.3.3.2 MPR02: Budget Variance

Attività	Punteggio	Esito
Progettazione architetturale e codifica	5.042	Accettabile

Tabella 18: Esiti indice del Budget Variance

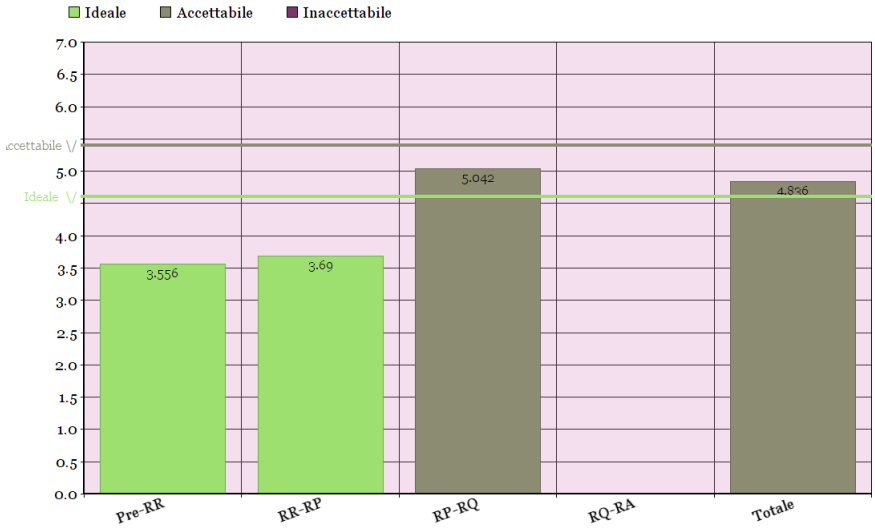


Figura 9: Esiti del Budget Variance prima del RQ

A.3.3.3 MPRS01: Code Coverage

In quanto non sono stati implementati tutti i test di controllo qualità il valore delle metriche generate è da considerarsi indicativo e prossimo a cambiamento.
Valore atteso $\geq 80\%$
Valore ottenuto 76%

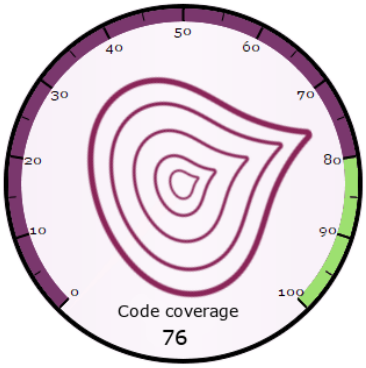
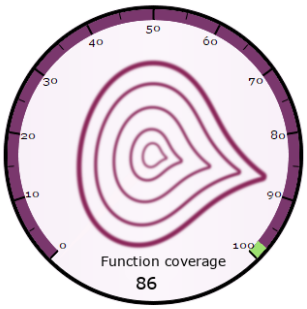
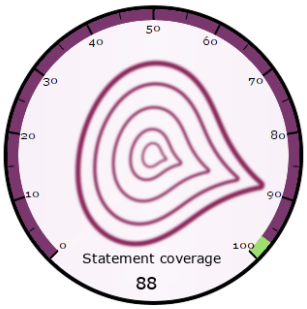


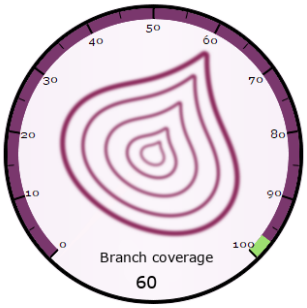
Figura 10: Esito code coverage



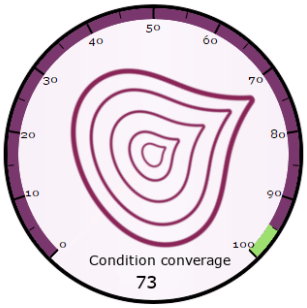
(a) MPRS02: Function coverage
Valore atteso: 98%
Valore ottenuto: 86%



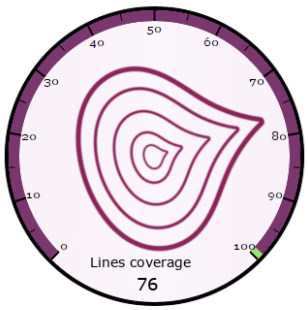
(b) MPRS02: Statement coverage
Valore atteso: 97%
Valore ottenuto: 88%



(c) MPRS04: Branch coverage
Valore atteso: 95%
Valore ottenuto: 60%



(d) MPRS05: Condition coverage
Valore atteso: 95%
Valore ottenuto: 73%



(e) MPRS06: Lines coverage
Valore atteso: 99%
Valore ottenuto: 76%

A.3.4 Analisi sul prodotto

A.3.4.1 MPDS01: Copertura requisiti obbligatori

Indica la percentuale dei requisiti obbligatori coperti dall'implementazione.
Valore atteso: $\geq 100\%$
Valore ottenuto 165 su 176 quindi il 94%

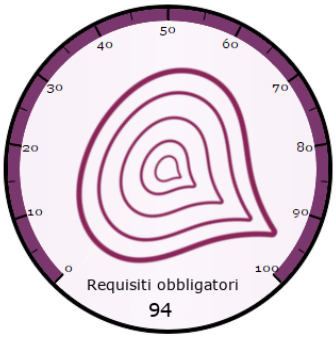


Figura 12: Esito copertura requisiti

A.3.4.2 MPDS02: Copertura requisiti accettati

Indica la percentuale dei requisiti desiderabili e facoltativi coperti dall'implementazione.
Valore atteso: $\geq 60\%$
Valore ottenuto 7 su 9 quindi il 76%

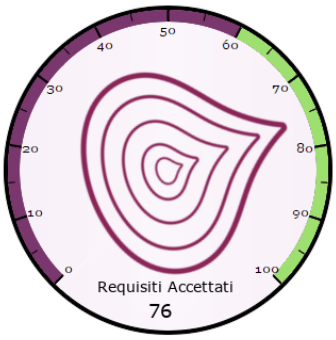


Figura 13: Esito copertura requisiti accettati

A.3.4.3 MPDS09: Accoppiamento di componenti

Indica quanto strettamente sono indipendenti i componenti e quanti componenti sono esenti da impatti da cambiamenti negli altri componenti.
Valore atteso: ≤ 20
Valore ottenuto: 6

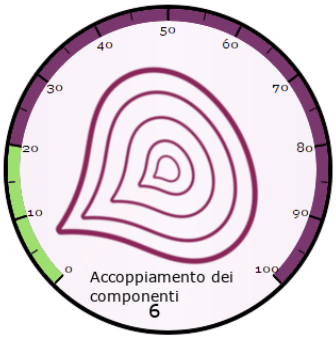


Figura 14: Esito accoppiamento componenti

A.3.4.4 MPDS10: Interfaccia utente auto-esplicativa

Indica la percentuale degli elementi d'informazione e dei passi che sono presentati all'utente inesperto in modo che questi possa completare un'attività senza un addestramento preliminare o assistenza esterna.
Valore atteso: $\geq 60\%$
Valore ottenuto: 86%

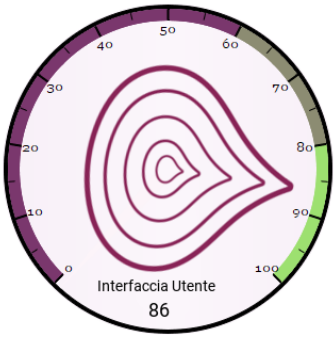


Figura 15: Esito interfaccia auto-esplicativa

A.3.4.5 MPDS06: Densità errori

Indica la percentuale dei test che si sono concluse con un fallimento.
Valore atteso: $\leq 30\%$
Valore ottenuto: 70%

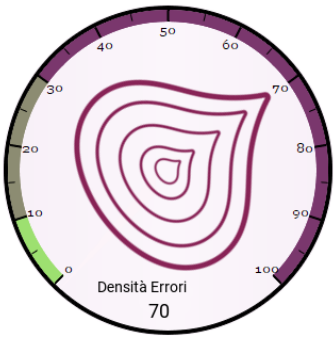


Figura 16: Esito densità errori

A.3.4.6 MPDS07: Capacità di analisi degli errori

Indica la percentuale di modifiche effettuate in risposta agli errori che hanno portato all'introduzione di nuovi errori in altre componenti del sistema.
Valore atteso: $\geq 40\%$
Valore ottenuto: 77%

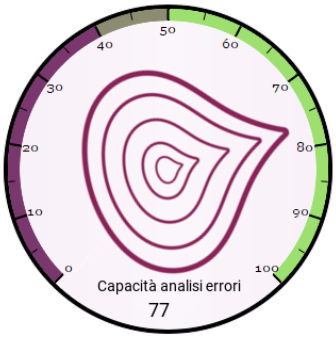


Figura 17: Esito analisi errori

A.3.4.7 MPDS08: Efficienza delle modifiche

Indica la percentuale di modifiche effettuate in risposta agli errori che hanno portato all'introduzione di nuove errori in altre componenti del sistema.
Valore atteso: $\geq 80\%$
Valore ottenuto: 50%

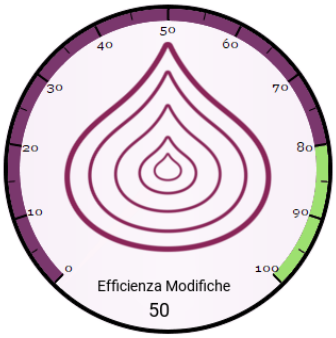


Figura 18: Esito efficienza modifiche

A.3.4.8 MPDS12: Duplicazione del codice

Indica la percentuale di codice duplicato rilevato da SonarQube.
Valore atteso: $\leq 20\%$
Valore ottenuto: $0,6\%$

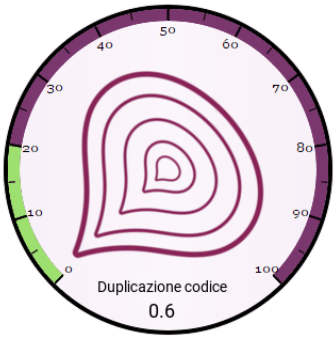


Figura 19: Esito efficienza modifiche

A.3.5 Analisi sui test

A.3.5.1 TS01: Test passati

Test totali: 117
Test passati: 59
Valore atteso: $\geq 80\%$
Valore ottenuto: 50%

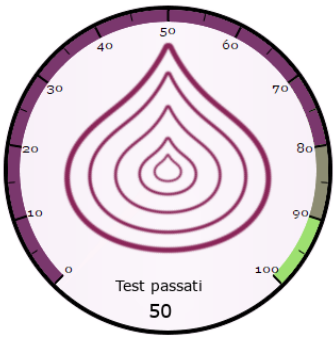


Figura 20: Test passati

A.3.5.2 TS02: Test falliti

Test totali: 117
Test passati: 58
Valore atteso: $\leq 20\%$
Valore ottenuto: 50%

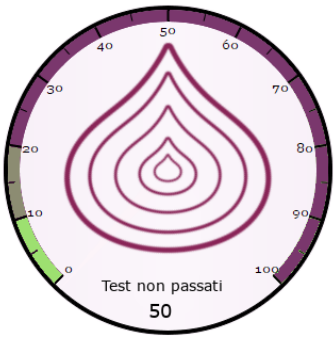


Figura 21: Test falliti

A.3.5.3 TS03: Test eseguiti in rapporto ai requisiti

Test totali sui requisiti: 176
Test totali eseguiti: 52
Valore atteso: 100%
Valore ottenuto: 30%

Figura 22: Test eseguiti in rapporto ai requisiti

A.4 Revisione di Accettazione

Nel periodo precedente alla consegna per la Revisione di Accettazione, sono stati verificati i documenti redatti ponendo particolare attenzione alle sezioni aggiunte o modificate dei vari documenti rispetto alla loro precedente versione.
La documentazione è stata verificata secondo le procedure descritte nelle *norme di progetto*, nello specifico si è utilizzata la tecnica del *walkthrough* per una analisi statica del documento. Questa analisi ha fatto emergere alcuni errori che sono stati trattati nel seguente modo:

- Correzione degli errori rilevati;
- Inserimento degli errori più frequenti nella lista di controllo;
- Applicazione del ciclo PDCA in modo da ottimizzare l'applicazione del metodo di inspection e rendere più efficiente ed efficace il processo di verifica.

Si è poi passato al metodo *Inspection*, usando la lista di controllo stilata per controllare i documenti precedentemente verificati, ponendo maggiore attenzione agli errori più frequenti. Infine si sono calcolate le nuove metriche che sono descritte nella sezione seguente.
L'avanzamento dei processi e del prodotto sono stati controllati e verificati applicando la procedura descritta nell'apposita sezione delle *norme di progetto*, quindi sono stati valutati i valori calcolati nelle metriche riportati in seguito.

A.4.1 Considerazioni sulle metriche utilizzate

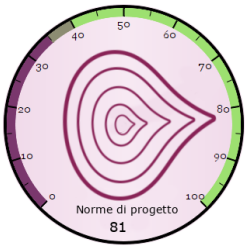
Va fatto notare che in questo periodo di lavoro ci siamo dedicati prettamente alla produzione di test e alla loro applicazione per garantire il passaggio degli standard qualitativi che ci siamo imposti negli scorsi periodi. Com'è previsto secondo il *modello a V*, abbiamo implementato e verificato i test di unità, integrazione e sistema, di conseguenza le metriche calcolate sono da considerarsi attendibili in quanto la vera e propria attività di testing è stata ultimata in questo periodo di lavoro, ossia RA.

A.4.2 Analisi sui documenti

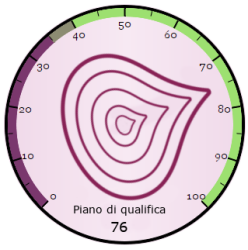
A.4.2.1 MPDD01: Indice Gulpease

Documento	Punteggio	Esito
Norme di progetto v4.0.0	81	superato
Piano di qualifica v4.0.0	76	superato
Piano di progetto v4.0.0	79	superato
Manuale sviluppatore v2.0.0	79	superato
Manuale utente v2.0.0	70	superato
Glossario v4.0.0	69	superato

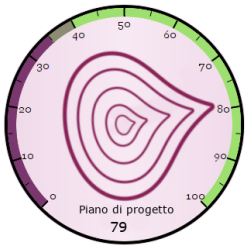
Tabella 19: Esiti indici di Gulpease



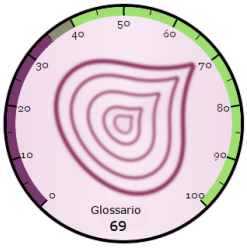
(a) Norme di progetto v4.0.0



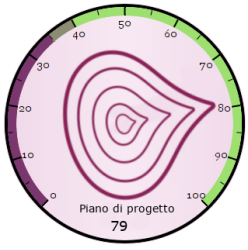
(b) Piano di qualifica v4.0.0



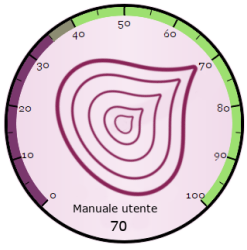
(c) Piano di progetto v4.0.0



(d) Glossario v4.0.0



(e) M sviluppatore v2.0.0



(f) M utente v2.0.0

A.4.3 Analisi sui processi

A.4.3.1 MPR01: Schedule Variance

Attività	Punteggio	Valutazione
Norme di progetto v4.0.0	3	Ideale
Piano di qualifica v4.0.0	0	Ideale
Piano di progetto v4.0.0	0	Ideale
Glossario v4.0.0	5	Ideale
Validazione	1	Ideale
Collaudo	0	Ideale
Manuale sviluppatore v2.0.0	2	Ideale
Manuale utente v2.0.0	1	Ideale

Tabella 20: Esiti indici dello Schedule Variance

L'Esito del *schedule variance* risulta essere secondo le aspettative.

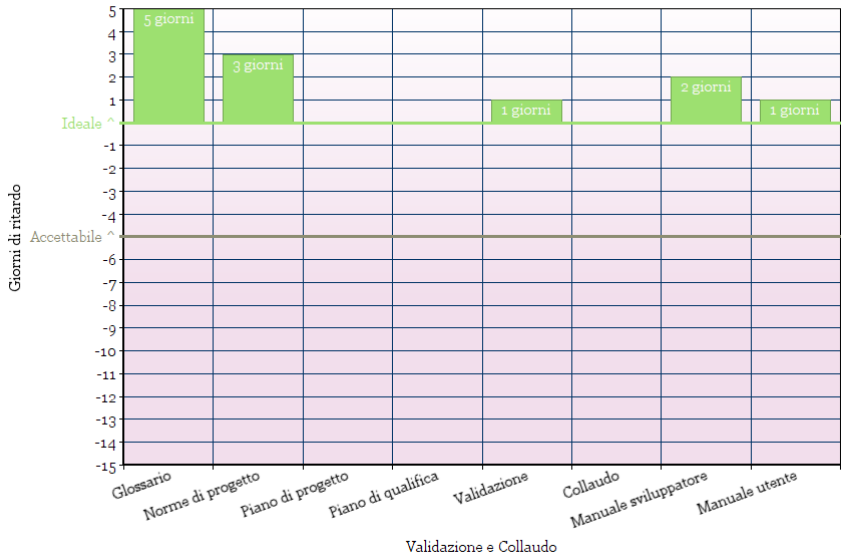


Figura 24: Esito schedule variance prima del RA

A.4.3.2 MPR02: Budget Variance

Attività	Punteggio	Esito
Validazione e Collaudo	4.823	Accettabile
Totale	5.397	Accettabile

Tabella 21: Esiti indice del Budget Variance

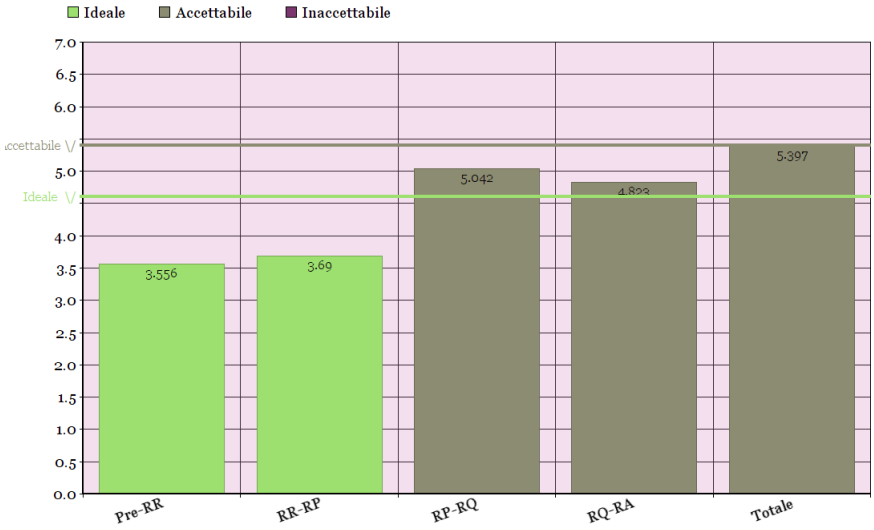


Figura 25: Esiti del Budget Variance prima del RA

A.4.3.3 MPRS01: Code Coverage

Valore atteso $\geq 80\%$
Valore ottenuto 80%

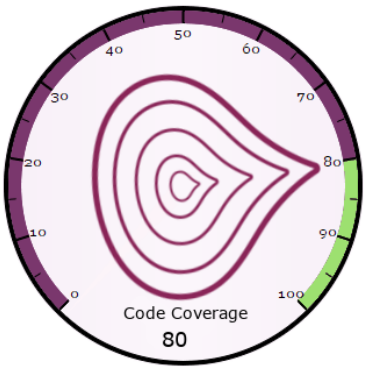
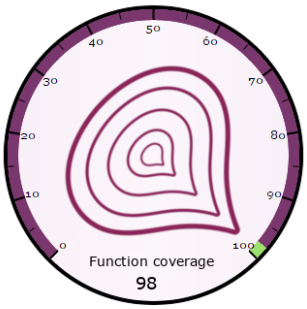
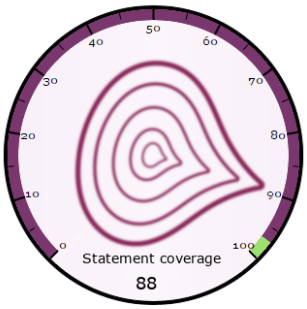


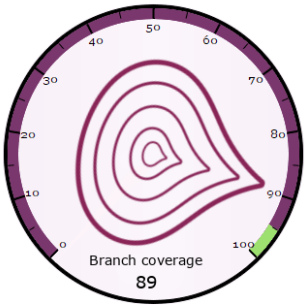
Figura 26: Esito code coverage



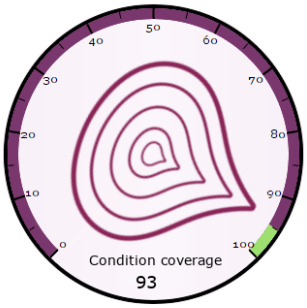
(a) MPRS02: Function coverage
Valore atteso: 98%
Valore ottenuto: 98%



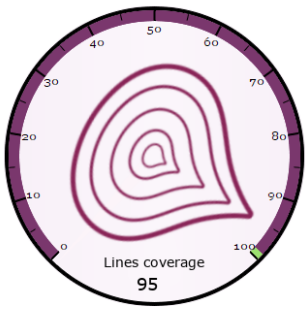
(b) MPRS02: Statement coverage
Valore atteso: 97%
Valore ottenuto: 88%



(c) MPRS04: Branch coverage
Valore atteso: 95%
Valore ottenuto: 89%



(d) MPRS05: Condition coverage
Valore atteso: 95%
Valore ottenuto: 93%



(e) MPRS06: Lines coverage
Valore atteso: 99%
Valore ottenuto: 95%

A.4.4 Analisi sul prodotto

A.4.4.1 MPDS01: Copertura requisiti obbligatori

Indica la percentuale dei requisiti obbligatori coperti dall'implementazione.
Valore atteso: $\geq 100\%$
Valore ottenuto 176 su 176 quindi il 100%

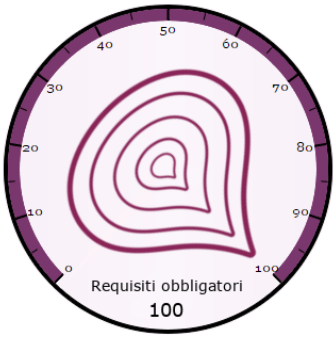


Figura 28: Esito copertura requisiti

A.4.4.2 MPDS02: Copertura requisiti accettati

Indica la percentuale dei requisiti desiderabili e facoltativi coperti dall'implementazione.
Valore atteso: $\geq 60\%$
Valore ottenuto 7 su 9 quindi il 76%

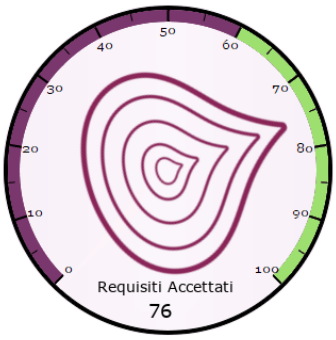


Figura 29: Esito copertura requisiti accettati

A.4.4.3 MPDS09: Accoppiamento di componenti

Indica quanto strettamente sono indipendenti i componenti e quanti componenti sono esenti da impatti da cambiamenti negli altri componenti.
Valore atteso: ≤ 20
Valore ottenuto: 6

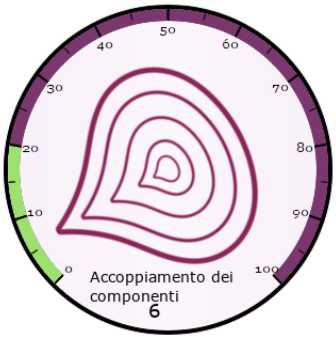


Figura 30: Esito accoppiamento componenti

A.4.4.4 MPDS10: Interfaccia utente auto-esplicativa

Indica la percentuale degli elementi d'informazione e dei passi che sono presentati all'utente inesperto in modo che questi possa completare un'attività senza un addestramento preliminare o assistenza esterna.
Valore atteso: $\geq 60\%$
Valore ottenuto: 86%

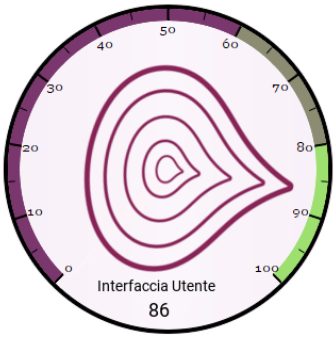


Figura 31: Esito interfaccia auto-esplicativa

A.4.4.5 MPDS06: Densità errori

Indica la percentuale dei test che si sono concluse con un fallimento.
Valore atteso: $\leq 30\%$
Valore ottenuto: 20%

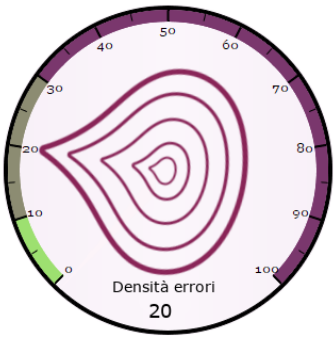


Figura 32: Esito densità errori

A.4.4.6 MPDS07: Capacità di analisi degli errori

Indica la percentuale di modifiche effettuate in risposta agli errori che hanno portato all'introduzione di nuovi errori in altre componenti del sistema.
Valore atteso: $\geq 40\%$
Valore ottenuto: 85%

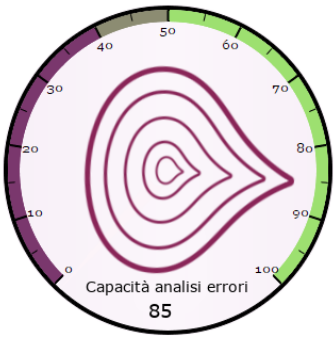


Figura 33: Esito analisi errori

A.4.4.7 MPDS08: Efficienza delle modifiche

Indica la percentuale di modifiche effettuate in risposta agli errori che hanno portato all'introduzione di nuove errori in altre componenti del sistema.
Valore atteso: $\geq 80\%$
Valore ottenuto: 87%

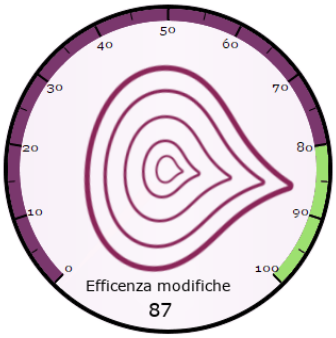


Figura 34: Esito efficienza modifiche

A.4.4.8 MPDS12: Duplicazione del codice

Indica la percentuale di codice duplicato rilevato da SonarQube.
Valore atteso: $\leq 20\%$
Valore ottenuto: 0%

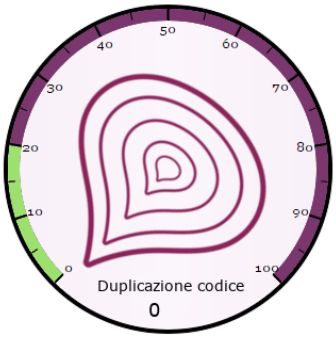


Figura 35: Esito efficienza modifiche

A.4.4.9 MPDS03: Accuratezza rispetto alle attese

Indica la percentuale di risultati concordi alle attese.
Valore atteso: $\geq 60\%$
Valore ottenuto: 85%

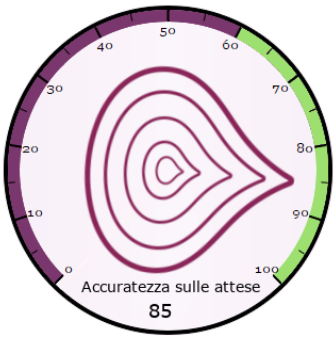


Figura 36: Esito accuratezza rispetto alle attese

A.4.4.10 MPDS04: Tempo medio di risposta

Indica il tempo medio che intercorre fra la richiesta software di una determinata funzionalità e la restituzione del risultato all'utente.
Valore atteso: $\leq 5s$
Valore ottenuto: 2s
In questa metrica è considerato anche la metrica MPDS05 ossia l'adeguatezza del tempo di risposta ossia:
Valore atteso: $\geq 40\%$
Valore ottenuto: 74%

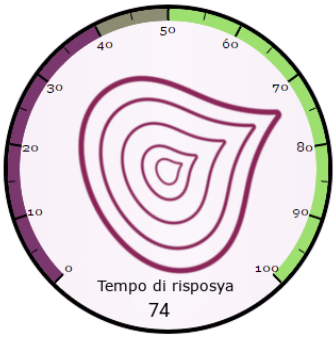


Figura 37: Esito tempo medio di risposta

A.4.4.11 MPDS11: Comprensibilità dei messaggi d'errore

Indica la percentuale dei messaggi d'errore che dichiarano la ragione dell'errore e suggeriscono come risolverlo.
Valore atteso: $\geq 70\%$
Valore ottenuto: 74%

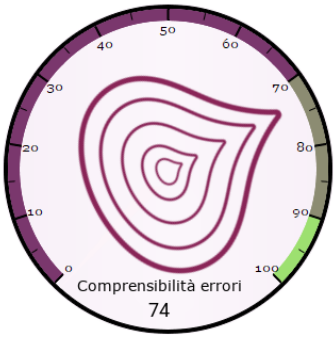


Figura 38: Esito comprensibilità dei messaggi

A.4.5 Analisi sui test

A.4.5.1 TS01: Test passati

Test totali: 250
Test passati: 250
Valore atteso: $\geq 80\%$
Valore ottenuto: 100%

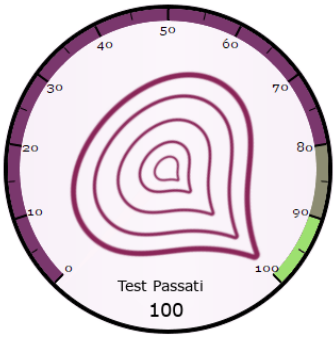


Figura 39: Test passati

A.4.5.2 TS02: Test falliti

Test totali: 250
Test falliti: 0
Valore atteso: $\leq 20\%$
Valore ottenuto: 0%

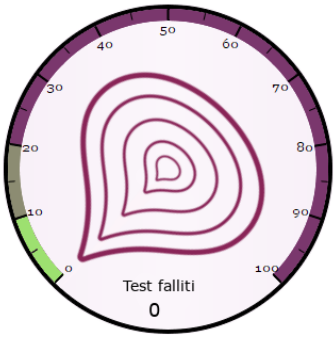


Figura 40: Test falliti

A.4.5.3 TS03: Test eseguiti in rapporto ai requisiti

Test totali sui requisiti: 172 Test totali eseguiti: 172
Valore atteso: 100%
Valore ottenuto: 100%

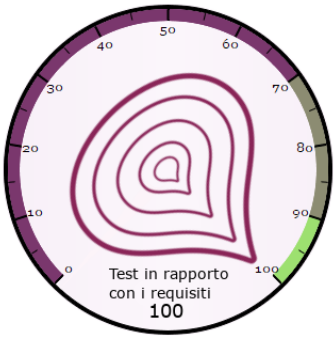


Figura 41: Test eseguiti in rapporto ai requisiti