



Piano di Qualifica

7DOS - 22 Marzo 2019

Informazioni sul documento

Versione	3.0.0
Responsabile	Giovanni Sorice
Verifica	Marco Costantino Nicolò Tartaggia
Redazione	Lorenzo Busin Giacomo Barzon Michele Roverato Andrea Trevisin
Stato	Approvato
Uso	Esterno
Destinato a	Prof.Tullio Vardanega Prof.Riccardo Cardin Zucchetti 7DOS
Email	7dos.swe@gmail.com

Descrizione

Questo documento descrive le operazioni di verifica e validazione relative al progetto *G&B*.

Diario delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
3.0.0	2019-03-22	<i>Approvazione del documento per la RQ</i>	Giovanni Sorice	Responsabile
2.7.0	2019-03-21	<i>Verifica del documento</i>	Nicolò Tartaggia	Verificatore
2.6.0	2019-03-20	<i>Verifica del documento</i>	Marco Costantino	Verificatore
2.5.0	2019-03-19	<i>Fine stesura §C.4</i>	Lorenzo Busin	Amministratore
2.4.3	2019-03-19	<i>Stesura §C.3</i>	Andrea Trevisin	Amministratore
2.4.2	2019-03-19	<i>Stesura §C.2</i>	Giacomo Barzon	Amministratore
2.4.1	2019-03-19	<i>Inizio stesura §C.4</i>	Lorenzo Busin	Amministratore
2.4.0	2019-03-19	<i>Stesura §C.1 - (vedi Verbale del 2019-03-18)</i>	Michele Roverato	Amministratore
2.3.0	2019-03-18	<i>Stesura §4</i>	Giacomo Barzon	Amministratore
2.2.1	2019-03-18	<i>Inseriti grafici misurazioni in §B.2 - (vedi Verbale del 2019-03-18)</i>	Lorenzo Busin	Amministratore
2.2.0	2019-03-18	<i>Stesura §A</i>	Giacomo Barzon	Amministratore
2.1.0	2019-03-18	<i>Spostata specifica dei test in §4</i>	Andrea Trevisin	Amministratore
2.0.1	2019-03-18	<i>Rimossi contenuti che attenevano alle Norme da §2 e §3 - (vedi Verbale del 2019-03-18)</i>	Michele Roverato	Amministratore
2.0.0	2019-02-07	<i>Approvazione del documento per la RP</i>	Marco Costantino	Responsabile
1.8.0	2019-02-06	<i>Verifica del documento</i>	Giacomo Barzon	Verificatore
1.7.0	2019-02-05	<i>Verifica del documento</i>	Michele Roverato	Verificatore
1.6.1	2019-02-04	<i>Miglioramento sezione §4.3</i>	Lorenzo Busin	Amministratore
1.6.0	2019-02-02	<i>Correzioni a §2.3</i>	Nicolò Tartaggia	Amministratore
1.5.0	2019-02-01	<i>Miglioramenti a §2.1.5 e §2.1.6</i>	Lorenzo Busin	Amministratore

Versione	Data	Descrizione	Autore	Ruolo
1.4.0	2019-01-30	<i>Verifica del documento</i>	Giacomo Barzon	Verificatore
1.3.0	2019-01-29	<i>Verifica del documento</i>	Michele Roverato	Verificatore
1.2.0	2019-01-21	<i>Spostati contenuti in §A</i>	Andrea Trevisin	Amministratore
1.1.0	2019-01-21	<i>Rimossi contenuti che attengono alle Norme</i>	Giovanni Sorice	Amministratore
1.0.0	2019-01-01	<i>Approvazione del documento per la RR</i>	Marco Costantino	Responsabile
0.5.0	2019-01-01	<i>Verifica del documento</i>	Michele Roverato	Verificatore
0.4.0	2018-12-29	<i>Verifica del documento</i>	Lorenzo Busin	Verificatore
0.3.1	2018-12-28	<i>Miglioramenti a §A.1</i>	Giovanni Sorice	Amministratore
0.3.0	2018-12-26	<i>Stesura §A</i>	Giovanni Sorice	Amministratore
0.2.2	2018-12-13	<i>Completamento stesura §2</i>	Nicolò Tartaggia	Amministratore
0.2.1	2018-12-13	<i>Miglioramenti a §2.3</i>	Giacomo Barzon	Analista
0.2.0	2018-12-05	<i>Stesura §2.3</i>	Giacomo Barzon	Analista
0.1.0	2018-12-05	<i>Inizio stesura sezione §2</i>	Nicolò Tartaggia	Amministratore
0.0.2	2018-12-05	<i>Stesura della sezione §1</i>	Andrea Trevisin	Amministratore
0.0.1	2018-12-05	<i>Stesura dello scheletro del documento</i>	Andrea Trevisin	Amministratore

Indice

1	Introduzione	8
1.1	Scopo del documento	8
1.2	Scopo del prodotto	8
1.3	Glossario	8
1.4	Maturità del documento	8
1.5	Riferimenti	8
1.5.1	Normativi	8
1.5.2	Informativi	9
2	Qualità di processo	10
2.1	Pianificazione e controllo	10
2.1.1	Obiettivi	10
2.1.2	Metriche	10
2.2	Gestione dei rischi	10
2.2.1	Obiettivi	11
2.2.2	Metriche	11
2.3	Gestione dei test	11
2.3.1	Obiettivi	11
2.3.2	Metriche	11
2.4	Versionamento e build	12
2.4.1	Obiettivi	12
2.4.2	Metriche	12
2.5	Riassunto delle metriche di processo	13
3	Qualità di prodotto	14
3.1	Qualità dei documenti	14
3.1.1	Obiettivi	14
3.1.2	Metriche	14
3.2	Qualità del software	14
3.2.1	Functional Suitability	14
3.2.1.1	Obiettivi	15
3.2.1.2	Metriche	15
3.2.2	Performance Efficiency	15
3.2.2.1	Obiettivi	15
3.2.2.2	Metriche	15
3.2.3	Usability	15
3.2.3.1	Obiettivi	15
3.2.3.2	Metriche	16
3.2.4	Reliability	16
3.2.4.1	Obiettivi	16
3.2.4.2	Metriche	16
3.2.5	Maintainability	16
3.2.5.1	Obiettivi	16
3.2.5.2	Metriche	17

3.3	Riassunto delle metriche di prodotto	17
4	Specifica dei test	17
4.1	Test di accettazione	17
4.2	Test di sistema	20
4.3	Test di integrazione	23
4.4	Test di unità	24
	Appendici	28
A	Copertura dei requisiti	28
A.1	Resoconto copertura dei requisiti	31
B	Resoconto delle attività di verifica	33
B.1	Revisioni	33
B.1.1	Revisione dei Requisiti	33
B.1.2	Revisione di Progettazione	34
B.1.2.1	Consolidamento dei requisiti	34
B.1.2.2	Progettazione architetturale	36
B.1.3	Revisione di Qualifica	37
B.1.3.1	Progettazione di dettaglio e codifica	37
B.2	Misurazioni	39
B.2.1	Schedule Variance	39
B.2.2	Budget Variance	39
B.2.3	Numero rischi non previsti	39
B.2.4	Indisponibilità dei servizi esterni	40
B.2.5	Percentuale di test eseguiti	40
B.2.6	Percentuale test case passati	41
B.2.7	Percentuale test case falliti	41
B.2.8	Code coverage	42
B.2.9	Tempo medio necessario al team per risolvere un errore	42
B.2.10	Efficienza nella progettazione dei test	43
B.2.11	Percentuale di errori corretti	43
B.2.12	Media commit a settimana	44
B.2.13	Media build Travis a settimana	44
B.2.14	Percentuale build Travis superate	45
B.2.15	Gunning fog index	45
B.2.16	Indice di Gulpease	46
B.2.17	Numero di errori grammaticali	46
B.2.18	Functional Implementation Completeness	47
B.2.19	Average Functional Implementation Correctness	47
B.2.20	Tempo di risposta	48
B.2.21	Average Learning Time	48
B.2.22	Failure Density	49
B.2.23	Operazioni con gestione errori	49
B.2.24	Failure Analysis	50
B.2.25	Comment Ratio	50

C	Specifica dei test	52
C.1	Test di accettazione	52
C.2	Test di sistema	63
C.3	Test di integrazione	86
C.4	Test di unità	87

Elenco delle tabelle

2	Riassunto delle metriche di processo	13
3	Riassunto delle metriche di prodotto	17
4	Riassunto test di accettazione	18
5	Riassunto test di sistema	21
6	Riassunto test di integrazione	23
7	Riassunto test di unità	27
8	Copertura requisiti funzionali	30
9	Maturità dei processi, Consolidamento	34
10	Maturità dei processi, Progettazione Architetturale	36
11	Maturità dei processi, Progettazione di dettaglio e codifica	37
12	Specifica test di accettazione	62
13	Specifica test di sistema	85
14	Specifica test di integrazione	86
15	Specifica test di unità	142

Elenco delle figure

1	Resoconto test di accettazione	19
2	Resoconto test di sistema	22
3	Resoconto test di integrazione	23
4	Resoconto test di unità	27
5	Resoconto requisiti funzionali obbligatori	31
6	Resoconto requisiti funzionali desiderabili	31
7	Resoconto requisiti funzionali opzionali	32
8	Grafico del metodo PDCA, fase di Analisi	34
9	Grafico del metodo PDCA, fase di Consolidamento dei requisiti	35
10	Grafico del metodo PDCA, fase di Progettazione architetturale	36
11	Grafico del metodo PDCA, fase di Progettazione di dettaglio e codifica	38
12	Misurazione Schedule Variance	39
13	Misurazione Budget Variance	39
14	Misurazione numero rischi non previsti	40
15	Misurazione indisponibilità dei servizi esterni	40
16	Misurazione percentuale di test eseguiti	41
17	Misurazione percentuale test case passati	41
18	Misurazione percentuale test case falliti	42
19	Misurazione code coverage	42
20	Misurazione tempo medio necessario per risolvere un errore	43
21	Misurazione efficienza nella progettazione dei test	43
22	Misurazione percentuale di errori corretti	44
23	Misurazione media commit a settimana	44
24	Misurazione media build Travis a settimana	45
25	Misurazione percentuale build Travis superate	45
26	Misurazione Gunning fog index	46
27	Misurazione indice di Gulpease	46
28	Misurazione numero di errori grammaticali	47
29	Misurazione Functional Implementation Completeness	47
30	Misurazione Average Functional Implementation Correctness	48
31	Misurazione tempo di risposta	48
32	Misurazione Average Learning Time	49
33	Misurazione Failure Density	49
34	Misurazione	50
35	Misurazione Failure Analysis	50
36	Misurazione Comment Ratio	51

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di esporre dettagliatamente le norme, le metodologie e gli standard che il gruppo 7DOS intende adottare per assicurare che ogni *prodotto_g*, di natura documentale o applicativa, aderisca ai vincoli di *qualità_g* stabiliti dal proponente. Per garantire il rispetto di tali vincoli si prevede un continuo *processo_g* di verifica delle attività svolte dal gruppo, al fine di individuare eventuali problematiche nel minor tempo possibile permettendo immediati interventi risolutivi.

1.2 Scopo del prodotto

Il prodotto da realizzare consiste in un *plug-in_g* per il software di monitoraggio *Grafana_g*, da sviluppare in linguaggio *JavaScript_g*. Il prodotto dovrà svolgere almeno le seguenti funzioni:

- Leggere la definizione di una *rete Bayesiana_g*, memorizzata in formato *JSON_g*;
- Associare dei nodi della rete Bayesiana ad un flusso di dati presente nel sistema di Grafana;
- Ricalcolare i valori delle probabilità della rete secondo regole temporali prestabilite;
- Derivare nuovi dati dai nodi della rete non collegati al flusso di dati, e fornirli al sistema di Grafana;
- Visualizzare i dati mediante il sistema di creazione di grafici e *dashboard_g* a disposizione.

L'azienda proponente prevede di utilizzare il prodotto per il monitoraggio di sistemi gestionali in Cloud; tuttavia, dato l'obiettivo di rendere il prodotto open-source, esso dovrà essere utilizzabile indipendentemente dal particolare sistema che si desidera monitorare.

1.3 Glossario

Per rendere la lettura del documento più semplice, chiara e comprensibile viene allegato il *Glossario v3.0.0* nel quale sono contenute le definizioni dei termini tecnici, dei vocaboli ambigui, degli acronimi e delle abbreviazioni. La presenza di un termine all'interno del Glossario è segnalata con una "g" posta come pedice (esempio: *Glossario_g*).

1.4 Maturità del documento

Il presente documento sarà soggetto ad incrementi futuri. Per questo motivo, non si pone l'obiettivo di risultare completo. Tutto ciò che riguarda la pianificazione degli incrementi, può essere trovato nel *Piano di Progetto v3.0.0* all'interno della quarta sezione.

1.5 Riferimenti

1.5.1 Normativi

- **Norme di Progetto:** *Norme di Progetto v3.0.0*;

- **Capitolato d'appalto C3:** G&B monitoraggio intelligente di processi DevOps
<https://www.math.unipd.it/~tullio/IS-1/2018/Progetto/C3.pdf>;
- **Verbali:** *Verbale del 2018-12-11, Verbale del 2019-03-18.*

1.5.2 Informativi

- **ISO/IEC 12207:**
https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf;
- **ISO/IEC 25010:**
[https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en](https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en;);
- **ISO/IEC 29119:**
<https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29119:-1:ed-1:v1:en>;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Processi Software:**
<https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L03.pdf>;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Verifica e validazione: introduzione:**
<https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L16.pdf>;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Verifica e validazione: analisi statica:**
<https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L17.pdf>;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Verifica e validazione: analisi dinamica:**
<https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L18.pdf>;
- **Grafana_g Code Styleguide:**
<http://docs.grafana.org/plugins/developing/code-styleguide/>;
- **Angular TypeScript_g Code Styleguide:**
<https://angular.io/guide/styleguide>;
- **Software Engineering - Ian Sommerville - 10th Edition**(Capitolo 24).

2 Qualità di processo

È impossibile creare prodotti di alta qualità se il proprio *way of working_g* è scadente: risulta quindi fondamentale che i processi attuati, *in primis*, garantiscano un elevato livello qualitativo. Il gruppo 7DOS ha deciso di adottare la normativa *ISO/IEC 15504_g*, anche nota come *SPICE_g*, e di seguire il principio di miglioramento continuo (*PDCA_g*).

Per il controllo della qualità di processo il gruppo 7DOS utilizzerà l'approccio a maturità di processo, in quanto previsto dalle buone pratiche di *management_g* e più adatto ad un gruppo inesperto.

2.1 Pianificazione e controllo

Questo processo ha come scopo la pianificazione dello sviluppo del progetto, andando a definire suddivisione delle attività da svolgere, organizzazione oraria del lavoro, pianificazione e controllo dei costi e standard di supporto e accertandosi che il team abbia padronanza delle tecnologie utilizzate.

2.1.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Pianificazione:** organizzazione oraria del lavoro, prestando attenzione nell'assegnazione dei compiti ai vari membri del gruppo e attenendosi ai costi preventivati;
- **Budget:** controllo e verifica della pianificazione monetaria cercando di attenersi ai costi preventivati;
- **Standard:** attenersi a standard ben precisi, affinché il gruppo abbia sempre una linea guida di supporto;
- **Preparazione del personale:** assicurarsi che ciascun membro del personale abbia familiarità con le tecnologie richieste in ogni *task_g*, in modo tale da garantire produttività dell'ambiente lavorativo.

2.1.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Schedule Variance* (*SV_g*);
- *Budget Variance* (*BV_g*).

2.2 Gestione dei rischi

Questo processo ha come scopo il riconoscimento dei rischi a cui il progetto può incorrere. Il gruppo 7DOS mira, così facendo, a ridurre il più possibile il verificarsi di essi.

2.2.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Identificazione nuovi rischi:** ad ogni fase del progetto il gruppo identificherà i possibili nuovi rischi;
- **Analisi dei rischi:** i rischi individuati verranno analizzati in modo da poter fornire procedure automatiche per prevenire che uno di essi si verifichi.

2.2.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Numero rischi non previsti;*
- *Indisponibilità servizi esterni.*

2.3 Gestione dei test

Questo processo ha come scopo la definizione di misurazioni sull'esecuzione dell'analisi dinamica, in modo tale da garantire una gestione efficace di essa.

2.3.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Efficienza dei test:** i test devono essere progettati per individuare il maggior numero di difetti possibili. In questo modo è possibile realizzare un prodotto di qualità che soddisfa i requisiti specificati;
- **Risoluzione tempestiva dei problemi riscontrati:** le problematiche evidenziate dai test devono essere risolte nel minor tempo possibile dal gruppo;
- **Eseguire tutti i test necessari:** il sistema si evolve. Molti dei difetti precedentemente segnalati vengono corretti. Ogni volta che viene risolto un errore o è stata aggiunta una nuova funzionalità, è necessario eseguire tutti i test per assicurarsi che il nuovo software modificato abbia effettivamente corretto gli errori noti e non abbia introdotto nuovi errori.

2.3.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Percentuale test case passati;*
- *Percentuale test case falliti;*
- *Tempo medio necessario al team per risolvere un errore;*
- *Efficienza nella progettazione dei test;*
- *Percentuale di errori corretti;*

- *Percentuale dei test eseguiti;*
- *Code coverage.*

2.4 Versionamento e build

Questo processo ha come scopo la quantificazione e il controllo dei *Commit_g*, in modo tale da verificare l'effettivo sviluppo di versioni sempre più complete e stabili del prodotto. Inoltre, lo strumento di integrazione continua *Travis CI_g* permetterà di eseguire le *build_g* in modo automatico, permettendo una verifica della correttezza del codice rispetto alle norme.

2.4.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Commit frequenti:** ogni membro del gruppo deve garantire una buona frequenza di commit, affinché il prodotto sia sempre aggiornato con le ultime modifiche;
- **Modifiche di piccole dimensioni:** ogni commit deve comportare una quantità di modifiche limitata. In questo modo, ogni versione del prodotto è monitorabile e verificabile;
- **Buona riuscita delle build:** ogni membro deve accertarsi che ogni commit comporti una build positiva, in modo tale evitare la diffusione di errori.

2.4.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Media commit a settimana;*
- *Media build Travis a settimana;*
- *Percentuale build Travis superate.*

2.5 Riassunto delle metriche di processo

Nome Metrica	Range accettabile	Range ottimale
Schedule Variance	≥ -5 giorni	≥ 0 giorni
Budget Variance	$\geq -10\%$	$\geq 0\%$
Numero rischi non previsti	≤ 3	0
Indisponibilità servizi esterni	≥ -5	≥ 0
Percentuale test case passati	100%	100%
Percentuale test case falliti	0%	0%
Tempo medio necessario al team per risolvere un errore	0h-4h	0h-2h
Efficienza nella progettazione dei test	0.5h-3h	0.5h-2h
Percentuale dei difetti corretti	90%-100%	95%-100%
Percentuale dei test eseguiti	85%-100%	95%-100%
Code coverage	80%-100%	90%-100%
Media commit a settimana	≥ 20	≥ 30
Media build Travis a settimana	≥ 20	≥ 30
Percentuale build Travis superate	70%-100%	80%-100%

Tabella 2: Riassunto delle metriche di processo

3 Qualità di prodotto

Per poter garantire che il prodotto realizzato sia di alta qualità, è necessario definire un modello per la valutazione di quest'ultima; il team 7DOS, per questo motivo, ha scelto di adottare il modello di qualità delineato nello standard *ISO/IEC 25010_g*, anche noto come *SQuaRE_g*.

Tale modello comprende 8 caratteristiche (ciascuna divisa in sotto-caratteristiche, per un totale di 31) che vanno prese in considerazione durante lo sviluppo del progetto per garantire un'elevata qualità complessiva del prodotto finale.

Per praticità e rilevanza ai fini del prodotto, sono state selezionate 5 caratteristiche da considerare e per ciascuna sono state individuate le sotto-caratteristiche più rilevanti al progetto, da perseguire come obiettivi prioritari. In particolare, sono state scartate: *Compatibility_g*, in quanto andando a realizzare un plug-in (di natura integrato in un sistema preesistente) è stata giudicata superflua; *Security_g*, in quanto il plug-in non dovrà gestire autenticazione o raccolta di dati; ed infine *Portability_g* in quanto essendo il prodotto un plug-in per un determinato sistema, non è rilevante la sua portabilità ad altri ambienti.

3.1 Qualità dei documenti

I documenti prodotti devono possedere caratteristiche consistenti. In particolare, essi devono essere leggibili, comprensibili e corretti a livello ortografico e sintattico.

3.1.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Leggibilità:** i documenti prodotti dovranno essere leggibili e comprensibili da persone con almeno una licenza di istruzione media;
- **Correttezza ortografica:** i documenti prodotti dovranno essere privi di errori ortografici.

3.1.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Gunning fog index*;
- *Indice di Gulpease*;
- *Numero di errori grammaticali*.

3.2 Qualità del software

3.2.1 Functional Suitability

Caratteristica che esprime il grado di soddisfacimento dei requisiti espliciti ed impliciti da parte di un prodotto o servizio.

3.2.1.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Functional Completeness:** l'insieme delle funzioni rispettano le aspettative;
- **Functional Correctness:** i risultati ottenuti sono corretti e presentano livelli di precisione desiderati.

3.2.1.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Functional Implementation Completeness;*
- *Average Functional Implementation Correctness.*

3.2.2 Performance Efficiency

Caratteristica che esprime le prestazioni relative al sistema come quantità di risorse utilizzate per eseguire una determinata funzionalità.

3.2.2.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Time Behaviour:** tempi di risposta ed elaborazione assumono valori adeguati;
- **Resource Utilization:** le risorse disponibili utilizzate durante l'esecuzione di una funzionalità sono adeguate rispetto alla funzionalità stessa.

3.2.2.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Tempo di risposta.*

3.2.3 Usability

Caratteristica che esprime il grado con cui un prodotto o sistema può essere usato da un determinato utente.

3.2.3.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Learnability:** il prodotto è comprensibile dall'utente e il tempo per conoscere a pieno le sue funzionalità è accettabile;
- **Operability:** le funzionalità offerte devono essere coerenti con ciò che l'utente si aspetta;

- **User Error Protection:** il sistema presenta procedure per proteggere gli utenti dal commettere errori.

3.2.3.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Average Learning Time.*

3.2.4 Reliability

Caratteristica che esprime il grado con cui un prodotto esegue correttamente determinate funzioni mentre è in uso.

3.2.4.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Maturity:** evitare il più possibile malfunzionamenti in caso di *fault_g*;
- **Fault Tolerance:** se si verificano errori, vengono attivate procedure di gestione dell'errore in modo da non influenzare le prestazioni.

3.2.4.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Failure Density*;
- *Operazioni con gestione errori.*

3.2.5 Maintainability

Caratteristica che esprime il grado di efficacia ed efficienza con cui il prodotto può essere modificato, tramite azioni di correzione, o aggiornato, tramite azioni di miglioramento.

3.2.5.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Modularity:** il sistema presenta componenti tali che una modifica ad uno di essi ha il minimo impatto su tutte le altre componenti;
- **Analyzability:** deve essere possibile analizzare l'impatto nel sistema di uno specifico cambiamento ad una o più delle sue parti, ai fini di risalire rapidamente eventuali cause che hanno generato un malfunzionamento;
- **Modifiability:** deve essere possibile eseguire cambiamenti di componenti originali senza introdurre possibili errori;

- **Testability:** deve essere possibile testare il sistema per validare le varie modifiche e per valutare la qualità del sistema.

3.2.5.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v3.0.0*, sono le seguenti:

- *Failure Analysis*;
- *Comment Ratio*.

3.3 Riassunto delle metriche di prodotto

Nome Metrica	Intervallo limite	Range accettabile	Range ottimale
Gunning fog index	/	12-15	0-12
Indice di Gulpease	0%-100%	40%-100%	60%-100%
Numero di errori grammaticali	/	0	0
Functional Implementation Completeness	0%-100%	75%-100%	100%
Average Functional Implementation Correctness	0%-100%	80%-100%	95%-100%
Tempo di risposta	/	0-8 sec	0-3 sec
Average Learning Time	/	0-30	0-15
Failure Density	0%-100%	0%-10%	0%
Operazioni con gestione errori	0%-100%	80%-100%	95%-100%
Failure Analysis	0%-100%	60%-100%	80%-100%
Comment Ratio	0%-100%	60%-100%	80%-100%

Tabella 3: Riassunto delle metriche di prodotto

4 Specifica dei test

4.1 Test di accettazione

Test	Requisito	Stato
TA-1	R0F1	NI
TA-1.1	R1F1.1	NI
TA-2	R0F2	NI
TA-2.1	R0F2.1	NI
TA-2.2	R0F2.2	NI
TA-2.3	R1F2.3	NI
TA-3	R0F3	NI
TA-3.1	R1F3.1	NI
TA-4	R0F4	NI
TA-4.1	R1F4.1	NI
TA-5	R0F5	NI
TA-5.1	R1F5.1	NI
TA-5.2	R1F5.2	NI
TA-5.3	R1F5.3	NI
TA-5.4	R1F5.4	NI
TA-5	R1F5.5	NI
TA-5.6	R1F5.6	NI
TA-5.7	R1F5.7	NI
TA-6	R1F6	NI
TA-6.1	R1F6.1	NI
TA-6.2	R1F6.2	NI
TA-7	R1F7	NI
TA-7.1	R1F7.1	NI
TA-7.2	R1F7.2	NI
TA-7.3	R1F7.4	NI
TA-7.4	R1F7.5	NI
TA-7.5	R1F7.6	NI
TA-7.6	R1F7.7	NI
TA-8	R2F8	NI
TA-9	R2F9	NI
TA-10	R2F10	NI
TA-10.1	R2F10.1	NI
TA-10.2	R2F10.2	NI
TA-10.4	R2F10.4	NI
TA-10.5	R2F10.5	NI
TA-10.6	R2F10.6	NI

Tabella 4: Riassunto test di accettazione

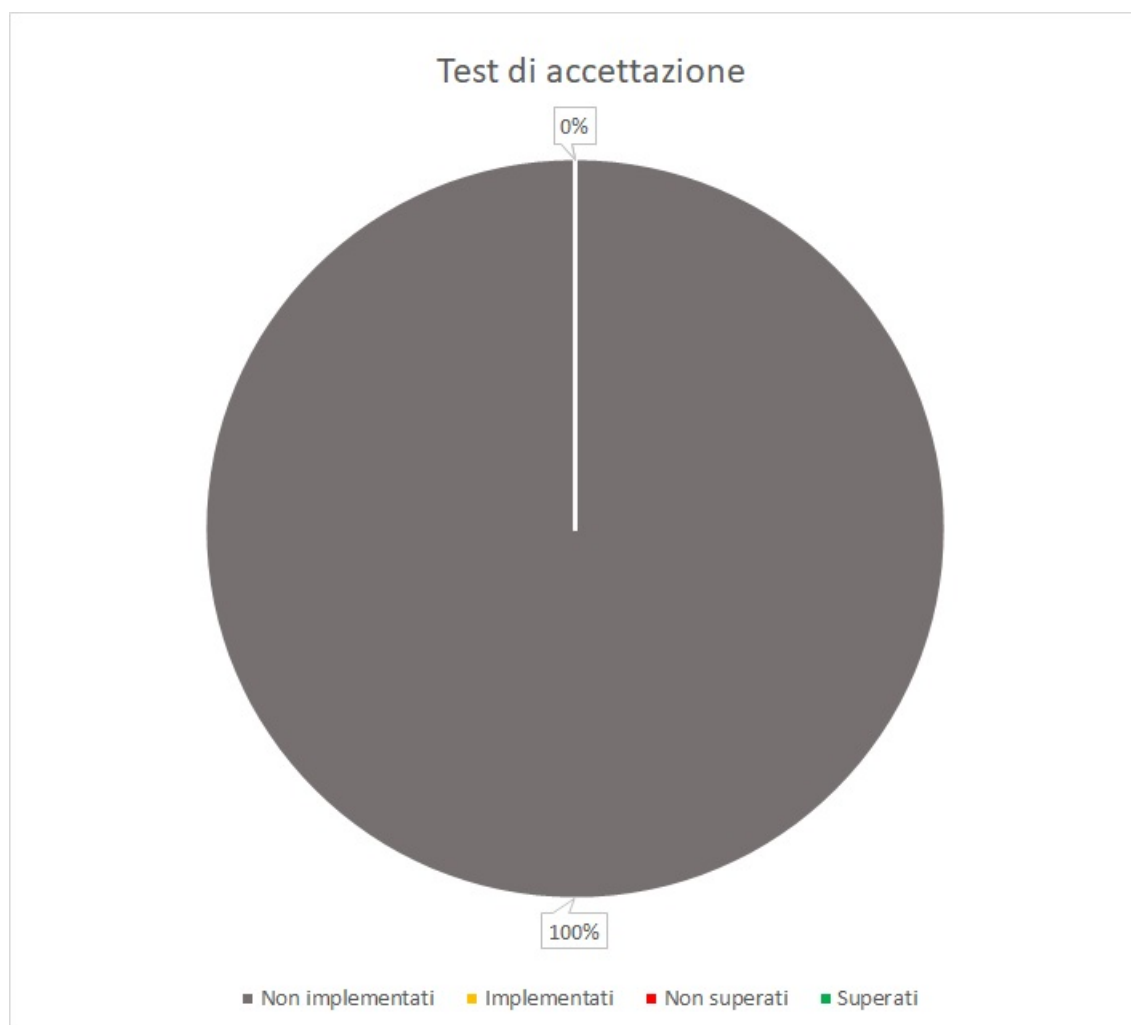


Figura 1: Resoconto test di accettazione

4.2 Test di sistema

Test	Requisito	Stato
TS-1	R0F1	S
TS-1.1	R1F1.1	S
TS-2	R0F2	S
TS-2.1	R0F2.1	S
TS-2.2	R0F2.2	S
TS-2.3	R1F2.3	S
TS-3	R0F3	S
TS-3.1	R1F3.1	S
TS-4	R0F4	S
TS-4.1	R1F4.1	S
TS-5	R0F5	S
TS-5.1	R1F5.1	S
TS-5.2	R1F5.2	S
TS-5.3	R1F5.3	S
TS-5.4	R1F5.4	S
TS-5.5	R1F5.5	S
TS-5.6	R1F5.6	S
TS-5.6.1	R1F5.6.1	S
TS-5.6.2	R1F5.6.2	S
TS-5.6.3	R1F5.6.3	S
TS-5.7	R1F5.7	S
TS-5.7.1	R1F5.7.1	S
TS-6	R1F6	S
TS-6.1	R1F6.1	S
TS-6.1.1	R1F6.1.1	S
TS-6.1.2	R1F6.1.2	S
TS-6.1.3	R1F6.1.3	S
TS-6.2	R1F6.2	S
TS-7	R1F7	NI
TS-7.1	R1F7.1	NI
TS-7.1.1	R1F7.1.1	NI
TS-7.1.2	R1F7.1.2	NI
TS-7.1.3	R1F7.1.3	NI
TS-7.1.4	R1F7.1.4	NI
TS-7.1.4.1	R1F7.1.4.1	NI
TS-7.1.4.2	R1F7.1.4.2	NI
TS-7.1.4.3	R1F7.1.4.3	NI
TS-7.2	R1F7.2	NI
TS-7.2.1	R1F7.2.1	NI
TS-7.2.2	R1F7.2.2	NI

Test	Requisito	Stato
TS-7.2.2.1	R1F7.2.2.1	NI
TS-7.2.2.2	R1F7.2.2.2	NI
TS-7.2.2.3	R1F7.2.2.3	NI
TS-7.2.2.4	R1F7.2.2.4	NI
TS-7.3	R1F7.3	NI
TS-7.4	R1F7.4	NI
TS-7.4.1	R1F7.4.1	NI
TS-7.4.2	R1F7.4.2	NI
TS-7.5	R1F7.5	NI
TS-7.6	R1F7.6	NI
TS-7.6.1	R1F7.6.1	NI
TS-7.6.2	R1F7.6.2	NI
TS-7.7	R1F7.7	NI
TS-7.7.1	R1F7.7.1	NI
TS-7.7.2	R1F7.7.2	NI
TS-7.7.3	R1F7.7.3	NI
TS-7.7.4	R1F7.7.4	NI
TS-8	R2F8	NI
TS-9	R2F9	NI
TS-10	R2F10	S
TS-10.1	R2F10.1	S
TS-10.2	R2F10.2	S
TS-10.3	R2F10.3	S
TS-10.3.1	R2F10.3.1	S
TS-10.3.2	R2F10.3.2	S
TS-10.3.3	R2F10.3.3	S
TS-10.3.4	R2F10.3.4	S
TS-10.3.5	R2F10.3.5	S
TS-10.4	R2F10.4	S
TS-10.4.1	R2F10.4.1	S
TS-10.4.2	R2F10.4.2	S
TS-10.4.3	R2F10.4.3	S
TS-10.4.3.1	R2F10.4.3.1	S
TS-10.4.3.2	R2F10.4.3.2	S
TS-10.4.3.3	R2F10.4.3.3	S
TS-10.5	R2F10.5	S
TS-10.6	R2F10.6	S

Tabella 5: Riassunto test di sistema

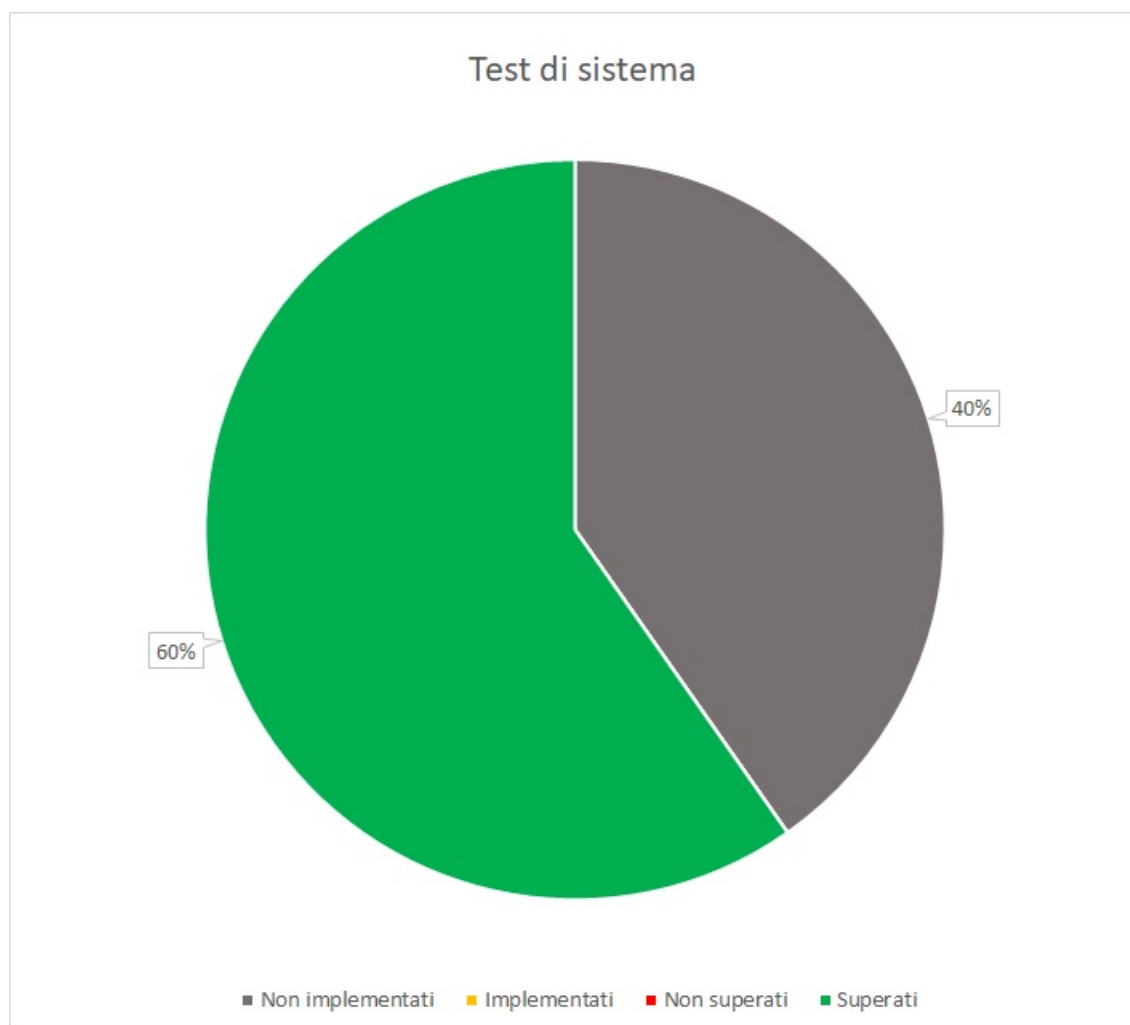


Figura 2: Resoconto test di sistema

4.3 Test di integrazione

Test	Componente	Stato
TI-1	Client - Grafana.	S
TI-2	Grafana - <i>InfluxDB_g</i>	S
TI-3	Grafana - JsBayes	S
TI-4	Grafana - Raintank	S

Tabella 6: Riassunto test di integrazione



Figura 3: Resoconto test di integrazione

4.4 Test di unità

Test	Stato
TU-1	S
TU-2	S
TU-3	S
TU-4	S
TU-5	S
TU-6	S
TU-7	S
TU-8	S
TU-9	S
TU-10	S
TU-11	S
TU-12	S
TU-13	S
TU-14	S
TU-15	S
TU-16	S
TU-17	S
TU-18	S
TU-19	S
TU-20	S
TU-21	S
TU-22	S
TU-23	S
TU-24	S
TU-25	S
TU-26	S
TU-27	S
TU-28	S
TU-29	S
TU-30	S
TU-31	S
TU-32	S
TU-33	S
TU-34	S
TU-35	S
TU-36	S
TU-37	S
TU-38	S
TU-39	S
TU-40	S

Test	Stato
TU-41	S
TU-42	S
TU-43	S
TU-44	S
TU-45	S
TU-46	S
TU-47	S
TU-48	S
TU-49	S
TU-50	S
TU-51	S
TU-52	S
TU-53	S
TU-54	S
TU-55	S
TU-56	S
TU-57	S
TU-58	S
TU-59	S
TU-60	S
TU-61	S
TU-62	S
TU-63	S
TU-64	S
TU-65	S
TU-66	N.I.
TU-67	S
TU-68	S
TU-69	S
TU-70	S
TU-71	S
TU-72	S
TU-73	S
TU-74	S
TU-75	S
TU-76	S
TU-77	S
TU-78	S
TU-79	S
TU-80	S
TU-81	S
TU-82	S

Test	Stato
TU-83	S
TU-84	S
TU-85	S
TU-86	S
TU-87	S
TU-88	S
TU-89	S
TU-90	S
TU-91	S
TU-92	S
TU-93	S
TU-94	S
TU-95	S
TU-96	S
TU-97	S
TU-98	S
TU-99	S
TU-100	S
TU-101	S
TU-102	S
TU-103	S
TU-104	S
TU-105	S
TU-106	S
TU-107	S
TU-108	S
TU-109	S
TU-110	S
TU-111	S
TU-112	S
TU-113	S
TU-114	S
TU-115	S
TU-116	S
TU-117	S
TU-118	S
TU-119	S
TU-120	S
TU-121	S
TU-122	S
TU-123	S
TU-124	S

Test	Stato
TU-125	S
TU-126	S
TU-127	S
TU-128	N.I.
TU-129	S
TU-130	S
TU-131	S
TU-132	S
TU-133	S
TU-134	S
TU-133	S
TU-135	S
TU-136	S
TU-137	S
TU-138	S
TU-139	S
TU-140	S
TU-141	S

Tabella 7: Riassunto test di unità

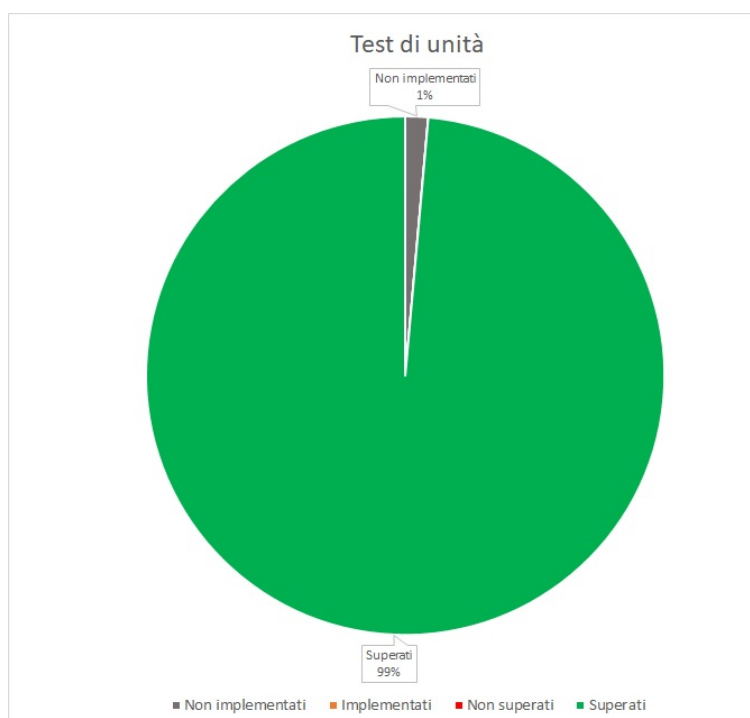


Figura 4: Resoconto test di unità

A Copertura dei requisiti

Requisito	Stato
R0F1	Implementato
R1F1.1	Implementato
R0F2	Implementato
R0F2.1	Implementato
R0F2.2	Implementato
R1F2.3	Implementato
R0F3	Implementato
R1F3.1	Implementato
R0F4	Implementato
R1F4.1	Implementato
R0F5	Implementato
R1F5.1	Implementato
R1F5.2	Implementato
R1F5.3	Implementato
R1F5.4	Implementato
R1F5.5	Implementato
R1F5.6	Implementato
R1F5.6.1	Implementato
R1F5.6.2	Implementato
R1F5.6.3	Implementato
R1F5.7	Implementato
R1F5.7.1	Implementato
R1F6	Implementato
R1F6.1	Implementato
R1F6.1.1	Implementato
R1F6.1.2	Implementato
R1F6.1.3	Implementato

Requisito	Stato
R1F6.2	Implementato
R1F7	Non implementato
R1F7.1	Non implementato
R1F7.1.1	Non implementato
R1F7.1.2	Non implementato
R1F7.1.3	Non implementato
R1F7.1.4	Non implementato
R1F7.1.4.1	Non implementato
R1F7.1.4.2	Non implementato
R1F7.1.4.3	Non implementato
R1F7.2	Non implementato
R1F7.2.1	Non implementato
R1F7.2.2	Non implementato
R1F7.2.2.1	Non implementato
R1F7.2.2.2	Non implementato
R1F7.2.2.3	Non implementato
R1F7.2.2.3.1	Non implementato
R1F7.2.2.3.2	Non implementato
R1F7.2.2.4	Non implementato
R1F7.3	Non implementato
R1F7.4	Non implementato
R1F7.4.1	Non implementato
R1F7.4.2	Non implementato
R1F7.5	Non implementato
R1F7.6	Non implementato
R1F7.6.1	Non implementato
R1F7.6.2	Non implementato
R1F7.7	Non implementato

Requisito	Stato
R1F7.7.1	Non implementato
R1F7.7.2	Non implementato
R1F7.7.3	Non implementato
R1F7.7.4	Non implementato
R2F8	Non implementato
R2F9	Non implementato
R2F10	Implementato
R2F10.1	Implementato
R2F10.2	Implementato
R2F10.3	Implementato
R2F10.3.1	Implementato
R2F10.3.2	Implementato
R2F10.3.3	Implementato
R2F10.3.4	Implementato
R2F10.3.5	Implementato
R2F10.4	Implementato
R2F10.4.1	Implementato
R2F10.4.2	Implementato
R2F10.4.3	Implementato
R2F10.4.3.1	Implementato
R2F10.4.3.2	Implementato
R2F10.4.3.3	Implementato
R2F10.5	Implementato
R2F10.6	Implementato

Tabella 8: Copertura requisiti funzionali

A.1 Resoconto copertura dei requisiti

- Requisiti obbligatori

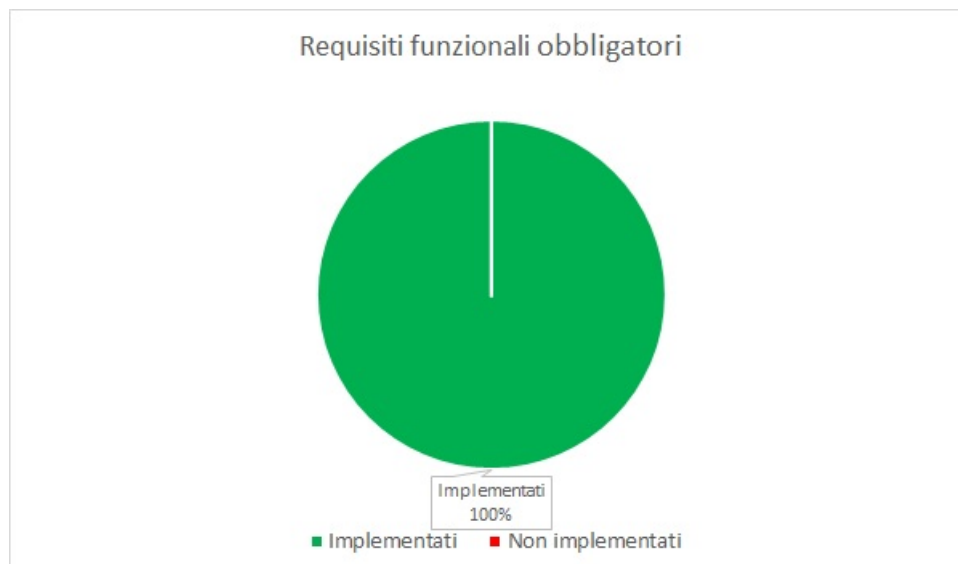


Figura 5: Resoconto requisiti funzionali obbligatori

- Requisiti desiderabili

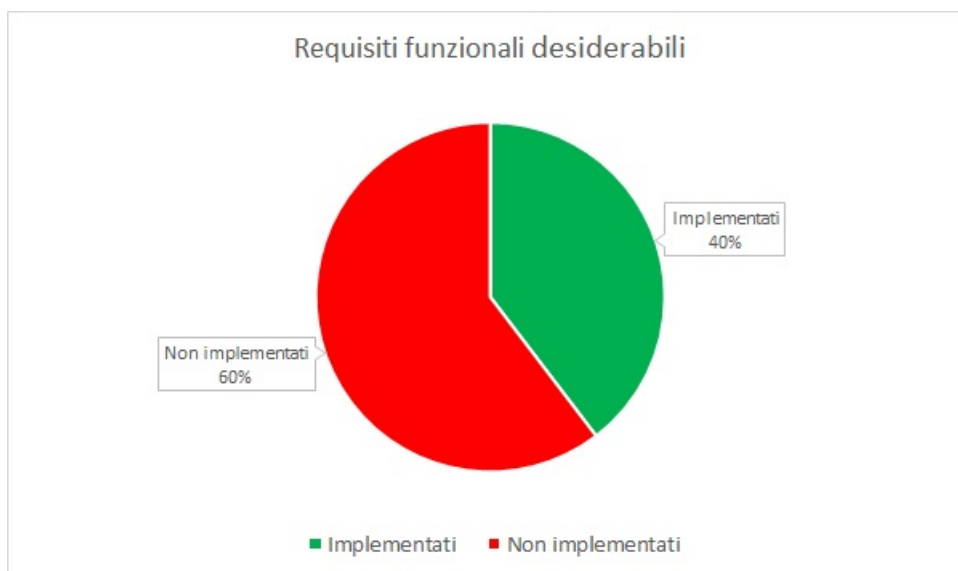


Figura 6: Resoconto requisiti funzionali desiderabili

- Requisiti opzionali

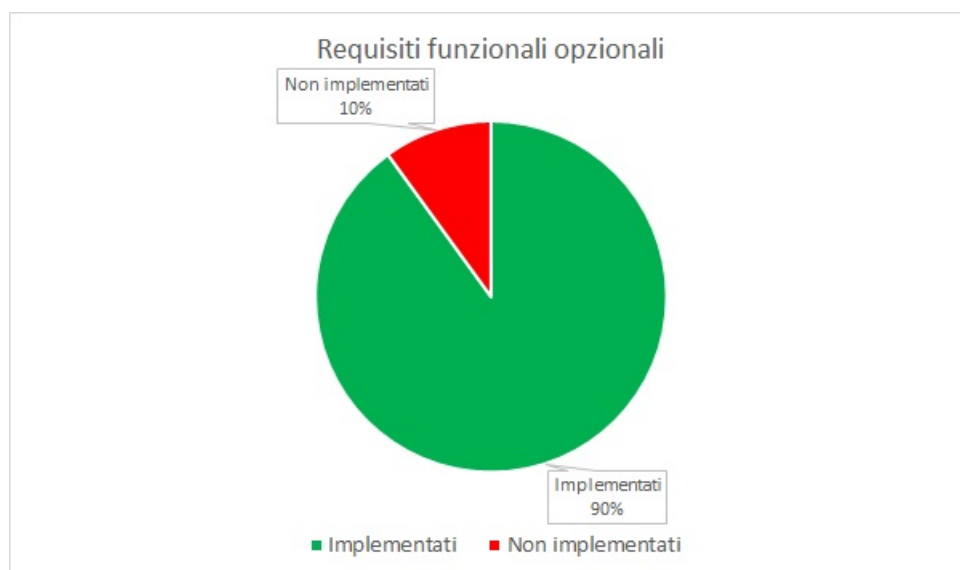


Figura 7: Resoconto requisiti funzionali opzionali

B Resoconto delle attività di verifica

B.1 Revisioni

Questa sezione riporta il resoconto delle attività di verifica svolte prima di ciascuna delle quattro revisioni stabilite dal committente (Revisione dei Requisiti, R. di Progettazione, R. di Qualifica e R. di Accettazione). Al termine di ogni revisione il committente segnalerà le problematiche riscontrate attraverso una valutazione globale dell'andamento del progetto ed una dettagliata per ciascun documento, permettendo al gruppo di eliminare problemi e criticità nel progetto per poi procedere su una base verificata e il più possibile corretta.

B.1.1 Revisione dei Requisiti

- *Norme di Progetto*: sono state effettuate le integrazioni richieste ed è stata aggiornata la struttura del documento in modo da rispettare gli stessi standard per ogni sezione; La sottosezione relativa alla verifica è stata aggiornata in modo da contenere le nuove metriche inserite e le metriche erroneamente inserite nel *Piano di Qualità v1.0.0*;
- *Analisi dei Requisiti*: sono state attuate le opportune modifiche suggerite. Alcuni casi d'uso sono stati leggermente rivisti ed è stato introdotto un caso d'uso più generico per ogni agglomerato di azioni con attitudini simili, modificando il caso d'uso principale;
- *Piano di Progetto*: sono state fatte alcune verifiche nell'uso di alcuni termini e standard come consigliato dal committente. Sono state aggiunte le opportune motivazioni e spiegazioni nelle scelte effettuate. La presentazione dei contenuti è stata rivista in modo da renderla più efficace;
- *Piano di Qualifica*: il documento è stato profondamente rivisto per struttura e contenuti secondo quanto specificato dal committente. Le specifiche degli standard utilizzati e le descrizioni delle metriche sono stati spostati in appendice alle *Norme di Progetto*. È stata aggiunta una sezione riguardante la pianificazione dei test, e le specifiche dei test sono state previste come futura aggiunta in appendice al documento. Infine, la strategia generale per la verifica ha assunto un ruolo centrale nella specifica degli obiettivi di qualità di processo e prodotto.

Il grafico rappresentante l'applicazione del metodo PDCA della fase di Analisi è:

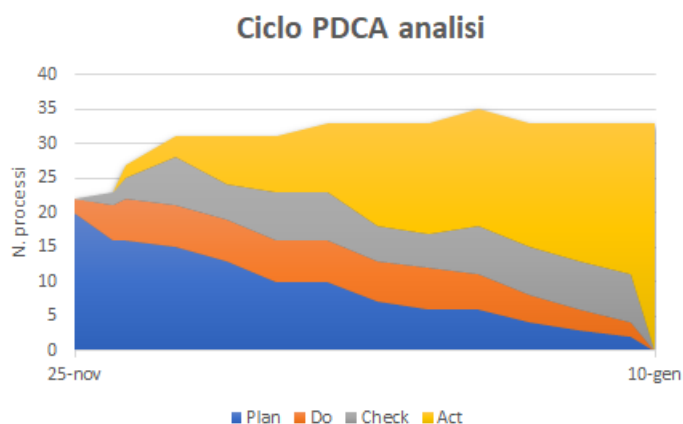


Figura 8: Grafico del metodo PDCA, fase di Analisi

Dal grafico possiamo estrapolare che:

- Alcuni dei processi pianificati hanno subito mutamenti, dovuti ad errori di pianificazione dati dalla poca esperienza del gruppo di lavoro;
- Il gruppo ha reso l'avanzamento dei processi omogeneo, nonostante alcuni rallentamenti dovuti alla sovrapposizione di impegni personali e universitari dei componenti del gruppo con la realizzazione del progetto. Nel complesso si vede come l'omogeneità è stata abbastanza rispettata.

B.1.2 Revisione di Progettazione

B.1.2.1 Consolidamento dei requisiti

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	3	Gestito con l'ausilio di nTask, ha permesso di rispettare le scadenze previste.
Gestione dei rischi	1	Non si sono verificate situazioni di rischio.
Gestione dei test	0	Verrà istanziato una volta presente la necessità di definire e condurre dei test.
Versionamento e build	0	Verrà istanziato una volta presente la necessità di versionare il prodotto software.

Tabella 9: Maturità dei processi e considerazioni

Il grafico rappresentante l'applicazione del metodo PDCA nella fase di Consolidamento dei requisiti è:

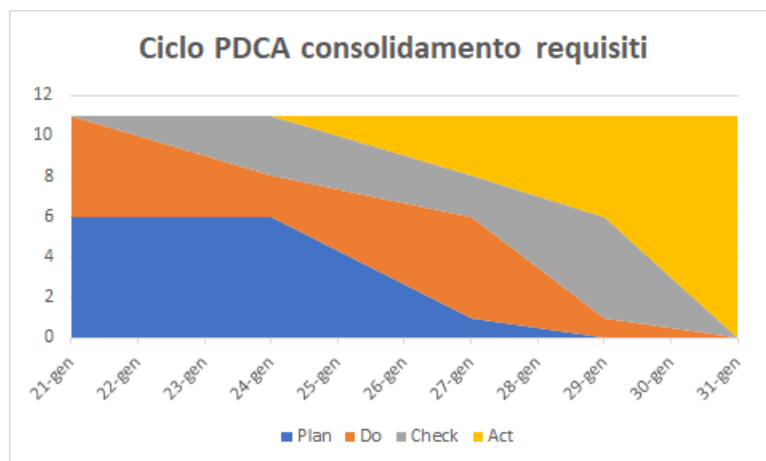


Figura 9: Grafico del metodo PDCA, fase di Consolidamento dei requisiti

Dal grafico possiamo estrapolare che:

- Trattandosi di un periodo relativamente breve (10 giorni), piccole variazioni nel numero di attività risultano in grossi cambiamenti grafici;
- Le attività in Plan sono state esaurite prima del termine del periodo, mentre quelle in Check e Act hanno avuto una permanenza maggiore; ciò è dovuto ad un numero di modifiche ai documenti relativamente basso ma di grande importanza e profondità (soprattutto per quanto riguarda il consolidamento del *Piano di Qualifica*), che quindi hanno richiesto periodi di Check e Act prolungati.

B.1.2.2 Progettazione architettuale

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	3	Ha permesso di rispettare le scadenze previste.
Gestione dei rischi	1	Si è dimostrato poco stabile al verificarsi di 3 situazioni di rischio (malattia di membri del gruppo), comportando una ridefinizione degli obiettivi per il periodo.
Gestione dei test	1	Sono stati pianificati i primi test di sistema, le cui specifiche tuttavia rimangono da definire.
Versionamento e build	1	Parzialmente automatizzato con l'utilizzo dello strumento <i>Travis_g</i> .

Tabella 10: Maturità dei processi e considerazioni

Il grafico rappresentante l'applicazione del metodo PDCA nella fase di Progettazione architettuale è:

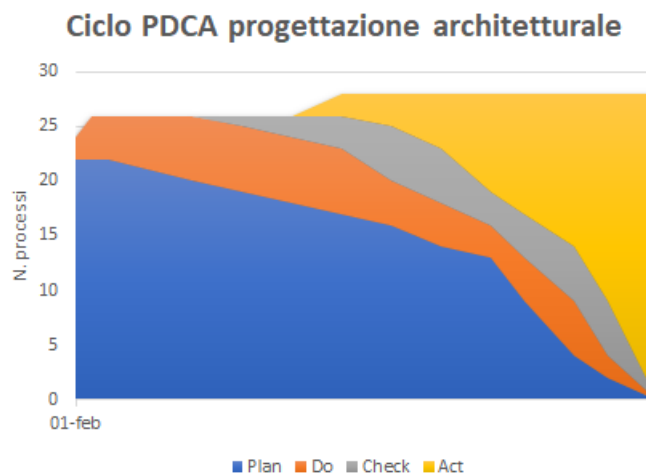


Figura 10: Grafico del metodo PDCA, fase di Progettazione architettuale

Dal grafico possiamo estrapolare che:

- Si nota uno stallo nell'esecuzione delle attività nella prima metà del periodo, dovuto principalmente, come previsto, alla sovrapposizione degli impegni universitari dei vari membri del gruppo. A questo si sono aggiunte le situazioni di rischio verificatesi, comportando ulteriori ritardi;

- Verso al fine del periodo si nota un rapido decremento delle attività in Do e Check, dovuto ad uno sforzo maggiorato per compensare i ritardi dovuti alle situazioni di rischio verificatesi.

B.1.3 Revisione di Qualifica

B.1.3.1 Progettazione di dettaglio e codifica

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	3	Ha permesso di rispettare le scadenze previste.
Gestione dei rischi	3	Si è dimostrato stabile al verificarsi di alcune situazioni di basso rischio, grazie alle istruzioni impartite dal <i>Responsabile di Progetto</i> .
Gestione dei test	3	Sono stati specificati ed implementati buona parte dei test. La copertura dei test di unità a registrato esiti molto positivi.
Versionamento e build	4	Automatizzato il processo tramite <i>Travis</i> . Ogni modifica apportata alla repository di GitHub scatuisce una serie di test sul codice, tenendo traccia dell'esito delle varie build e registrando valori utili per le nostre metriche, come il code coverage.

Tabella 11: Maturità dei processi e considerazioni

Il grafico rappresentante l'applicazione del metodo PDCA nella fase di Progettazione di dettaglio e codifica è:

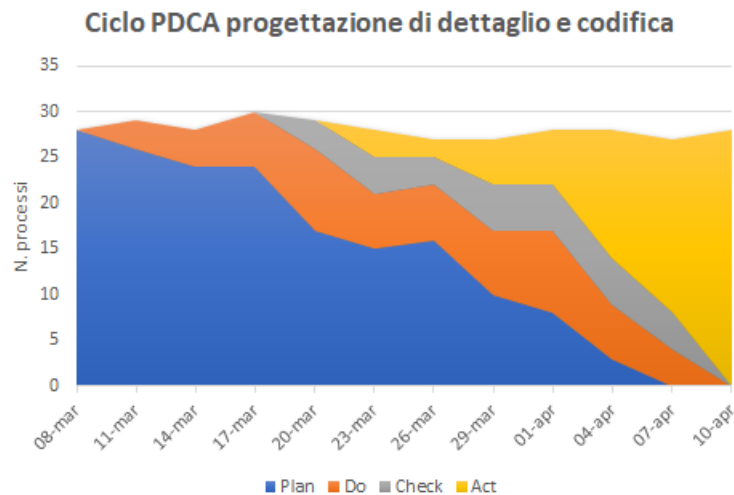


Figura 11: Grafico del metodo PDCA, fase di Progettazione di dettaglio e codifica

Dal grafico possiamo estrapolare che:

- Si nota una fase iniziale dovuta in maggior parte alla pianificazione delle attività finalizzate all'incremento dei documenti e allo sviluppo dei test;
- Il completamento delle attività pianificate comporta delle transizioni tra lo stato Do e lo stato Check che segue un andamento omogeneo, dovuto al fatto che non sono stati registrati scostamenti nelle tempistiche.

B.2 Misurazioni

B.2.1 Schedule Variance

Questa metrica è sempre rimasta in linea con il valore accettabile, ma non ha ancora raggiunto quello ottimale.

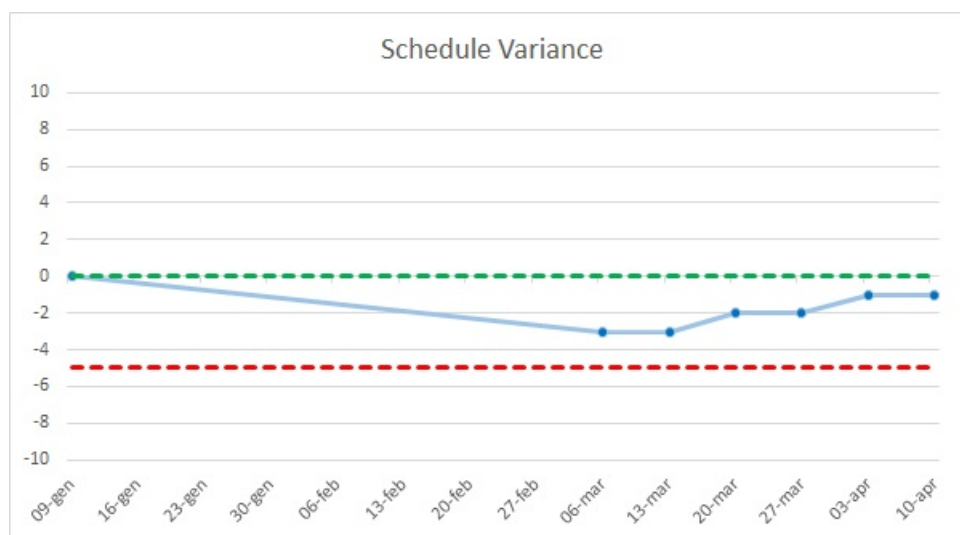


Figura 12: Misurazione Schedule Variance

B.2.2 Budget Variance

Questa metrica è sempre rimasta in linea con il valore ottimale.

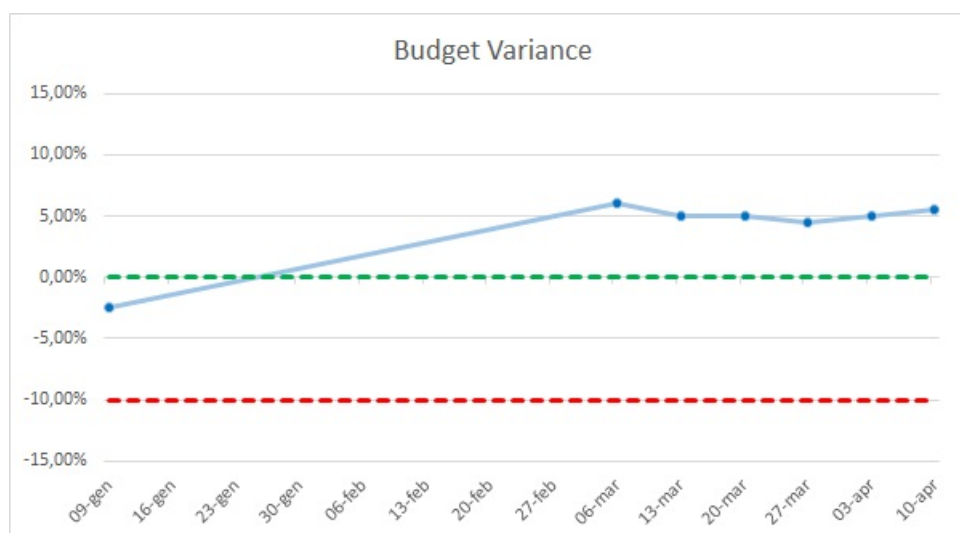


Figura 13: Misurazione Budget Variance

B.2.3 Numero rischi non previsti

Questa metrica è sempre rimasta in linea con il valore accettabile.

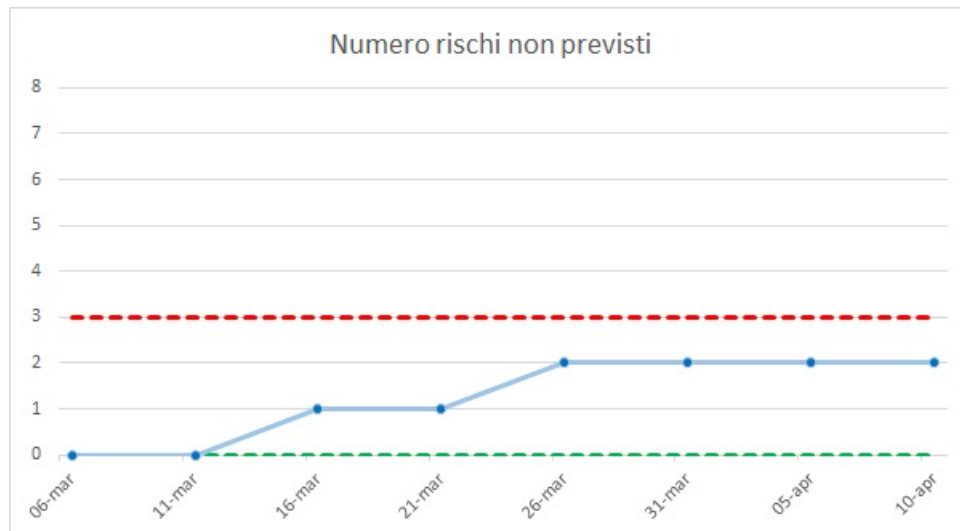


Figura 14: Misurazione numero rischi non previsti

B.2.4 Indisponibilità dei servizi esterni

Questa metrica è sempre rimasta in linea con il valore ottimale, tranne per il verificarsi dell'indisponibilità del servizio *TravisCI*.

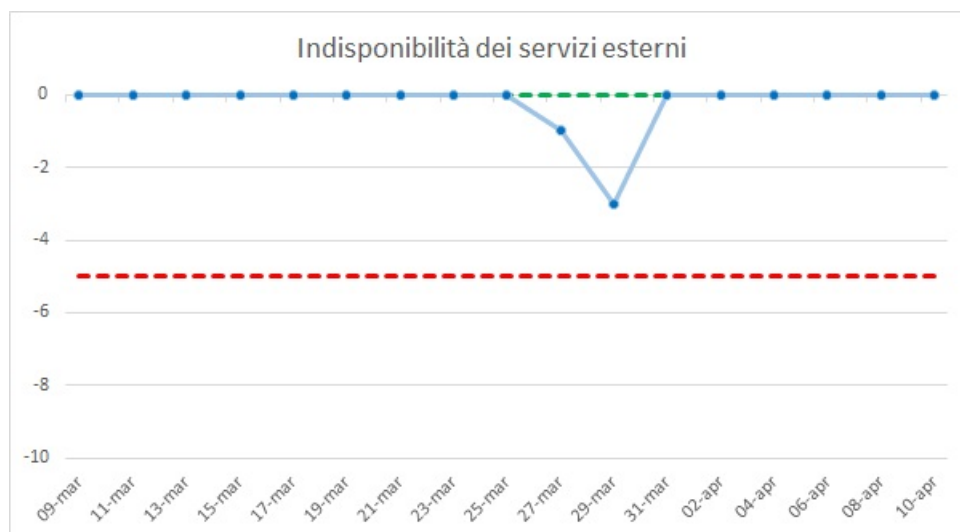


Figura 15: Misurazione indisponibilità dei servizi esterni

B.2.5 Percentuale di test eseguiti

Questa metrica inizialmente ha registrato dei valori non accettabili, ma con lo sviluppo dei test ha raggiunto un valore accettabile.

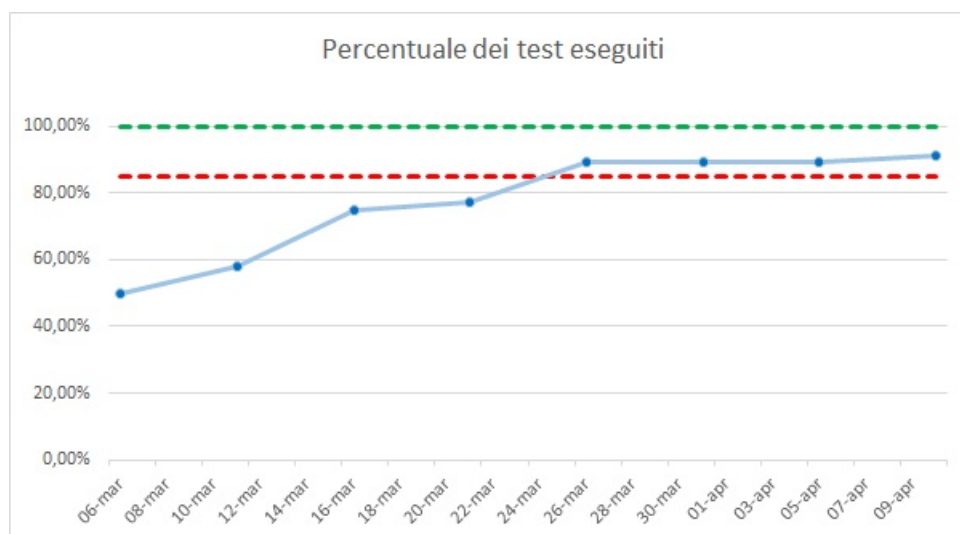


Figura 16: Misurazione percentuale di test eseguiti

B.2.6 Percentuale test case passati

Questa metrica ha quasi raggiunto un valore ottimale.

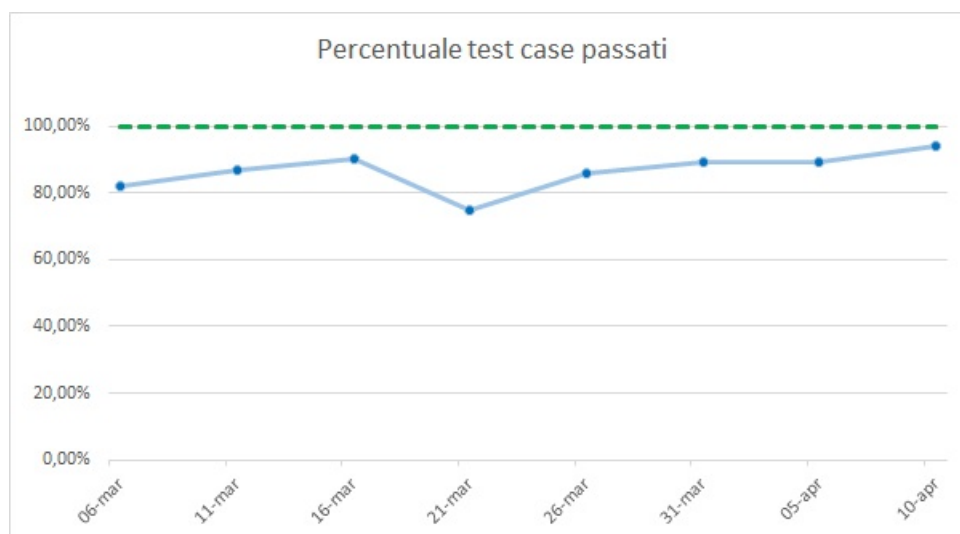


Figura 17: Misurazione percentuale test case passati

B.2.7 Percentuale test case falliti

Questa metrica ha quasi raggiunto un valore ottimale.

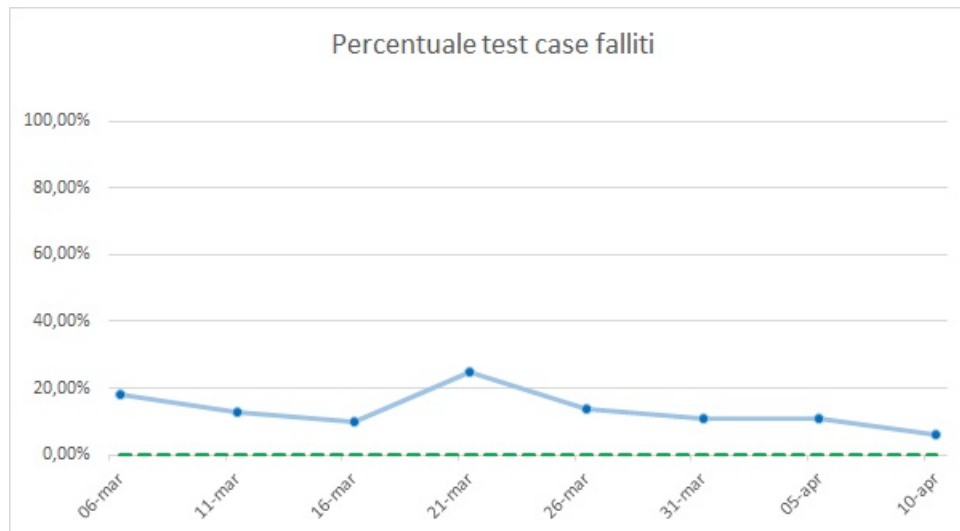


Figura 18: Misurazione percentuale test case falliti

B.2.8 Code coverage

Questa metrica inizialmente ha registrato dei valori non accettabili, ma con lo sviluppo dei test ha raggiunto un valore quasi ottimale.

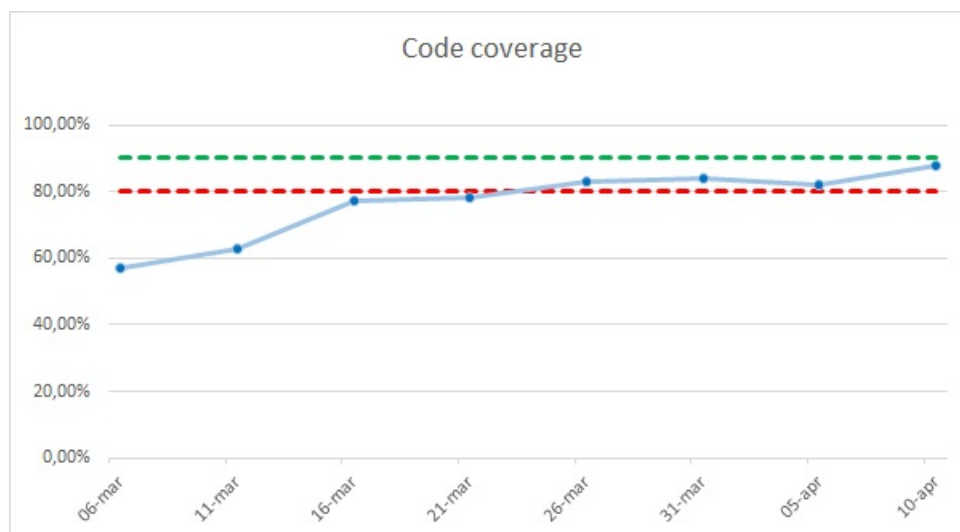


Figura 19: Misurazione code coverage

B.2.9 Tempo medio necessario al team per risolvere un errore

Questa metrica è sempre rimasta in linea con un valore ottimale, tranne in un'occasione causata da un errore consistente.

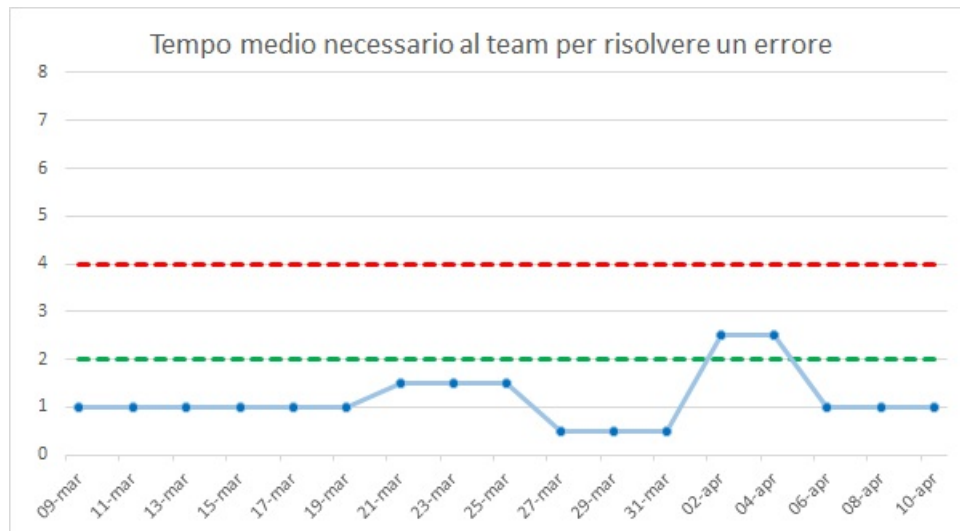


Figura 20: Misurazione tempo medio necessario per risolvere un errore

B.2.10 Efficienza nella progettazione dei test

Tranne per la fase di sviluppo dei test iniziale, ha sempre registrato un valore ottimale.

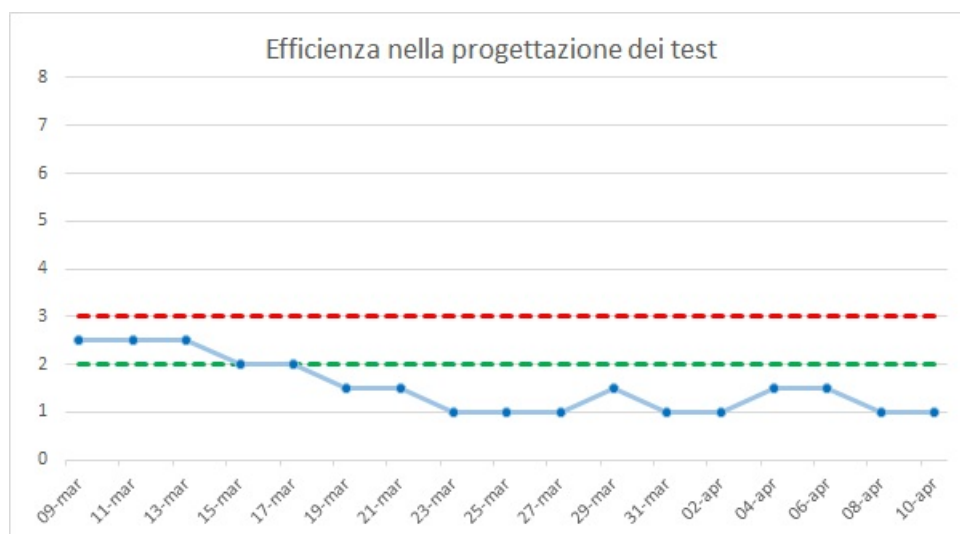


Figura 21: Misurazione efficienza nella progettazione dei test

B.2.11 Percentuale di errori corretti

La presenza di vari errori ha fatto registrare dei valori non accettabili, ma grazie al lavoro dei *Verificatori* questi sono stati corretti, raggiungendo un valore ottimale per questa metrica.

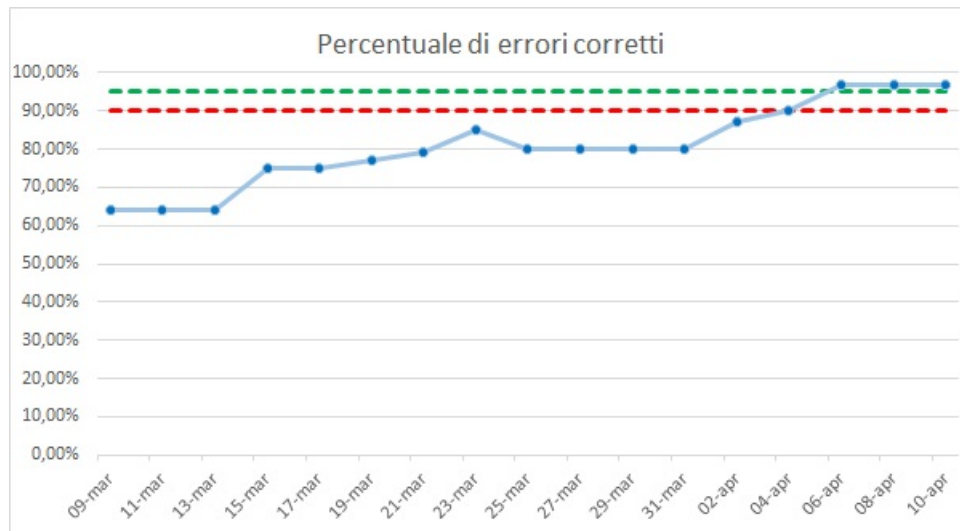


Figura 22: Misurazione percentuale di errori corretti

B.2.12 Media commit a settimana

Il lavoro costante del gruppo ha permesso di registrare dei valori accettabili, a volte anche ottimali, per questa metrica.

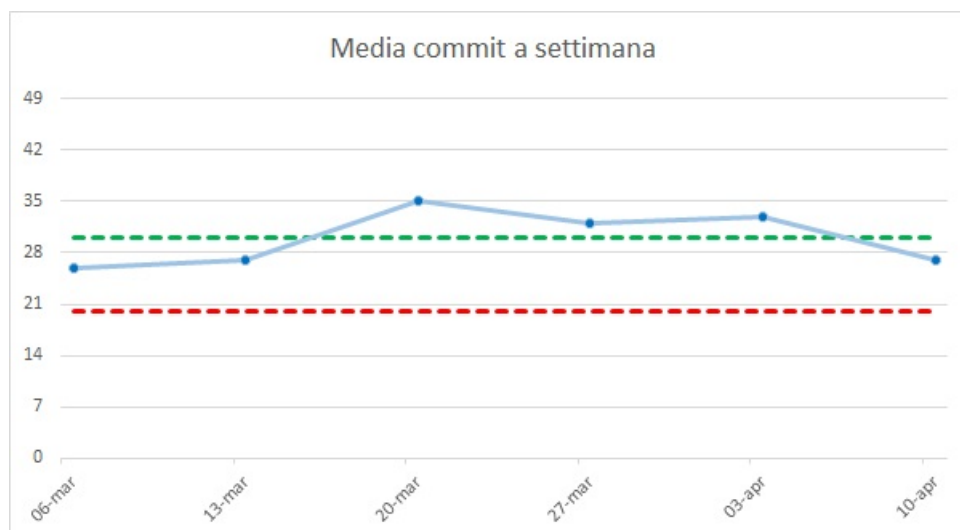


Figura 23: Misurazione media commit a settimana

B.2.13 Media build Travis a settimana

Il lavoro costante del gruppo ha permesso di registrare dei valori accettabili, a volte anche ottimali, per questa metrica.

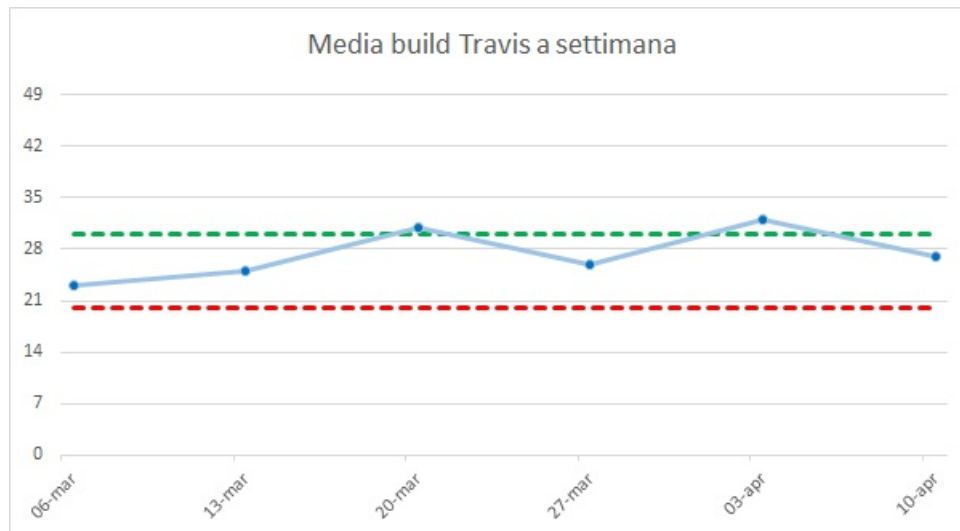


Figura 24: Misurazione media build Travis a settimana

B.2.14 Percentuale build Travis superate

La presenza di vari errori ha fatto registrare dei valori non accettabili inizialmente, ma una volta corrette queste criticità questa metrica ha raggiunto un valore accettabile.

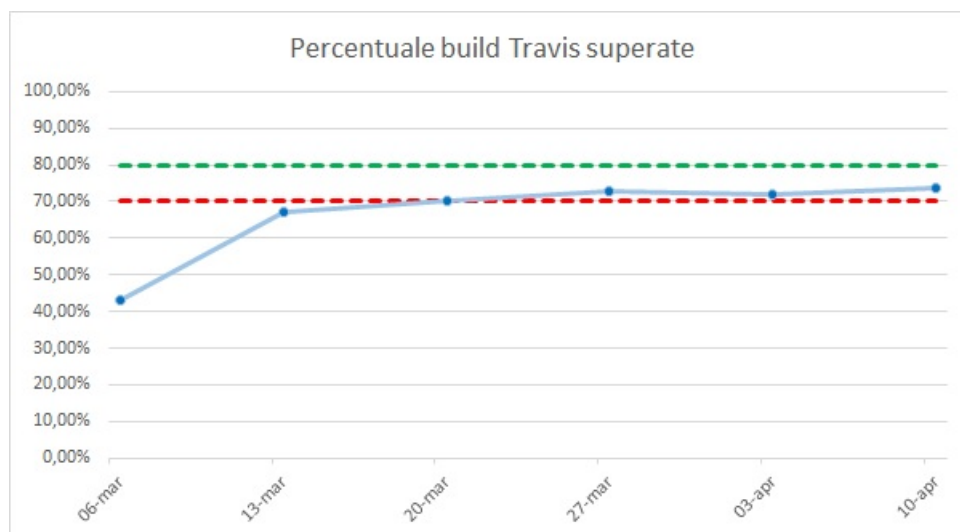


Figura 25: Misurazione percentuale build Travis superate

B.2.15 Gunning fog index

Tutti gli indici Gunning fox dei documenti rientrano nei vincoli dati. Per questo motivo i documenti redatti hanno raggiunto la leggibilità desiderata.

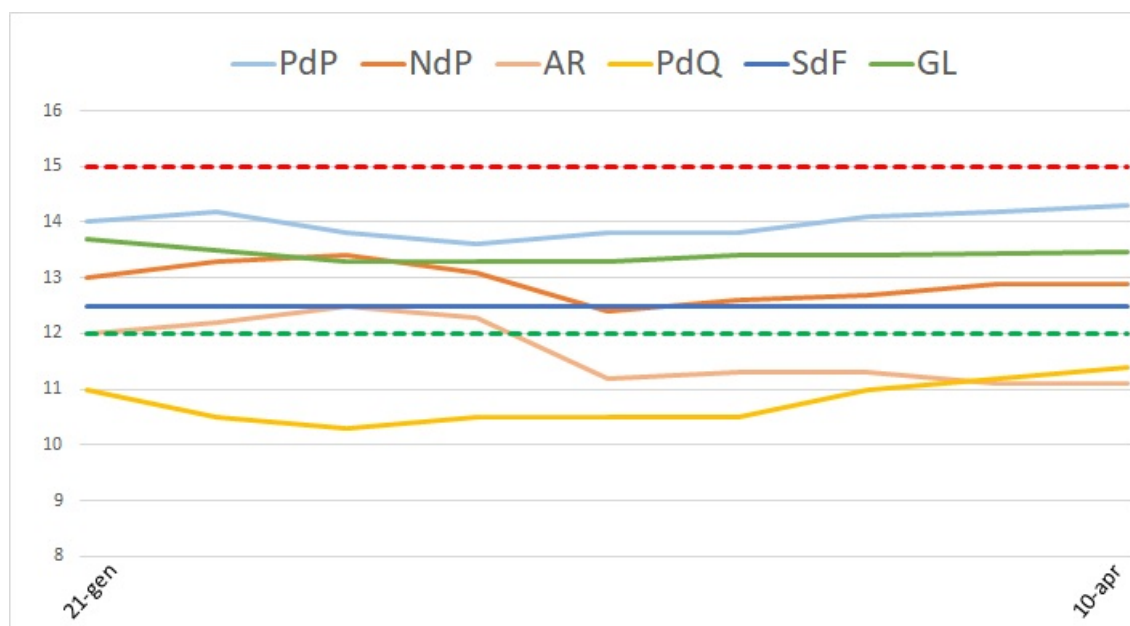


Figura 26: Misurazione Gunning fog index

B.2.16 Indice di Gulpease

Tutti gli indici Gulpease dei documenti rientrano nei vincoli dati. Per questo motivo i documenti redatti hanno raggiunto la leggibilità desiderata.

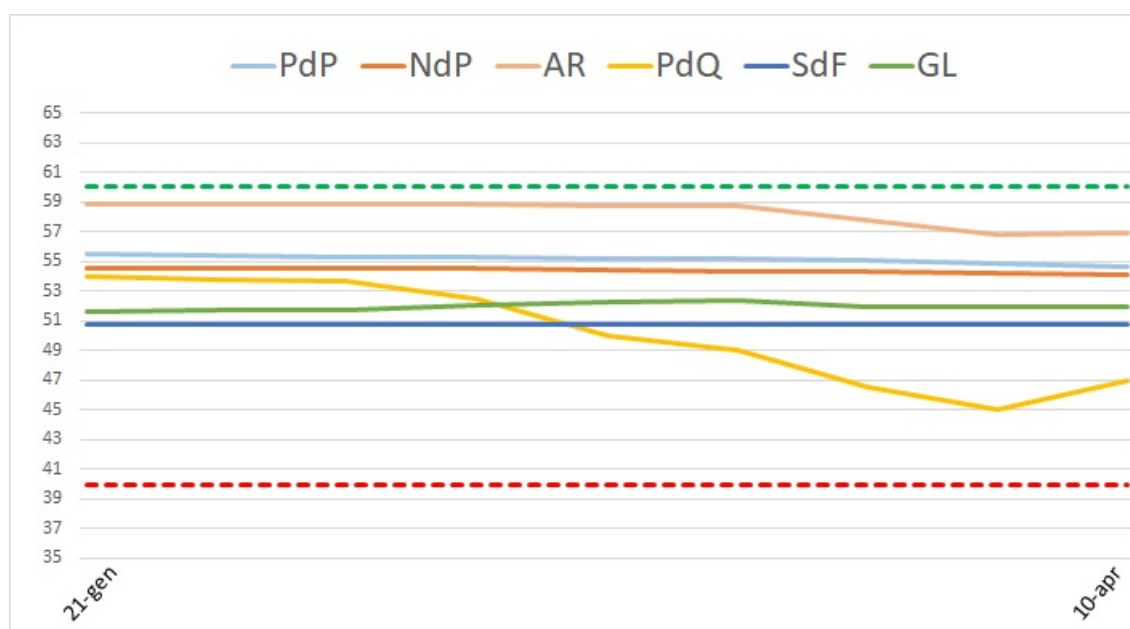


Figura 27: Misurazione indice di Gulpease

B.2.17 Numero di errori grammaticali

Grazie le procedura di verifica e correzione attuate e all'ausilio dello strumento di controllo ortografico integrato in *TexStudio_g*, il numero di errori ortografici per ogni documento ha

raggiunto il valore ottimale di 0.

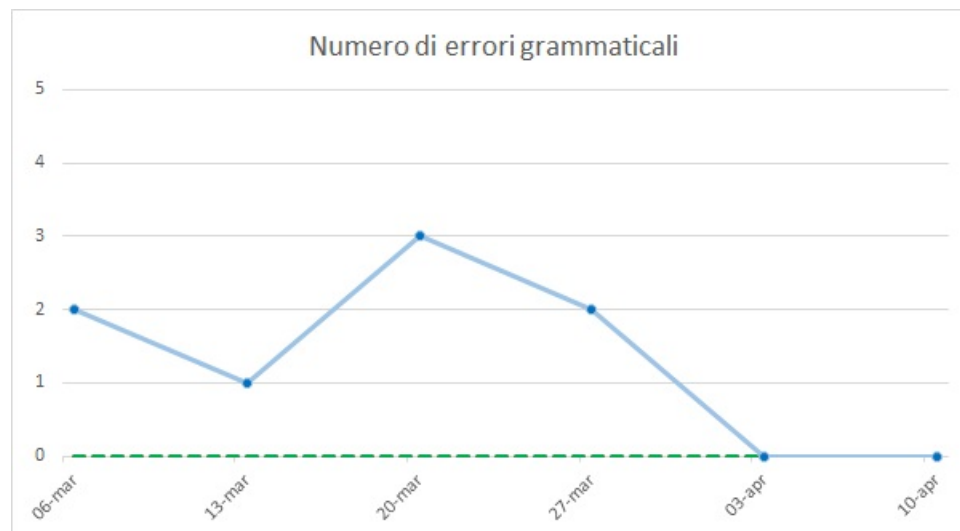


Figura 28: Misurazione numero di errori grammaticali

B.2.18 Functional Implementation Completeness

La scarsa copertura delle funzionalità inizialmente ha fatto registrare valori non accettabili, ma grazie al lavoro del team questa metrica ha raggiunto un valore accettabile, anche se non ancora ottimale.

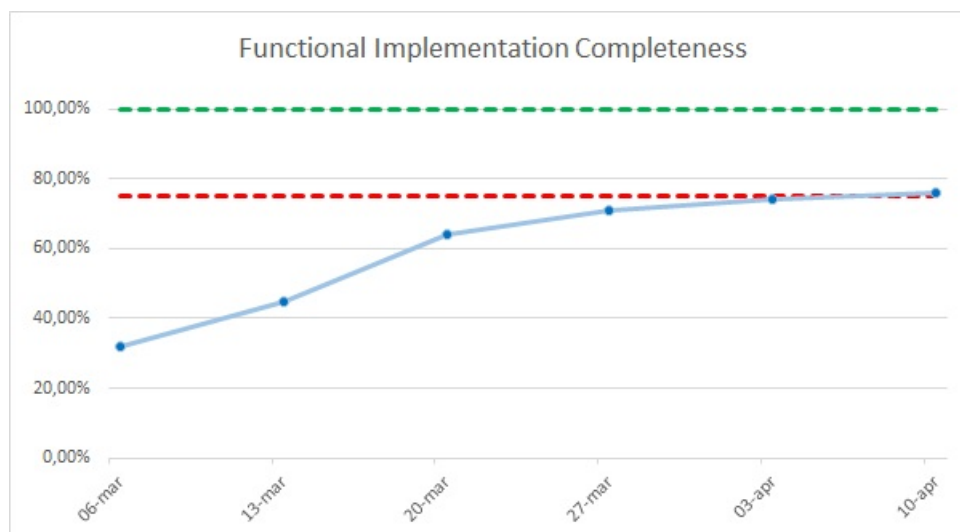


Figura 29: Misurazione Functional Implementation Completeness

B.2.19 Average Functional Implementation Correctness

La presenza di alcune criticità inizialmente ha fatto registrare valori non accettabili, ma grazie al lavoro del team questa metrica ha raggiunto un valore accettabile, anche se non ancora ottimale.

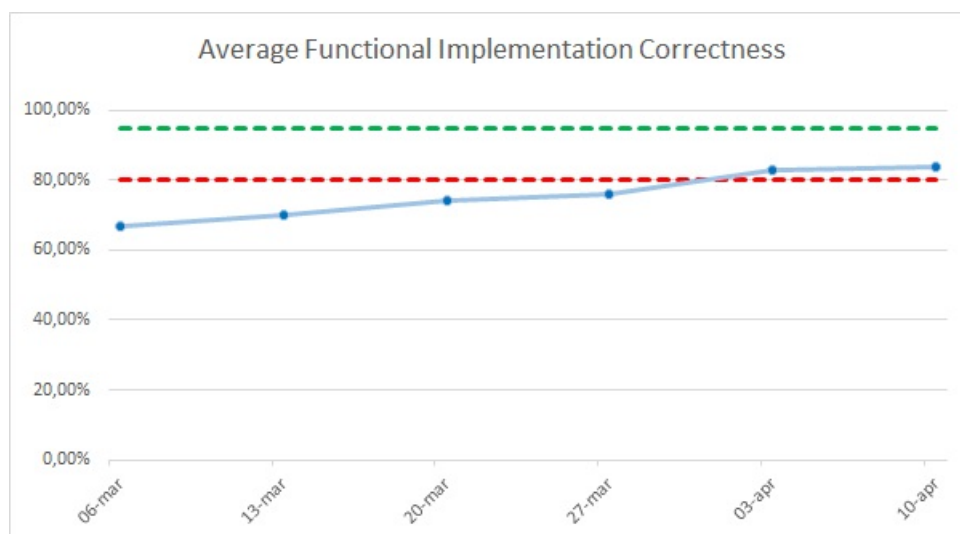


Figura 30: Misurazione Average Functional Implementation Correctness

B.2.20 Tempo di risposta

Il valore di questa metrica è sempre rimasto in linea con il valore ottimale.

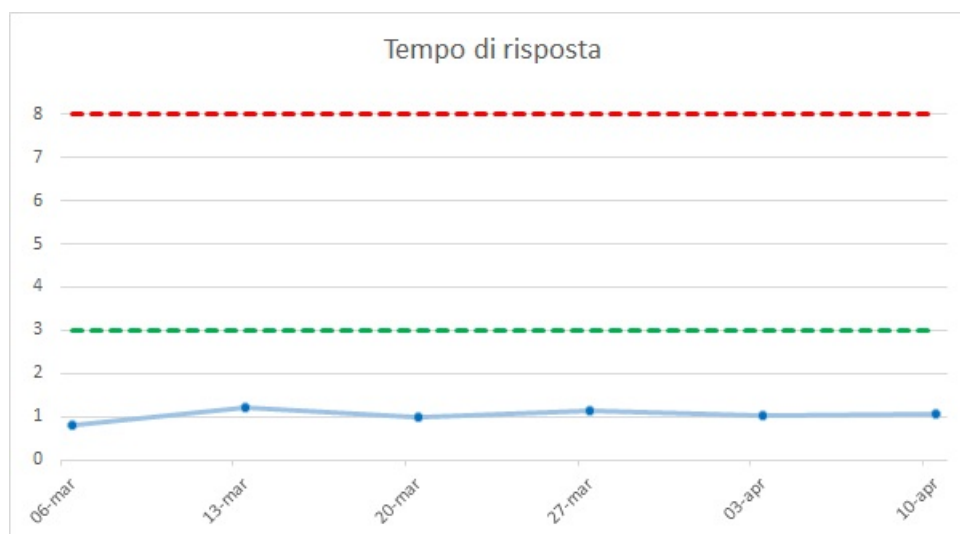


Figura 31: Misurazione tempo di risposta

B.2.21 Average Learning Time

Il valore di questa metrica è sempre rimasto in linea con il valore accettabile.

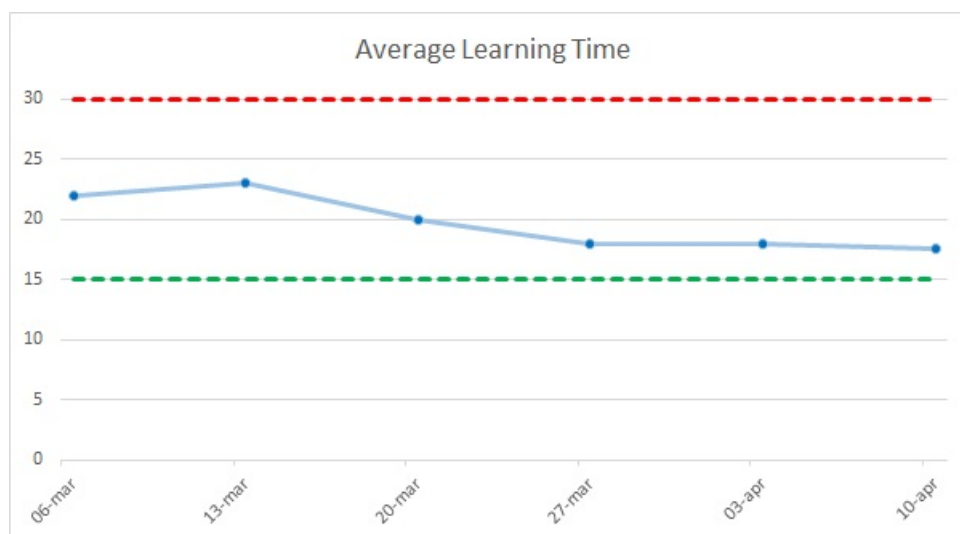


Figura 32: Misurazione Average Learning Time

B.2.22 Failure Density

La presenza di alcune criticità iniziali ha fatto registrare valori non accettabili, ma grazie al lavoro del team questa metrica ha raggiunto un valore accettabile, anche se non ancora ottimale.

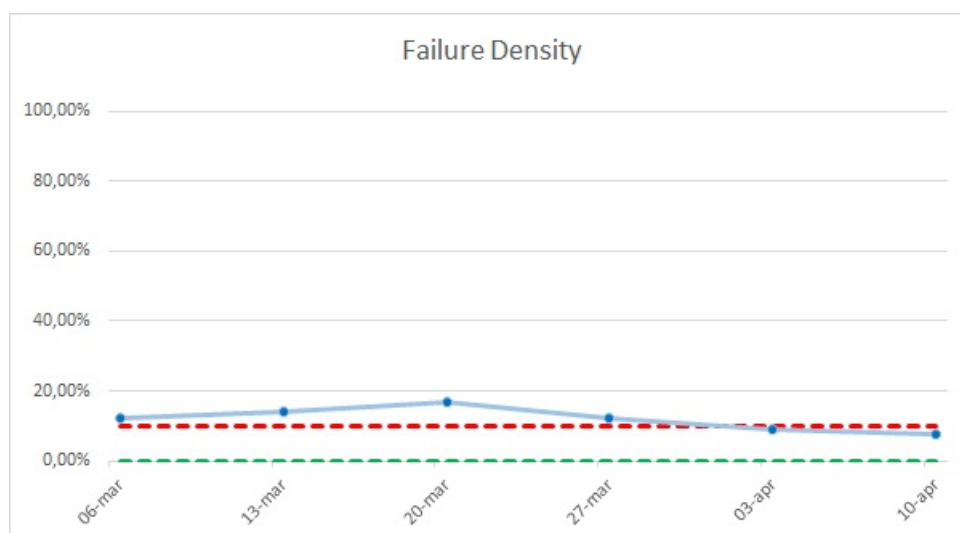


Figura 33: Misurazione Failure Density

B.2.23 Operazioni con gestione errori

Inizialmente abbiamo registrato dei valori non accettabili a causa della scarsa gestione degli errori, ma grazie allo sviluppo dei test siamo riusciti ad ottenere un valore accettabile per questa metrica.

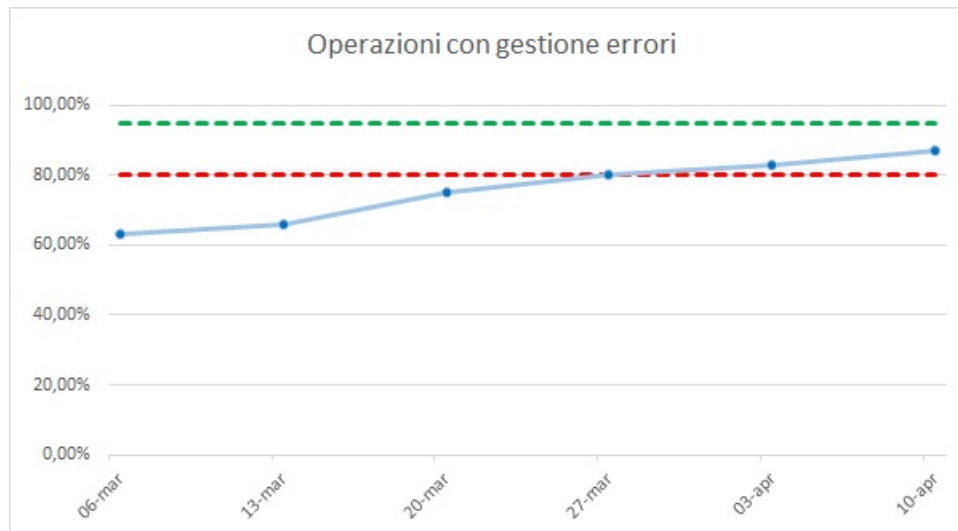


Figura 34: Misurazione

B.2.24 Failure Analysis

Il valore di questa metrica è sempre rimasto in linea con il valore accettabile.

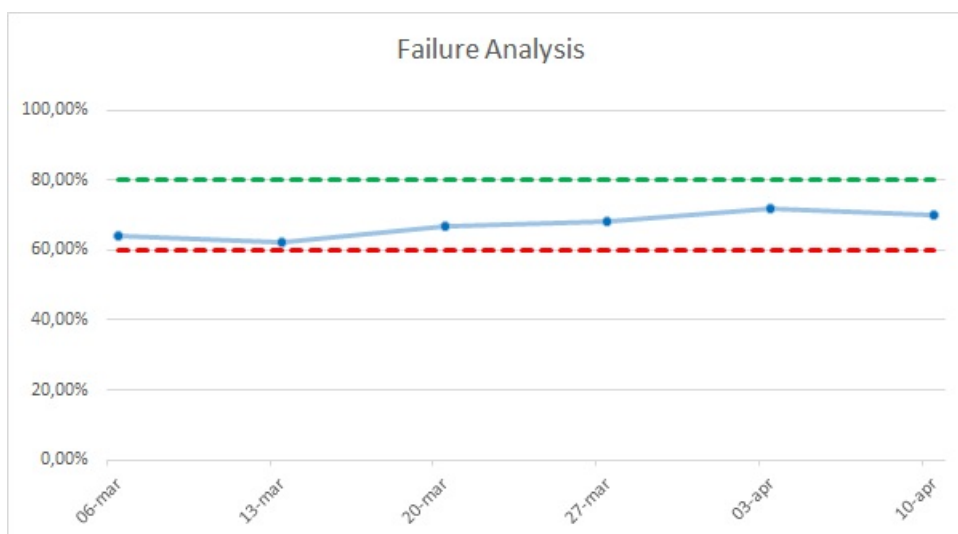


Figura 35: Misurazione Failure Analysis

B.2.25 Comment Ratio

La presenza di pochi commenti nel codice inizialmente ha fatto registrare dei valori non accettabili, ma con il lavoro del team questa metrica ha raggiunto un valore accettabile.

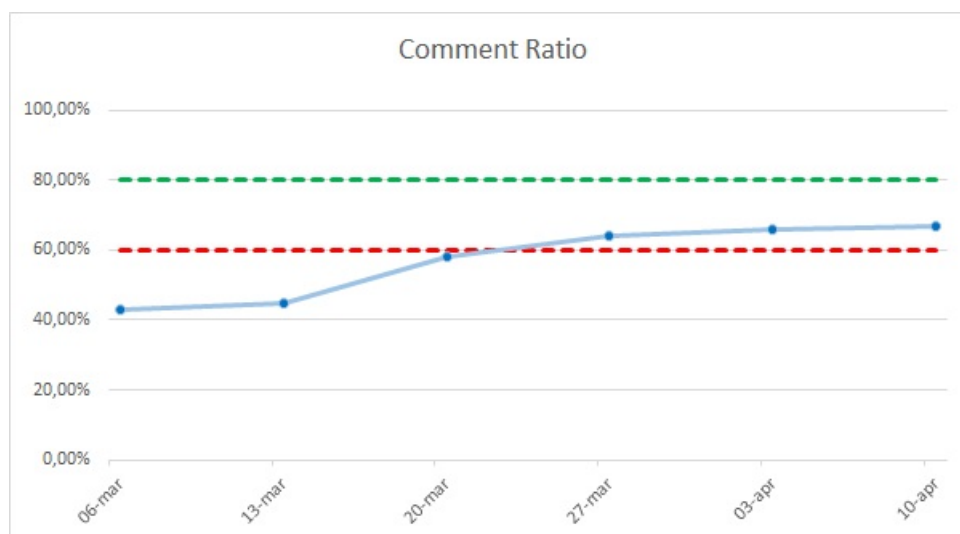


Figura 36: Misurazione Comment Ratio

C Specifica dei test

C.1 Test di accettazione

Test	Specifica
TA-1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di leggere la definizione di una rete Bayesiana importando un file in formato JSON; • Caso: <ul style="list-style-type: none"> – Input: file JSON con la definizione di rete; – Risultato atteso: il sistema importa la definizione della rete mostrandone il contenuto. • Procedura: <ul style="list-style-type: none"> – L'utente crea un nuovo 7DOS panel; – L'utente si posiziona sulla scheda di <i>"Edit"</i>; – L'utente si posiziona sul <i>"JSON import"</i> panel; – L'utente preme il pulsante per importare un file JSON; – L'utente seleziona il file da importare; – Conferma la selezione del file.
TA-2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta la gestione della connessione tra i nodi della rete ed il flusso di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua la connessione tra flusso dati e nodi della rete. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda di <i>"Edit"</i>; – L'utente si posiziona sul <i>"Network"</i> panel; – L'utente seleziona il flusso di dati da monitorare.
TA-2.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di connettere un nodo della rete al flusso di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema connette il nodo selezionato e lo monitora rispetto al flusso dati. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda di <i>"Edit"</i>; – L'utente si posiziona sul <i>"Network"</i> panel; – L'utente seleziona il nodo da monitorare rispetto al flusso di dati.

Test	Specifica
TA-2.2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di disconnettere un nodo della rete al flusso di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema disconnette il nodo selezionato. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda di <i>"Edit"</i>; – L'utente si posiziona sul <i>"Network"</i> panel; – L'utente deselecta il nodo da monitorare rispetto al flusso di dati.
TA-2.3	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di modificare un nodo della rete connesso al flusso di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema apporta le modifiche richieste. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda di <i>"Edit"</i>; – L'utente si posiziona sul <i>"Network"</i> panel; – L'utente modifica il nodo da monitorare rispetto al flusso di dati.
TA-3	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di applicare il ricalcolo delle probabilità della rete secondo regole temporali stabilite dall'utente.; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua il ricalcolo delle probabilità della rete aggiornandone lo stato dei nodi. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i dati provenienti dal flusso; – Il sistema li elabora calcolando le probabilità dei nodi; – Il sistema aggiorna lo stato dei nodi.

Test	Specifica
TA-3.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di modificare le regole temporali per effettuare il ricalcolo delle probabilità della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema apporta le modifiche richieste alle regole temporali per il ricalcolo delle probabilità. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda di <i>"Edit"</i>; – L'utente si posiziona sul <i>"Network"</i> panel; – L'utente modifica le regole temporali per il ricalcolo delle probabilità della rete.
TA-4	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di fornire nuovi dati a Grafana derivati dai nodi della rete non direttamente collegati al flusso di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema elabora i dati proveniente dai nodi non connessi al flusso. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati; – Il sistema usa i risultati ottenuti per calcolare le probabilità dei nodi non connessi.
TA-4.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta l'aggiornamento dei dati secondo una frequenza stabilita dall'utente; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua l'aggiornamento costante dei dati. • Procedura: <ul style="list-style-type: none"> – Il sistema, dopo un tempo predefinito, legge nuovi dati in ingresso; – Il sistema elabora i nuovi dati.

Test	Specifica
TA-5	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di leggere i dati provenienti dal flusso ed elaborarli, visualizzandone il risultato attraverso un grafico; • Caso: <ul style="list-style-type: none"> – Input: flusso di dati; – Risultato atteso: grafico derivato dall'elaborazione dei dati. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati ricevuti; – Il sistema costruisce un grafico sulla base dei risultati ottenuti.
TA-5.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema aggiorni i dati presenti nel grafico in base alla frequenza stabilita dall'utente; • Caso: <ul style="list-style-type: none"> – Input: nuovi dati dal flusso; – Risultato atteso: il sistema aggiorna i dati nel grafico. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i nuovi dati proveniente dal flusso, in base alla frequenza stabilita; – Il sistema effettua il ricalcolo delle probabilità della rete aggiornando lo stato dei nodi, in base alla frequenza stabilita; – Il sistema elabora i risultati e li mostra attraverso un grafico.
TA-5.2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di creare un nuovo panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema crea il nuovo panel e lo mostra all'interno della dashboard. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sulla voce <i>add panel</i>; – L'utente sceglie la tipologia di panel da creare.

Test	Specifica
TA-5.3	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di spostare un panel all'interno della dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema sposta panel nella nuova posizione scelta dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente trascina il panel che vuole spostare nella posizione desiderata.
TA-5.4	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di cancellare un panel all'interno della dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema cancella panel scelto dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sull'intestazione del panel; – L'utente seleziona la voce <i>"Remove"</i>; – Conferma l'eliminazione del panel.
TA-5.5	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di minimizzare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica la dimensione del panel come scelto dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente si posiziona sull'angolo in basso a destra del panel; – L'utente modifica minimizza il panel.

Test	Specifica
TA-5.6	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente configurare un panel dopo la sua creazione; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema configura le opzioni del panel come stabilito dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente configura le opzioni del panel come desidera.
TA-5.7	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di modificare un panel all'interno della dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica le opzioni del panel come stabilito dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente modifica le opzioni del panel come desidera.
TA-6	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di definire alert in base a livelli di soglia raggiunti dai nodi non collegati al flusso dei dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema crea un nuovo alert. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di un panel di tipo <i>graph</i>; – L'utente si posiziona sulla schermata di <i>edit</i>; – L'utente si posiziona sulla scheda "<i>Alert</i>"; – L'utente clicca sul pulsante "<i>Create alert</i>".

Test	Specifica
TA-6.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di configurare i parametri di un alert; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema configura i parametri dell'alert. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda "Alert"; – L'utente seleziona l'alert da configurare; – L'utente configura i parametri come desidera.
TA-6.2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta all'utente di impostare il modo in cui viene notificata l'attivazione di un alert; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema imposta il sistema di notifica dell'alert. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda "Alert"; – L'utente seleziona l'alert da configurare; – L'utente imposta come deve venire notificata l'attivazione dell'alert.
TA-7	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di disegnare una rete Bayesiana con un editor grafico specializzato; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: la rete Bayesiana è stata disegnata. • Procedura: <ul style="list-style-type: none"> – L'utente avvia l'editor grafico; – L'utente inizia a disegnare una rete Bayesiana;
TA-7.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di creare un nuovo nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il nuovo della rete è stato creato. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>add node</i>; – L'utente sceglie la posizione in cui crearlo.

Test	Specifica
TA-7.2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di modificare i parametri di un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: i parametri del nodo vengono modificati. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente modifica i parametri secondo le sue preferenze; – L'utente conferma le modifiche.
TA-7.3	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di creare un collegamento tra due nodi della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il nuovo collegamento viene creato. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"add link"</i>; – L'utente sceglie i nodi da collegare.
TA-7.4	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di eliminare un collegamento tra due nodi della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il collegamento viene eliminato. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"remove link"</i>; – L'utente sceglie il collegamento da rimuovere.
TA-7.5	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di salvare la rete realizzata su un file JSON; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il file JSON con la definizione della rete viene salvato. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"save network"</i>; – L'utente sceglie il percorso in cui salvare il file; – L'utente conferma la scelta.

Test	Specifica
TA-7.6	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi e sia in grado di gestire eventuali errori derivati dalla modifica di un nodo; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente effettua delle modifiche non valide.
TA-8	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di applicare più reti Bayesiane a diversi oggetti di monitoraggio; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema avvia il monitoraggio di diversi oggetti. • Procedura: <ul style="list-style-type: none"> – L'utente importa più reti Bayesiane; – L'utente connette le reti a diversi flussi di dati.
TA-9	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di creare una rete Bayesiana a partire dai dati raccolti sul campo; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema crea una rete Bayesiana. • Procedura: <ul style="list-style-type: none"> – Il sistema raccoglie i dati provenienti dal flusso; – Il sistema li elabora per costruire una rete Bayesiana.
TA-10	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di condividere grafici presenti in una dashboard o in un singolo panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema condivide il grafico secondo l'opzione scelta dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa alla dashboard o ad un panel.

Test	Specifica
TA-10.1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di visualizzare il link diretto ad una dashboard o ad un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra il link generato con cui l'utente può effettuare la condivisione. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa alla dashboard o ad un panel; – L'utente si posiziona sulla scheda "<i>Link</i>"; – L'utente configura le opzioni per generare il link per la condivisione.
TA-10.2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di visualizzare il codice per l'inclusione di un panel in una pagina web; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: Il sistema mostra il frame generato con cui l'utente può effettuare la condivisione. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel; – L'utente si posiziona sulla scheda "<i>Embed</i>"; – L'utente configura le opzioni per generare il frame per la condivisione.
TA-10.4	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di condividere lo snapshot di una dashboard o di un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua la condivisione dello snapshot. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente configura le opzioni per la condivisione dello snapshot.

Test	Specifica
TA-10.5	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di visualizzare il codice JSON contenente la definizione di una dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra il codice JSON con la definizione della dashboard. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione <i>"Share dashboard"</i>; – L'utente si posiziona sulla scheda <i>"Export"</i>; – L'utente clicca sul pulsante per visualizzare il JSON.
TA-10.6	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema permetta di salvare il file JSON contenente la definizione di una dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua il salvataggio del codice JSON con la definizione della dashboard. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione <i>"Share dashboard"</i>; – L'utente si posiziona sulla scheda <i>"Export"</i>; – L'utente clicca sul pulsante per salvare il JSON.

Tabella 12: Specifica test di accettazione

C.2 Test di sistema

Test	Specifica
TS-1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile e leggere la definizione della rete Bayesiana da un file in formato JSON; • Caso: <ul style="list-style-type: none"> – Input: file JSON con la definizione di una rete Bayesiana; – Risultato atteso: il sistema legge il JSON e costruisce la rete. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante per importare un file; – L'utente seleziona il file; – L'utente conferma la selezione.
TS-1.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile validare un file JSON; • Caso: <ul style="list-style-type: none"> – Input: file JSON; – Risultato atteso: il sistema mostra degli errori se il file non è valido. • Procedura: <ul style="list-style-type: none"> – L'utente seleziona un JSON da importare; – L'utente conferma la selezione; – Il sistema verifica la validità del file.
TS-2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile gestire la connessione tra i nodi della rete ai rispettivi flussi di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua la connessione tra flusso dati e nodi della rete. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sul "Network" panel; – L'utente seleziona il flusso dati da monitorare.

Test	Specifica
TS-2.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile connettere un nodo della rete ad un flusso di dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema connette il nodo selezionato al flusso dati. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sul "<i>Network</i>" panel; – L'utente sceglie quale nodo monitorare rispetto al flusso dati.
TS-2.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile disconnettere un nodo della rete da un flusso di dati.; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema disconnette il nodo selezionato dal flusso dati. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sul "<i>Network</i>" panel; – L'utente sceglie quale nodo disconnettere rispetto al flusso dati.
TS-2.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia essere possibile modificare il flusso di dati connesso ad un nodo; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema seleziona un altro flusso di dati da associare alla rete. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sul "<i>Network</i>" panel; – L'utente modifica la fonte del flusso dati.

Test	Specifica
TS-3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile applicare il ricalcolo delle probabilità della rete secondo regole temporali prestabilite; • Caso: <ul style="list-style-type: none"> – Input: flusso di dati; – Risultato atteso: il sistema effettua il ricalcolo delle probabilità della rete aggiornandone lo stato dei nodi. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i dati provenienti dal flusso; – Il sistema li elabora calcolando le probabilità dei nodi; – Il sistema aggiorna lo stato dei nodi.
TS-3.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare le suddette regole temporali; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema apporta le modifiche richieste alle regole temporali per il ricalcolo delle probabilità. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda di "Edit"; – L'utente si posiziona sul "Network" panel; – L'utente modifica le regole temporali per il ricalcolo delle probabilità della rete.
TS-4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile fornire nuovi dati al sistema di Grafana derivati dai nodi della rete non collegati al flusso di monitoraggio; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema elabora i dati proveniente dai nodi non connessi al flusso. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati; – Il sistema usa i risultati ottenuti per calcolare le probabilità dei nodi non connessi.

Test	Specifica
TS-4.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile aggiornare i dati in base alla frequenza stabilita; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua l'aggiornamento costante dei dati. • Procedura: <ul style="list-style-type: none"> – Il sistema, dopo un tempo predefinito, legge nuovi dati in ingresso; – Il sistema elabora i nuovi dati.
TS-5	<ul style="list-style-type: none"> • Progettazione: verifica che i dati siano disponibili al sistema di creazione di grafici e dashboard per la loro visualizzazione; • Caso: <ul style="list-style-type: none"> – Input: flusso di dati; – Risultato atteso: grafico derivato dall'elaborazione dei dati. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati ricevuti; – Il sistema costruisce un grafico sulla base dei risultati ottenuti.
TS-5.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile aggiornare la dashboard in base alla frequenza stabilita; • Caso: <ul style="list-style-type: none"> – Input: nuovi dati dal flusso; – Risultato atteso: il sistema aggiorna i dati nel grafico. • Procedura: <ul style="list-style-type: none"> – Il sistema legge i nuovi dati proveniente dal flusso, in base alla frequenza stabilita; – Il sistema effettua il ricalcolo delle probabilità della rete aggiornando lo stato dei nodi, in base alla frequenza stabilita; – Il sistema elabora i risultati e li mostra attraverso un grafico.

Test	Specifica
TS-5.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile creare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema crea il nuovo panel e lo mostra all'interno della dashboard. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sulla voce <i>add panel</i>; – L'utente sceglie la tipologia di panel da creare.
TS-5.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile spostare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema sposta panel nella nuova posizione scelta dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente trascina il panel che vuole spostare nella posizione desiderata.
TS-5.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile cancellare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema cancella panel scelto dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sull'intestazione del panel; – L'utente seleziona la voce <i>"Remove"</i>; – Conferma l'eliminazione del panel.
TS-5.5	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile minimizzare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica la dimensione del panel come scelto dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente si posiziona sull'angolo in basso a destra del panel; – L'utente modifica minimizza il panel.

Test	Specifica
TS-5.6	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile configurare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema configura le opzioni del panel come stabilito dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente configura le opzioni del panel come desidera.
TS-5.6.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare un flusso dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema associa i dati in ingresso al flusso. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sul <i>Network</i> panel; – L'utente seleziona la fonte del flusso dati.
TS-5.6.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: . • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sul <i>Network</i> panel; – L'utente seleziona il nodo da monitorare.
TS-5.6.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare un intervallo di tempo; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra nel grafico i dati elaborati in quell'intervallo di tempo. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni della dashboard; – L'utente seleziona l'intervallo di tempo.

Test	Specifica
TS-5.7	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica le opzioni del panel come stabilito dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente modifica le opzioni del panel come desidera.
TS-5.7.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile usare le modifiche standard di Grafana su un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema apporta le modifiche richieste. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del panel; – L'utente apporta le modifiche che desidera.
TS-6	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile definire alert in base a livelli di soglia raggiunti dai nodi non collegati al flusso dei dati; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema crea un nuovo alert. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di un panel di tipo <i>graph</i>; – L'utente si posiziona sulla schermata di <i>edit</i>; – L'utente si posiziona sulla scheda "<i>Alert</i>"; – L'utente clicca sul pulsante "<i>Create alert</i>".
TS-6.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile configurare i parametri di un alert; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema configura i parametri dell'alert. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda "<i>Alert</i>"; – L'utente seleziona l'alert da configurare; – L'utente configura i parametri come desidera.

Test	Specifica
TS-6.1.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inserire il nome di un alert; • Caso: <ul style="list-style-type: none"> – Input: stringa col nome dell>alert; – Risultato atteso: il sistema inserisce il nome dell>alert. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del panel; – L'utente scrive il nome dell>alert. – L'utente conferma il nome scelto.
TS-6.1.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inserire l'intervallo di verifica di un alert; • Caso: <ul style="list-style-type: none"> – Input: numero intero; – Risultato atteso: il sistema inserisce il tempo per l'intervallo di verifica. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del panel ; – L'utente seleziona il tempo per l'intervallo di verifica. – L'utente conferma la selezione.
TS-6.1.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inserire la condizione di attivazione di un alert; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema inserisce la condizione di attivazione. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del panel ; – L'utente inserisce la condizione di attivazione. – L'utente conferma l'inserimento.

Test	Specifica
TS-6.2	<ul style="list-style-type: none"> ● Progettazione: verifica che sia possibile impostare il modo in cui viene notificata l'attivazione di un alert; ● Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema imposta il sistema di notifica dell'alert. ● Procedura: <ul style="list-style-type: none"> – L'utente si posiziona sulla scheda "Alert"; – L'utente seleziona l'alert da configurare; – L'utente imposta come deve venire notificata l'attivazione dell'alert.
TS-7	<ul style="list-style-type: none"> ● Progettazione: verifica che sia possibile disegnare la rete Bayesiana con un editor grafico specializzato; ● Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: la rete Bayesiana è stata disegnata. ● Procedura: <ul style="list-style-type: none"> – L'utente avvia l'editor grafico; – L'utente inizia a disegnare una rete Bayesiana;
TS-7.1	<ul style="list-style-type: none"> ● Progettazione: verifica che sia possibile creare un nodo della rete; ● Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il nuovo della rete è stato creato. ● Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>add node</i>; – L'utente sceglie la posizione in cui crearlo.
TS-7.1.1	<ul style="list-style-type: none"> ● Progettazione: verifica che sia possibile inizializzare la lista di predecessori del nodo; ● Caso: <ul style="list-style-type: none"> – Input: elenco dei predecessori; – Risultato atteso: la lista dei predecessori viene inizializzata. ● Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce l'elenco dei predecessori.

Test	Specifica
TS-7.1.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inizializzare la lista di successori del nodo; • Caso: <ul style="list-style-type: none"> – Input: elenco dei successori; – Risultato atteso: la lista dei successori viene inizializzata. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce l'elenco dei successori.
TS-7.1.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inizializzare il nome del nodo; • Caso: <ul style="list-style-type: none"> – Input: stringa col nome del nodo; – Risultato atteso: il nome del nodo viene inizializzato. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce il nome.
TS-7.1.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inizializzare la CPT associata al nodo; • Caso: <ul style="list-style-type: none"> – Input: dati della CPT; – Risultato atteso: la CPT di un nodo viene inizializzata. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce i dati per la CPT.
TS-7.1.4.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inizializzare la lista degli stati associata alla CPT del nodo; • Caso: <ul style="list-style-type: none"> – Input: elenco degli stati; – Risultato atteso: la lista degli stati associata alla CPT viene inizializzata. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce gli stati associati alla CPT.

Test	Specifica
TS-7.1.4.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inizializzare la lista delle combinazioni degli stati dei nodi predecessori associata alla CPT del nodo; • Caso: <ul style="list-style-type: none"> – Input: elenco delle combinazioni; – Risultato atteso: la lista delle combinazioni degli stati dei nodi predecessori viene inizializzata. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce le combinazioni degli stati dei nodi predecessori.
TS-7.1.4.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inizializzare le celle della CPT; • Caso: <ul style="list-style-type: none"> – Input: dati della CPT; – Risultato atteso: le celle della CPT vengono inizializzate. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce i dati per le celle della CPT.
TS-7.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare i parametri di un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica i parametri di un nodo. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente modifica i parametri.
TS-7.2.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare il nome di un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: stringa con nome del nodo; – Risultato atteso: il sistema modifica il nome del nodo. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente inserisce il nuovo nome del nodo.

Test	Specifica
TS-7.2.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare la CPT associata ad un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: dati della CPT; – Risultato atteso: il sistema modifica la CPT. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente modifica la CPT.
TS-7.2.2.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile aggiungere uno stato alla CPT associata ad un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema aggiunge un nuovo stato alla CPT. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente aggiunge un nuovo stato alla CPT.
TS-7.2.2.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile eliminare uno stato dalla CPT associata ad un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema elimina uno stato dalla CPT. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente rimuove uno stato dalla CPT.
TS-7.2.2.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare i parametri associati ad uno stato della CPT associata ad un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica i parametri associati ad uno stato della CPT. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente modifica i parametri della CPT.

Test	Specifica
TS-7.2.2.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile modificare una cella della CPT; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema modifica la cella della CPT. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni del nodo; – L'utente modifica la cella della CPT.
TS-7.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile eliminare un nodo dalla rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema elimina il nodo dalla rete. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"add link"</i>; – L'utente sceglie i nodi da collegare.
TS-7.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile creare un collegamento tra due nodi della rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il collegamento viene eliminato. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"remove link"</i>; – L'utente sceglie il collegamento da rimuovere.
TS-7.4.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile indicare il nodo di partenza del collegamento; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema seleziona il nodo di partenza. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"add link"</i>; – L'utente sceglie il nodo di partenza.

Test	Specifica
TS-7.4.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile indicare il nodo di arrivo del collegamento; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema seleziona il nodo di partenza. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"add link"</i>; – L'utente sceglie il nodo di arrivo.
TS-7.5	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile eliminare un collegamento dalla rete; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il file JSON con la definizione della rete viene salvato. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>"save network"</i>; – L'utente sceglie il percorso in cui salvare il file; – L'utente conferma la scelta.
TS-7.6	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile salvare la rete su file JSON; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente effettua delle modifiche non valide.
TS-7.6.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile indicare il nome del file JSON su cui si vuole salvare la struttura della rete; • Caso: <ul style="list-style-type: none"> – Input: stringa con il nome del file; – Risultato atteso: il sistema inserisce il nome del file. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>save network</i>; – L'utente inserisce il nome del file.

Test	Specifica
TS-7.6.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile indicare il percorso del file system in cui si vuole salvare il file JSON contenente la struttura della rete; • Caso: <ul style="list-style-type: none"> – Input: percorso del file system; – Risultato atteso: il sistema inserisci il percorso in cui salvare il file. • Procedura: <ul style="list-style-type: none"> – L'utente preme il pulsante <i>save network</i>; – L'utente inserisce percorso del file.
TS-7.7	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile gestire errori relativi alla modifica di un nodo; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente effettua delle modifiche non valide.
TS-7.7.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile gestire l'inserimento di valori non validi per il nome di un nodo; • Caso: <ul style="list-style-type: none"> – Input: stringa non valida; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente inserisce un nome non valido.
TS-7.7.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile gestire l'inserimento di valori non validi per il nome di uno stato associato alla CPT di un nodo della rete; • Caso: <ul style="list-style-type: none"> – Input: stringa non valida; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente inserisce un nome non valido.

Test	Specifica
TS-7.7.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile gestire l'inserimento di valori non validi per l'intervallo associato ad uno stato del nodo; • Caso: <ul style="list-style-type: none"> – Input: numero intero; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente inserisce un valore non valido.
TS-7.7.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia e possibile gestire l'inserimento di valori non validi per una cella della tabella; • Caso: <ul style="list-style-type: none"> – Input: numero intero; – Risultato atteso: il sistema mostra un messaggio di errore. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona nelle impostazioni di un nodo; – L'utente inserisce un valore non valido.
TS-8	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile applicare più reti Bayesiane in oggetti di monitoraggio diversi; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema avvia il monitoraggio di diversi oggetti. • Procedura: <ul style="list-style-type: none"> – L'utente importa più reti Bayesiane; – L'utente connette le reti a diversi flussi di dati.
TS-9	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile creare una rete Bayesiana a partire dai dati raccolti sul campo anziché svilupparla con la collaborazione degli esperti del settore; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema crea una rete Bayesiana. • Procedura: <ul style="list-style-type: none"> – Il sistema raccoglie i dati provenienti dal flusso; – Il sistema li elabora per costruire una rete Bayesiana.

Test	Specifica
TS-10	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile condividere un grafico; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema condivide il grafico secondo l'opzione scelta dall'utente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa alla dashboard o ad un panel.
TS-10.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile visualizzare il link diretto ad una dashboard o ad un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra il link generato con cui l'utente può effettuare la condivisione. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa alla dashboard o ad un panel; – L'utente si posiziona sulla scheda "<i>Link</i>"; – L'utente configura le opzioni per generare il link per la condivisione.
TS-10.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile visualizzare il codice per l'inclusione di un panel in una pagina web; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra il frame generato con cui l'utente può effettuare la condivisione. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel; – L'utente si posiziona sulla scheda "<i>Embed</i>"; – L'utente configura le opzioni per generare il frame per la condivisione.

Test	Specifica
TS-10.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare le opzioni di visualizzazione per la condivisione dei grafici; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema seleziona le opzioni richieste. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel o una dashboard; – L'utente configura le opzioni di visualizzazione.
TS-10.3.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare la visualizzazione dell'intervallo di tempo corrente in un grafico; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra l'intervallo di tempo corrente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel o una dashboard; – L'utente seleziona la visualizzazione dell'intervallo di tempo.
TS-10.3.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile deselezionare la visualizzazione dell'intervallo di tempo corrente in un grafico; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema nasconde l'intervallo di tempo corrente. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel o una dashboard; – L'utente deseleziona la visualizzazione dell'intervallo di tempo.

Test	Specifica
TS-10.3.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare la visualizzazione di variabili di template in un grafico; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra le variabili di template. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel o una dashboard; – L'utente seleziona la visualizzazione delle variabili di template.
TS-10.3.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile deselegionare la visualizzazione di variabili di template in un grafico; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema nasconde le variabili di template. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel o una dashboard; – L'utente deselegiona la visualizzazione delle variabili di template.
TS-10.3.5	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare il tema di un grafico; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra il tema selezionato. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa ad un panel o una dashboard; – L'utente seleziona il tema.

Test	Specifica
TS-10.4	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile condividere uno snapshot di una dashboard o di un panel; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua la condivisione dello snapshot. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente configura le opzioni per la condivisione dello snapshot.
TS-10.4.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile pubblicare uno snapshot sull'istanza locale dell'utente; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema pubblica lo snapshot sull'istanza. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente preme il pulsante per pubblicare lo snapshot sulla sua istanza.

Test	Specifica
TS-10.4.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile pubblicare uno snapshot su Raintank; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema pubblica lo snapshot su Raintank. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente preme il pulsante per pubblicare lo snapshot su Raintank.
TS-10.4.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile configurare le opzioni di visualizzazione di uno snapshot; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema configura le opzioni selezionate. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente configura le opzioni di visualizzazione.
TS-10.4.3.1	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inserire il nome di uno snapshot; • Caso: <ul style="list-style-type: none"> – Input: stringa con il nome; – Risultato atteso: il sistema inserisce il nome dello snapshot. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente inserisce il nome dello snapshot.

Test	Specifica
TS-10.4.3.2	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile selezionare il tempo di permanenza di uno snapshot; • Caso: <ul style="list-style-type: none"> – Input: numero intero; – Risultato atteso: il sistema inserisce il tempo di permanenza. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente inserisce il tempo di permanenza dello snapshot.
TS-10.4.3.3	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile inserire il tempo massimo per il caricamento dei dati in uno snapshot; • Caso: <ul style="list-style-type: none"> – Input: numero intero; – Risultato atteso: il sistema inserisce il tempo di timeout. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share</i>" relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda "<i>Snapshot</i>"; – L'utente inserisce il tempo di timeout dello snapshot.
TS-10.5	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile visualizzare il codice JSON contenente la definizione di una dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema mostra il codice JSON con la definizione della dashboard. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione "<i>Share dashboard</i>"; – L'utente si posiziona sulla scheda "<i>Export</i>"; – L'utente clicca sul pulsante per visualizzare il JSON.

Test	Specifica
TS-10.6	<ul style="list-style-type: none"> • Progettazione: verifica che sia possibile salvare il file JSON contenente la definizione di una dashboard; • Caso: <ul style="list-style-type: none"> – Input: nessuno; – Risultato atteso: il sistema effettua il salvataggio del codice JSON con la definizione della dashboard. • Procedura: <ul style="list-style-type: none"> – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione <i>"Share dashboard"</i>; – L'utente si posiziona sulla scheda <i>"Export"</i>; – L'utente clicca sul pulsante per salvare il JSON.

Tabella 13: Specifica test di sistema

C.3 Test di integrazione

Test	Specifica
TI-1	verifica l'integrazione tra client e Grafana.
TI-2	verifica l'integrazione tra Grafana ed <i>InfluxDB_g</i> .
TI-3	verifica l'integrazione tra Grafana e JsBayes.
TI-4	verifica l'integrazione tra Grafana e Raintank.

Tabella 14: Specifica test di integrazione

C.4 Test di unità

Test	Specifica
Scope	BoolValue::constructor
TU-1	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi una stringa non valida per il costruttore di <i>BoolValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: stringa non valida; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid string parameter"</i>. • Procedura: <pre> 1 let str: string; 2 expect(() => new BoolValue(true, str)).to.throw(TypeError, "↵ invalid string parameter"); </pre>
TU-2	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi un valore booleano non valido per il costruttore di <i>BoolValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: boolean non valido; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid boolean parameter"</i>. • Procedura: <pre> 1 let str: boolean; 2 expect(() => new BoolValue(str, "name1")).to.throw(TypeError, ↵ "invalid boolean parameter"); </pre>
Scope	BoolValue::isValueType
TU-3	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>isValueType()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i> valido; – Risultato atteso: True. • Procedura: <pre> 1 const boolV = new BoolValue(true, "name2"); 2 const result: boolean = boolV.isValueType("true"); 3 expect(result).to.equal(true); </pre>

Test	Specifica
TU-4	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con un parametro errato; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const boolV = new BoolValue(false, "name3"); 2 const result: boolean = boolV.isValueType("true"); 3 expect(result).to.equal(false); </pre>
TU-5	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con un parametro casuale; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const boolV = new BoolValue(false, "name4"); 2 const result: boolean = boolV.isValueType("brzclld"); 3 expect(result).to.equal(false); </pre>
TU-6	<ul style="list-style-type: none"> ● Progettazione: verifica il corretto funzionamento di <i>isValueType()</i> senza parametro; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const boolV = new BoolValue(true, "name5"); 2 const result: boolean = boolV.isValueType(""); 3 expect(result).to.equal(false); </pre>

Test	Specifica
TU-7	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>isValueType()</i> con un parametro valido ma con la lettera maiuscola; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i>; – Risultato atteso: <i>True</i>. • Procedura: <pre> 1 const boolV = new BoolValue(true, "name6"); 2 const result: boolean = boolV.isValueType("True"); 3 expect(result).to.equal(true); </pre>
TU-8	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>isValueType()</i> con un parametro non definito; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i>; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid parameter"</i>. • Procedura: <pre> 1 const boolV = new BoolValue(true, "name7"); 2 let str: string; 3 expect(() => boolV.isValueType(str)).to.throw(TypeError, "↩ invalid parameter"); </pre>
Scope	BoolValue::getValueName
TU-9	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueName()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>BoolValue</i>; – Risultato atteso: stringa con nome dell'oggetto. • Procedura: <pre> 1 const name: string = "name8"; 2 const boolV = new BoolValue(true, name); 3 boolV.isValueType("true"); 4 boolV.isValueType("false"); 5 const result: string = boolV.getValueName(); 6 expect(result).to.equal(name); </pre>
Scope	RangeValue::constructor

Test	Specifica
TU-10	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi un nome come parametro non valido per il costruttore di <i>RangeValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: stringa col nome non valida; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid name parameter"</i>. • Procedura: <pre> 1 let str: string; 2 expect(() => new RangeValue(30, 70, str)).to.throw(TypeError, ↵ "invalid name parameter"); </pre>
TU-11	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi dei valori di range come parametri non validi per il costruttore di <i>RangeValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: valori del range non validi; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid range parameter"</i>. • Procedura: <pre> 1 let n1: number; 2 let n2: number; 3 expect(() => new RangeValue(n1, n2, "name1")).to.throw(↵ TypeError, "invalid range parameter"); </pre>
TU-12	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi parametri invertiti del range per il costruttore di <i>RangeValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: <i>minRange</i> e <i>maxRange</i> invertiti; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"maxRange is less then minRange"</i>. • Procedura: <pre> 1 expect(() => new RangeValue(70, 30, "name1")).to.throw(↵ TypeError, "maxRange is less then minRange"); </pre>

Test	Specifica
TU-13	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>RangeValue()</i> con gli stessi parametri di range; • Caso: <ul style="list-style-type: none"> – Input: <i>minRange</i> e <i>maxRange</i> uguali a 50; – Risultato atteso: stringa con nome dell'oggetto. • Procedura: <pre> 1 const name: string = "name1"; 2 const rangeV: RangeValue = new RangeValue(50, 50, name); 3 rangeV.getValueName(); 4 expect(rangeV.getValueName()).toEqual(name); </pre>
Scope	RangeValue::isValueType
TU-14	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>isValueType()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: True. • Procedura: <pre> 1 const rangeV = new RangeValue(30, 70, "name2"); 2 const result: boolean = rangeV.isValueType("50"); 3 expect(result).toEqual(true); </pre>
TU-15	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro il limite del range; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: True. • Procedura: <pre> 1 const rangeV = new RangeValue(30, 70, "name3"); 2 const result: boolean = rangeV.isValueType("70"); 3 expect(result).toEqual(true); </pre>

Test	Specifica
TU-16	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro casuale; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const rangeV = new RangeValue(30, 70, "name4"); 2 const result: boolean = rangeV.isValueType("brzcld"); 3 expect(result).to.equal(false); </pre>
TU-17	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro vuoto; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const rangeV = new RangeValue(30, 70, "name5"); 2 const result: boolean = rangeV.isValueType(""); 3 expect(result).to.equal(false); </pre>
TU-18	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro errato; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const rangeV = new RangeValue(30, 70, "name6"); 2 const result: boolean = rangeV.isValueType("20"); 3 expect(result).to.equal(false); </pre>

Test	Specifica
TU-19	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro non definito; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid string parameter"</i>.. ● Procedura: <pre> 1 const rangeV = new RangeValue(30, 70, "name7"); 2 let str: string; 3 expect(() => rangeV.isValueType(str)).to.throw(TypeError, "↔ invalid string parameter"); </pre>
TU-20	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> su un oggetto con gli stessi valori di range; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i> con <i>minRange</i> e <i>maxRange</i> uguali a 50; – Risultato atteso: True. ● Procedura: <pre> 1 const rangeV = new RangeValue(50, 50, "name8"); 2 const result: boolean = rangeV.isValueType("50"); 3 expect(result).to.equal(true); </pre>
Scope	RangeValue::getValueName
TU-21	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>getValueName()</i>; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>RangeValue</i>; – Risultato atteso: stringa con nome dell'oggetto. ● Procedura: <pre> 1 const name: string = "name9"; 2 const rangeV = new RangeValue(30, 70, name); 3 rangeV.isValueType("true"); 4 rangeV.isValueType("false"); 5 const result: string = rangeV.getValueName(); 6 expect(result).to.equal(name); </pre>
Scope	StringValue::constructor

Test	Specifica
TU-22	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi un valore come parametro non valido per il costruttore di <i>StringValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: stringa con value non valida; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid value parameter"</i>. • Procedura: <pre> 1 let str : string ; 2 expect (() => new StringValue("value1", str)).to.throw(↵ TypeError, "invalid value parameter"); </pre>
TU-23	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi un nome come parametro non valido per il costruttore di <i>StringValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: stringa con nome non valida; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid name parameter"</i>. • Procedura: <pre> 1 let str : string ; 2 expect (() => new StringValue(str, "name1")).to.throw(TypeError↵ , "invalid name parameter"); </pre>
Scope	StringValue::isValueType
TU-24	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>isValueType()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: True. • Procedura: <pre> 1 const stringV = new StringValue("value2", "name2"); 2 const result : boolean = stringV.isValueType("value2"); 3 expect(result).to.equal(true); </pre>

Test	Specifica
TU-25	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro errato; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const stringV = new StringValue("value3", "name3"); 2 const result: boolean = stringV.isValueType("true"); 3 expect(result).toEqual(false); </pre>
TU-26	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro casuale; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const stringV = new StringValue("value4", "name4"); 2 const result: boolean = stringV.isValueType("brzcld"); 3 expect(result).toEqual(false); </pre>
TU-27	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro vuoto; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const stringV = new StringValue("value4", "name4"); 2 const result: boolean = stringV.isValueType("brzcld"); 3 expect(result).toEqual(false); </pre>

Test	Specifica
TU-28	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro vuoto; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const stringV = new StringValue("value5", "name5"); 2 const result: boolean = stringV.isValueType(""); 3 expect(result).to.equal(false); </pre>
TU-29	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro corretto ma con la lettera maiuscola; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: False. ● Procedura: <pre> 1 const stringV = new StringValue("value6", "name6"); 2 const result: boolean = stringV.isValueType("VALUE6"); 3 expect(result).to.equal(false); </pre>
TU-30	<ul style="list-style-type: none"> ● Progettazione: verifica il funzionamento di <i>isValueType()</i> con parametro non definito; ● Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid parameter"</i>. ● Procedura: <pre> 1 const stringV = new StringValue("value7", "name7"); 2 let str: string; 3 expect(() => stringV.isValueType(str)).to.throw(TypeError, "↔ invalid parameter"); </pre>
Scope	StringValue::getValueName

Test	Specifica
TU-31	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueName()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>StringValue</i>; – Risultato atteso: stringa con nome dell'oggetto. • Procedura: <pre> 1 const name: string = "name8"; 2 const stringV = new StringValue("value8", name); 3 stringV.isValueType("true"); 4 stringV.isValueType("false"); 5 const result: string = stringV.getValueName(); 6 expect(result).toEqual(name); </pre>
Scope	ConcreteNodeAdapter::constructor
TU-32	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi parametri non validi per il costruttore di <i>ConcreteNodeAdapter()</i>; • Caso: <ul style="list-style-type: none"> – Input: parametro <i>node</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 let node: JNode; 2 const values: Array<AbstractValue> = new Array<AbstractValue>(); 3 expect(() => new ConcreteNodeAdapter(node, values)).toThrow(TypeError, "invalid node parameter"); </pre>

Test	Specifica
TU-33	<ul style="list-style-type: none"> • Progettazione: verifica che il sistema rilevi parametri non validi per il costruttore di <i>ConcreteNodeAdapter</i>(); • Caso: <ul style="list-style-type: none"> – Input: parametro <i>values</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid values parameter"</i>. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 let values: Array<AbstractValue>; 4 expect(() => new ConcreteNodeAdapter(n1, values)).to.throw(↵ TypeError, "invalid values parameter"); </pre>
Scope	ConcreteNodeAdapter::getSates
TU-34	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getStates</i>(); • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>ConcreteNodeAdapter</i> con stati [True, False]; – Risultato atteso: [True, False]. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNode = new ConcreteNodeAdapter(n1, new Array<↵ AbstractValue>()); 4 const states: Array<string> = concreteNode.getStates(); 5 expect(states.toLocaleString()).to.equal("true, false"); </pre>

Test	Specifica
TU-35	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>getStates()</code> controllando che venga ritornata una copia profonda dell'oggetto; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>ConcreteNodeAdapter</i> con stati [True, False]; – Risultato atteso: [True, False]. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNode = new ConcreteNodeAdapter(n1, new Array<↵ AbstractValue>()); 4 const states = concreteNode.getStates(); 5 states.reverse(); 6 expect(concreteNode.getStates().toLocaleString()).to.equal("↵ true, false"); </pre>
TU-36	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>getStates()</code> con stati di un nodo invertiti; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>ConcreteNodeAdapter</i> con stati [True, False]; – Risultato atteso: diverso da [False, True]. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNode = new ConcreteNodeAdapter(n1, new Array<↵ AbstractValue>()); 4 const states = concreteNode.getStates(); 5 expect(states.toLocaleString()).to.not.equal("false, true"); </pre>
Scope	ConcreteNodeAdapter::getValueName

Test	Specifica
TU-37	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueName()</i>; • Caso: <ul style="list-style-type: none"> – Input: array di <i>BoolValue</i>; – Risultato atteso: nome dei values. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNodeValues: Array<AbstractValue> = new Array<↵ AbstractValue>(); 4 concreteNodeValues.push(new BoolValue(false, "boolvalue-1")); 5 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 6 concreteNodeValues.push(new BoolValue(true, "boolvalue-3")); 7 const concreteNode = new ConcreteNodeAdapter(n1, ↵ concreteNodeValues); 8 const values: Array<AbstractValue> = concreteNode.getValues(); 9 expect(values[0].getValueName()).toEqual("boolvalue-1"); 10 expect(values[1].getValueName()).toEqual("boolvalue-2"); 11 expect(values[2].getValueName()).toEqual("boolvalue-3"); </pre>
TU-38	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueName()</i> controllando che venga ritornata una copia profonda dell'oggetto; • Caso: <ul style="list-style-type: none"> – Input: array di <i>BoolValue</i>; – Risultato atteso: nome dei values. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNodeValues: Array<AbstractValue> = new Array<↵ AbstractValue>(); 4 concreteNodeValues.push(new BoolValue(false, "boolvalue-1")); 5 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 6 concreteNodeValues.push(new BoolValue(true, "boolvalue-3")); 7 const concreteNode = new ConcreteNodeAdapter(n1, ↵ concreteNodeValues); 8 const values: Array<AbstractValue> = concreteNode.getValues(); 9 values.reverse(); 10 const newValues = concreteNode.getValues(); 11 expect(newValues[0].getValueName()).toEqual("boolvalue-1"); 12 expect(newValues[1].getValueName()).toEqual("boolvalue-2"); 13 expect(newValues[2].getValueName()).toEqual("boolvalue-3"); </pre>

Test	Specifica
TU-39	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueName()</i>; • Caso: <ul style="list-style-type: none"> – Input: array di <i>BoolValue</i>; – Risultato atteso: diverso da un nome predefinito. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNodeValues: Array<AbstractValue> = new Array<↵ AbstractValue>(); 4 concreteNodeValues.push(new BoolValue(false, "boolvalue-1")); 5 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 6 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 7 const concreteNode = new ConcreteNodeAdapter(n1, ↵ concreteNodeValues); 8 const values: Array<AbstractValue> = concreteNode.getValues(); 9 let str: string; 10 expect(values[0].getValueName()).to.not.equal("test"); 11 expect(values[1].getValueName()).to.not.equal(str); 12 expect(values[2].getValueName()).to.not.equal(""); </pre>
Scope	concreteNodeValues::findValue
TU-40	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>findValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: array di <i>BoolValue</i>;; – Risultato atteso: primo value con la stessa chiave di ricerca. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNodeValues: Array<AbstractValue> = new Array<↵ AbstractValue>(); 4 concreteNodeValues.push(new BoolValue(false, "boolvalue-1")); 5 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 6 concreteNodeValues.push(new BoolValue(true, "boolvalue-3")); 7 const concreteNode = new ConcreteNodeAdapter(n1, ↵ concreteNodeValues); 8 const value: AbstractValue = concreteNode.findValue("true"); 9 expect(value.getValueName()).to.equal("boolvalue-2"); </pre>

Test	Specifica
TU-41	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>findValue()</i> su un oggetto non presente nella lista; • Caso: <ul style="list-style-type: none"> – Input: array di <i>BoolValue</i>;; – Risultato atteso: NULL. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNodeValues: Array<AbstractValue> = new Array<↵ AbstractValue>(); 4 concreteNodeValues.push(new BoolValue(false, "boolvalue-1")); 5 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 6 concreteNodeValues.push(new BoolValue(true, "boolvalue-3")); 7 const concreteNode = new ConcreteNodeAdapter(n1, ↵ concreteNodeValues); 8 const value: AbstractValue = concreteNode.findValue("↵ valorechenonce"); 9 expect(value).to.equal(null); </pre>
TU-42	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>findValue()</i> su un oggetto non definito; • Caso: <ul style="list-style-type: none"> – Input: array di <i>BoolValue</i>;; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid parameter"</i>. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const concreteNodeValues: Array<AbstractValue> = new Array<↵ AbstractValue>(); 4 concreteNodeValues.push(new BoolValue(false, "boolvalue-1")); 5 concreteNodeValues.push(new BoolValue(true, "boolvalue-2")); 6 concreteNodeValues.push(new BoolValue(true, "boolvalue-3")); 7 const concreteNode = new ConcreteNodeAdapter(n1, ↵ concreteNodeValues); 8 let str: string; 9 expect(() => concreteNode.findValue(str)).to.throw(TypeError, ↵ "invalid parameter"); </pre>
Scope	ConcreteNetworkFactory::parseNetwork

Test	Specifica
TU-43	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> controllando il numero di nodi presenti; • Caso: <ul style="list-style-type: none"> – Input: file JSON con definizione di rete con due nodi; – Risultato atteso: 2. • Procedura: <pre> 1 const json = require("./CorrectNetwork.json"); 2 const jsonString: string = JSON.stringify(json); 3 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory().← .parseNetwork(jsonString); 4 expect(s.getNodeList().length).to.equal(2); </pre>
TU-44	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un input nullo; • Caso: <ul style="list-style-type: none"> – Input: NULL; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"JSON file content is null..."</i>. • Procedura: <pre> 1 let str: string; 2 expect(() => new ConcreteNetworkFactory().parseNetwork(str)).← to.throw(Error, "JSON file content is null..."); </pre>
TU-45	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON vuoto; • Caso: <ul style="list-style-type: none"> – Input: file JSON vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Bad Json Content!"</i>. • Procedura: <pre> 1 let str: string = ""; 2 expect(() => new ConcreteNetworkFactory().parseNetwork(str)).← to.throw(Error, "Bad Json Content!"); </pre>

Test	Specifica
TU-46	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON non valido; • Caso: <ul style="list-style-type: none"> – Input: file JSON non valido; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Bad Json Content!"</i>. • Procedura: <pre> 1 let json = require("./InvalidNetwork.json"); 2 const jsonString = JSON.stringify(json); 3 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, ""); </pre>
TU-47	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON che non rispetta la struttura di rete; • Caso: <ul style="list-style-type: none"> – Input: file JSON con un struttura di rete errata; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Bad Json Content!"</i>. • Procedura: <pre> 1 const jsonString = '{"nodes": {[]}}'; 2 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, "Bad Json Content!"); </pre>
TU-48	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con valori della CPT errati; • Caso: <ul style="list-style-type: none"> – Input: file JSON con valori della CPT errati; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"CPT error!"</i>. • Procedura: <pre> 1 let json = require("./IncorrectCpt.json"); 2 const jsonString = JSON.stringify(json); 3 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, "CPT error!"); </pre>

Test	Specifica
TU-49	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con range invertiti; • Caso: <ul style="list-style-type: none"> – Input: file JSON con range invertiti; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"maxRange is less then minRange"</i>. • Procedura: <pre> 1 let json = require("./InvertedMinMax.json"); 2 const jsonString = JSON.stringify(json); 3 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(TypeError, "maxRange is less then ↵ minRange"); </pre>
TU-50	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con valori dei nodi errati; • Caso: <ul style="list-style-type: none"> – Input: file JSON con valori dei nodi errati; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>TypeError</i> mostrando il messaggio <i>"invalid value parameter"</i>. • Procedura: <pre> 1 let json = require("./IncorrectType.json"); 2 const jsonString = JSON.stringify(json); 3 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(TypeError, "invalid value parameter"↵); </pre>

Test	Specifica
TU-51	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con un nodo padre non esistente; • Caso: <ul style="list-style-type: none"> – Input: file JSON con un nodo padre non esistente; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Node Parent not found in the network!"</i>. • Procedura: <pre> 1 let json = require("./IncorrectParent.json"); 2 const jsonString = JSON.stringify(json); 3 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, "Node Parent not found in the↵ network!"); </pre>
TU-52	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con un riferimento circolare diretto; • Caso: <ul style="list-style-type: none"> – Input: file JSON con un riferimento circolare diretto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Direct circular parenthood"</i>. • Procedura: <pre> 1 let json = require("./DCircularParenthood.json"); 2 const jsonString = JSON.stringify(json); 3 new ConcreteNetworkFactory().parseNetwork(jsonString); 4 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, "Direct circular parenthood")↵ ; </pre>

Test	Specifica
TU-53	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con un riferimento circolare indiretto; • Caso: <ul style="list-style-type: none"> – Input: file JSON con un riferimento circolare indiretto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Indirect circular parenthood"</i>. • Procedura: <pre> 1 let json = require("./ICircularParenthood.json"); 2 const jsonString = JSON.stringify(json); 3 new ConcreteNetworkFactory().parseNetwork(jsonString); 4 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, "Indirect circular parenthood↵ "); </pre>
TU-54	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con due nodi dallo stesso nome; • Caso: <ul style="list-style-type: none"> – Input: file JSON con un nodo dallo stesso nome di un altro nodo; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"The node ExampleNode2 already exist in the network!"</i>. • Procedura: <pre> 1 let json = require("./IncorrectName.json"); 2 const jsonString = JSON.stringify(json); 3 expect(() => new ConcreteNetworkFactory().parseNetwork(↵ jsonString)).to.throw(Error, "The node ExampleNode2 ↵ already exist in the network!"); </pre>

Test	Specifica
TU-55	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con probabilità della CPT non valide; • Caso: <ul style="list-style-type: none"> – Input: file JSON con probabilità CPT non valide; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Incorrect CPT probabilities"</i>. • Procedura: <pre> 1 expect (() => new ConcreteNetworkFactory().parseNetwork(↵ IncorrectCptJsonString, jsonSchemaString)).to.throw(Error, ↵ "Incorrect CPT probabilities"); </pre>
TU-56	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON con probabilità della CPT vuote; • Caso: <ul style="list-style-type: none"> – Input: file JSON con probabilità CPT vuote; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Empty CPT"</i>. • Procedura: <pre> 1 expect (() => new ConcreteNetworkFactory().parseNetwork(↵ EmptyCptJsonString, jsonSchemaString)).to.throw(Error, "↵ Empty CPT"); </pre>
TU-57	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON le righe della CPT errate; • Caso: <ul style="list-style-type: none"> – Input: file JSON con righe CPT errate; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Incorrect CPT number of rows"</i>. • Procedura: <pre> 1 expect (() => new ConcreteNetworkFactory().parseNetwork(↵ IncorrectCptRowsJsonString, jsonSchemaString)).to.throw(↵ Error, "Incorrect CPT number of rows"); </pre>

Test	Specifica
TU-58	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>ConcreteNetworkFactory()</i> con un JSON le colonne della CPT errate; • Caso: <ul style="list-style-type: none"> – Input: file JSON con colonne CPT errate; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Incorrect CPT number of columns"</i>. • Procedura: <pre> 1 expect (() => new ConcreteNetworkFactory().parseNetwork(↵ IncorrectCptColumnsJsonString, jsonSchemaString)).to.throw↵ (Error, "Incorrect CPT number of columns"); </pre>
Scope	ConcreteNetworkAdapter::constructor
TU-59	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>ConcreteNetworkAdapter()</i> con parametri non validi; • Caso: <ul style="list-style-type: none"> – Input: parametro <i>graph</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid graph parameter"</i>. • Procedura: <pre> 1 let graph: JGraph; 2 let list: Array<NodeAdapter> = new Array<NodeAdapter>(); 3 expect (() => new ConcreteNetworkAdapter(graph, list)).to.throw↵ (Error, "invalid graph parameter"); </pre>
TU-60	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>ConcreteNetworkAdapter()</i> con parametri non validi; • Caso: <ul style="list-style-type: none"> – Input: parametro <i>list</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid list parameter"</i>. • Procedura: <pre> 1 let graph: JGraph = jsbayes.newGraph(); 2 let list: Array<NodeAdapter>; 3 expect (() => new ConcreteNetworkAdapter(graph, list)).to.throw↵ (Error, "invalid list parameter"); </pre>

Test	Specifica
TU-61	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>ConcreteNetworkAdapter()</i> con parametri non validi; • Caso: <ul style="list-style-type: none"> – Input: parametro <i>list</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid list parameter"</i>. • Procedura: <pre> 1 let graph: JGraph = jsbayes.newGraph(); 2 let list: Array<NodeAdapter> = new Array<NodeAdapter>(); 3 expect(() => new ConcreteNetworkAdapter(graph, list)).to.throw((Error, "invalid list parameter"); </pre>
Scope	ConcreteNetworkAdapter::observeNode
TU-62	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>observeNode()</i>; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido; – Risultato atteso: TRUE.
TU-63	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>observeNode()</i> quando il nome di un nodo non è definito; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory(). parseNetwork(correctJsonString, jsonSchemaString); 2 expect(()=>network.observeNode("", "Example of string value")) .to.throw(Error, "invalid node parameter"); </pre>

Test	Specifica
TU-64	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>observeNode()</i> quando il nome di un nodo non è presente; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo errato; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Node FakeNode is not present in the network"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory().↵ parseNetwork(correctJsonString, jsonSchemaString); 2 expect(()=>network.observeNode("FakeNode", "Example of string ↵ value")).to.throw(Error, "Node FakeNode is not present in ↵ the network"); </pre>
TU-65	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>observeNode()</i> quando il valore di un nodo non è presente; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, valore nodo errato; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Node Example has not a value called FakeValue"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory().↵ parseNetwork(correctJsonString, jsonSchemaString); 2 expect(()=>network.observeNode("Example", "FakeValue")).to.↵ throw(Error, "Node Example has not a value called ↵ FakeValue"); </pre>
Scope	ConcreteNetworkAdapter::unobserveNode
TU-66	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>unobserveNode()</i>; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido; – Risultato atteso: TRUE.

Test	Specifica
TU-67	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>unobserveNode()</i> quando il nome del nodo non è definito; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory().↵ parseNetwork(correctJsonString, jsonSchemaString); 2 let name: string; 3 expect(()=>network.unobserveNode(name)).to.throw(Error, "↵ invalid node parameter"); </pre>
TU-68	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>unobserveNode()</i> quando il nome del nodo non è vuoto; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory().↵ parseNetwork(correctJsonString, jsonSchemaString); 2 expect(()=>network.unobserveNode("")).to.throw(Error, "invalid↵ node parameter"); </pre>
TU-69	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>unobserveNode()</i> quando il nome del nodo non è vuoto; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory↵ ().parseNetwork(correctJsonString, jsonSchemaString); 2 expect(()=>network.unobserveNode("")).to.throw(Error, "↵ invalid node parameter"); </pre>

Test	Specifica
TU-70	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>unobserveNode()</i> quando il nome del nodo non è presente; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo errato; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Node FakeNode is not present in the network"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory().↵ parseNetwork(correctJsonString, jsonSchemaString); 2 expect(()=>network.unobserveNode("FakeNode")).to.throw(Error, ↵ "Node FakeNode is not present in the network"); </pre>
Scope	ConcreteNetworkAdapter::sampleNetwork
TU-71	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>sampleNetwork()</i> quando il numero non è definito; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, numero non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid number parameter"</i>. • Procedura: <pre> 1 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory().↵ .parseNetwork(correctJsonString, jsonSchemaString); 2 let undNumber: number; 3 expect(()=>s.sampleNetwork(undNumber)).to.throw(Error, "↵ invalid number parameter"); </pre>

Test	Specifica
TU-72	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>sampleNetwork()</i> quando il numero è minore di zero; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, numero < 0; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid number parameter"</i>. • Procedura: <pre> 1 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()↵ .parseNetwork(correctJsonString, jsonSchemaString); 2 let negNumber: number = -50; 3 expect(()=>s.sampleNetwork(negNumber)).to.throw(Error, "↵ invalid number parameter"); </pre>
Scope	ConcreteNetworkAdapter::getNodeProbs
TU-73	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getNodeProbs()</i> quando il nome del nodo non è definito; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()↵ .parseNetwork(correctJsonString, jsonSchemaString); 2 let undName: string; 3 expect(()=>s.getNodeProbs(undName)).to.throw(Error, "invalid ↵ node parameter"); </pre>

Test	Specifica
TU-74	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>getNodeProbs()</code> quando il nome del nodo è vuoto; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory() .parseNetwork(correctJsonString, jsonSchemaString); 2 expect(() => s.getNodeProbs("")).toThrow(Error, "invalid node parameter"); </pre>
TU-75	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>getNodeProbs()</code> quando il nome del nodo è presente; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido, nome nodo errato; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Node IncorrectName is not present in the network"</i>. • Procedura: <pre> 1 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory() .parseNetwork(correctJsonString, jsonSchemaString); 2 expect(() => s.getNodeProbs("IncorrectName")).toThrow(Error, " Node IncorrectName is not present in the network"); </pre>
Scope	ConcreteNetworkAdapter::getNodeList
TU-76	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>getNodeList()</code>; • Caso: <ul style="list-style-type: none"> – Input: file JSON valido; – Risultato atteso: <i>"Example2"</i>. • Procedura: <pre> 1 const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory() .parseNetwork(correctJsonString, jsonSchemaString); 2 const nodeOne = s.getNodeList()[1]; 3 expect(nodeOne.getName()).toEqual("Example2"); </pre>
Scope	InputFlow::constructor

Test	Specifica
TU-77	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxInputFlow()</i> quando <i>database</i> non è definito; • Caso: <ul style="list-style-type: none"> – Input: <i>database</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid db parameter"</i>. • Procedura: <pre> 1 let db: string; 2 expect(()=> 3 new InfluxInputFlow(db, "query", 4 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "9957", ["user", "password"])) 5).to.throw(Error, "invalid db parameter"); </pre>
TU-78	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxInputFlow()</i> quando <i>query</i> non è definito; • Caso: <ul style="list-style-type: none"> – Input: <i>query</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid query parameter"</i>. • Procedura: <pre> 1 let query: string; 2 expect(()=> 3 new InfluxInputFlow("db", query, 4 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "9957", ["user", "password"])) 5).to.throw(Error, "invalid query parameter"); </pre>

Test	Specifica
TU-79	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxInputFlow()</i> quando <i>client</i> non è definito; • Caso: <ul style="list-style-type: none"> – Input: <i>client</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid client parameter"</i>. • Procedura: <pre> 1 let readClient: ReadClient; 2 expect (()=> 3 new InfluxInputFlow("db", "query", readClient) 4).to.throw(Error, "invalid client parameter"); </pre>
TU-80	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxInputFlow()</i> quando <i>database</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>database</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid db parameter"</i>. • Procedura: <pre> 1 expect (()=> 2 new InfluxInputFlow("", "query", 3 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "9957", ["user", "password"])) 4).to.throw(Error, "invalid db parameter"); </pre>

Test	Specifica
TU-81	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxInputFlow()</i> quando <i>query</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>query</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid query parameter"</i>. • Procedura: <pre> 1 expect (()=> 2 new InfluxInputFlow("db", "", 3 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "9957", ["user", "password"])) 4).to.throw(Error, "invalid query parameter"); </pre>
TU-82	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxInputFlow()</i>; • Caso: <ul style="list-style-type: none"> – Input: parametri corretti; – Risultato atteso: nessuna eccezione viene lanciata. • Procedura: <pre> 1 expect (()=> 2 new InfluxInputFlow("db", "query", 3 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "9957", ["user", "password"])) 4).to.not.throw(Error); </pre>
Scope	InfluxInputFlow::getResult

Test	Specifica
TU-83	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getResult()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InfluxInputFlow</i>; – Risultato atteso: viene ritornato il risultato. • Procedura: <pre> 1 const influxInput: InfluxInputFlow = new InfluxInputFlow ("↵ prova", "SELECT field1 FROM burglary", 2 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "8086", ["root", "root"])); 3 influxInput.getResult().then(function(result){ 4 expect(result.toString()).to.not.equal(null); </pre>
TU-84	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getResult()</i> quando c'è un errore nella query; • Caso: <ul style="list-style-type: none"> – Input: query errata; – Risultato atteso: il sistema mostra il messaggio <i>"Error: Query"</i>. • Procedura: <pre> 1 const influxInput: InfluxInputFlow = new InfluxInputFlow ("↵ prova", "SELECT field FROM measurement", 2 new ConcreteReadClientFactory().makeInfluxReadClient("http://↵ localhost", "8086", ["user", "password"])); 3 influxInput.getResult().then(function(){}).catch(function(e){ 4 expect(e.toString()).to.contain("Error: Query"); 5 }); </pre>
Scope	Datasource::constructor

Test	Specifica
TU-85	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>Datasource()</i> quando <i>url</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>url</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid url parameter</i>". • Procedura: <pre> 1 expect (()=> 2 expect (() => new DataSource("", "database")).to.throw(Error, "↔ invalid url parameter"); </pre>
TU-86	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>Datasource()</i> quando <i>database</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>database</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid db parameter</i>". • Procedura: <pre> 1 expect (() => new DataSource("http://localhost", "").to.throw(↔ Error, "invalid db parameter"); </pre>
TU-87	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>Datasource()</i> quando <i>username</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>username</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid username parameter</i>". • Procedura: <pre> 1 expect (() => new DataSource("http://localhost", "database", ""↔)).to.throw(Error, "invalid username parameter"); </pre>

Test	Specifica
TU-88	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>DataSource()</i> quando <i>password</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>password</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid password parameter"</i>. • Procedura: <pre>1 expect (() => new DataSource("http://localhost", "database", "↵ username", "")) .to.throw(Error, "invalid password ↵ parameter");</pre>
TU-89	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>DataSource()</i> quando <i>type</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>type</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid type parameter"</i>. • Procedura: <pre>1 expect (() => new DataSource("http://localhost", "database", "↵ username", "password", "")) .to.throw(Error, "invalid type ↵ parameter");</pre>
TU-90	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>DataSource()</i> quando <i>name</i> è vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>name</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid name parameter"</i>. • Procedura: <pre>1 expect (() => new DataSource("http://localhost", "database", "↵ username", "password", "type", "")) .to.throw(Error, "↵ invalid name parameter");</pre>
Scope	NetReader::constructor

Test	Specifica
TU-91	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>NetReader()</i> con parametro non definito; • Caso: <ul style="list-style-type: none"> – Input: parametro non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid parameter"</i>. • Procedura: <pre> 1 let networkAdapter : ConcreteNetworkAdapter ; 2 expect (()=> new NetReader(networkAdapter)).to.throw(Error , "↔ invalid parameter"); </pre>
Scope	NetReader::read
TU-92	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>read()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>NetReader</i>; – Risultato atteso: "Example". • Procedura: <pre> 1 const networkReader : NetReader = new NetReader(network); 2 networkReader.connectNode("Example", new DataSource("http://↔ localhost:8086/"), "SELECT Percent_DPC_Time FROM win_cpu")↔ ; 3 networkReader.read().then(function(result){ 4 expect(result.buildIterator().next().value.getNode().getName()↔).to.equal("Example"); </pre>
Scope	NetReader::connectNode

Test	Specifica
TU-93	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>connectNode()</i> con <i>node</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>node</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid node parameter"</i>. • Procedura: <pre> 1 let node: string; 2 expect(() => networkReader.connectNode(node, new DataSource("↔ http://localhost"), "query")).to.throw(Error, "invalid ↔ node parameter"); </pre>
TU-94	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>connectNode()</i> con <i>datasource</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>datasource</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid datasource parameter"</i>. • Procedura: <pre> 1 let dataS: DataSource; 2 expect(() => networkReader.connectNode("node", dataS, "query")↔).to.throw(Error, "invalid datasource parameter"); </pre>
TU-95	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>connectNode()</i> con <i>query</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>query</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid query parameter"</i>. • Procedura: <pre> 1 let query: string; 2 expect(() => networkReader.connectNode("node", new DataSource(↔ "http://localhost"), query)).to.throw(Error, "invalid ↔ query parameter"); </pre>

Test	Specifica
TU-96	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>connectNode()</code> con <code>node</code> vuoto; • Caso: <ul style="list-style-type: none"> – Input: <code>node</code> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <code>Error</code> mostrando il messaggio <code>"invalid node parameter"</code>. • Procedura: <pre> 1 expect (() => networkReader.connectNode("", new DataSource("↵ http://localhost"), "query")).to.throw(Error, "invalid ↵ node parameter"); </pre>
TU-97	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>connectNode()</code> con <code>query</code> vuoto; • Caso: <ul style="list-style-type: none"> – Input: <code>query</code> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <code>Error</code> mostrando il messaggio <code>"invalid query parameter"</code>. • Procedura: <pre> 1 expect (() => networkReader.connectNode("node", new DataSource(↵ "http://localhost"), "")).to.throw(Error, "invalid query ↵ parameter"); </pre>
Scope	ReusableReadClientPool::acquireReusable
TU-98	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <code>acquireReusable()</code> con <code>datasource</code> non definito; • Caso: <ul style="list-style-type: none"> – Input: <code>datasource</code> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <code>Error</code> mostrando il messaggio <code>"invalid datasource parameter"</code>. • Procedura: <pre> 1 let datasource: DataSource; 2 expect (() => ReusableReadClientPool.getInstance().↵ acquireReusable(datasource)).to.throw(Error, "invalid ↵ datasource parameter"); </pre>

Test	Specifica
TU-99	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>acquireReusable()</i> con <i>datasource</i> definito; • Caso: <ul style="list-style-type: none"> – Input: <i>datasource</i> definito; – Risultato atteso: TRUE. • Procedura: <pre> 1 let datasource: DataSource = new DataSource("http://localhost/↵ "); 2 let JSONstr: string = JSON.stringify(ReusableReadClientPool.↵ getInstance().acquireReusable(datasource)); 3 expect(JSONstr.includes("http://localhost/").to.equal(true); </pre>
TU-100	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>acquireReusable()</i> con <i>to_remove</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>to_remove</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid to_remove parameter</i>". • Procedura: <pre> 1 let to_remove: ReadClient; 2 expect(() => ReusableReadClientPool.getInstance().↵ releseReusable(to_remove)).to.throw(Error, "invalid ↵ to_remove parameter"); </pre>
Scope	CalcResultAggregate::constructor
TU-101	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResultAggregate()</i> con <i>collection</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>collection</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid collection parameter</i>". • Procedura: <pre> 1 let collection: Array<CalcResult>; 2 expect(()=> new CalcResultAggregate(collection)).to.throw(↵ Error, "invalid collection parameter"); </pre>

Test	Specifica
TU-102	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResultAggregate()</i> con <i>collection</i> vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>collection</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid collection parameter"</i>. • Procedura: <pre> 1 let collection : Array<CalcResult> = new Array<CalcResult>(); 2 expect(() => new CalcResultAggregate(collection)).to.throw(↵ Error, "invalid collection parameter"); </pre>
Scope	CalcResultAggregate::createIterator
TU-103	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>createIterator()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>CalcResultItem</i>, oggetto <i>CalcResult</i>; – Risultato atteso: "nodo". • Procedura: <pre> 1 let collection : Array<CalcResult> = new Array<CalcResult>(); 2 let itemsArray : Array<CalcResultItem> = new Array<↵ CalcResultItem>(); 3 itemsArray.push(new CalcResultItem("stringa", 1)); 4 collection.push(new CalcResult("nodo", itemsArray)); 5 expect(new CalcResultAggregate(collection).createIterator().↵ next().value.getNodeName()).to.equal("nodo"); </pre>
Scope	CalcResultItem::constructor
TU-104	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResultItem()</i> con parametri nulli; • Caso: <ul style="list-style-type: none"> – Input: parametri nulli; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid null parameter"</i>. • Procedura: <pre> 1 expect(() => new CalcResultItem(null, null)).to.throw(Error, "↵ invalid null parameter"); </pre>

Test	Specifica
TU-105	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResultItem()</i> con probabilità negativa; • Caso: <ul style="list-style-type: none"> – Input: probabilità < 0; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid probability parameter"</i>. • Procedura: <pre>1 expect (() => new CalcResultItem("n1",-5)).to.throw(Error, "↔ invalid probability parameter");</pre>
TU-106	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResultItem()</i> con probabilità maggiore di uno; • Caso: <ul style="list-style-type: none"> – Input: probabilità > 1; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid probability parameter"</i>. • Procedura: <pre>1 expect (() => new CalcResultItem("n1",1.1)).to.throw(Error, "↔ invalid parameter");</pre>
Scope	CalcResultItem::getValueName
TU-107	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueName()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>CalcResultItem</i>; – Risultato atteso: "n1". • Procedura: <pre>1 expect(new CalcResultItem("n1",0.5).getValueName()).to.equal("↔ n1");</pre>
Scope	CalcResultItem::getProbValue

Test	Specifica
TU-108	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getProbValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>CalcResultItem</i>; – Risultato atteso: "0.5". • Procedura: <pre> 1 expect (new CalcResultItem ("n1", 0.5).getProbValue()).to.equal((0.5) ; </pre>
Scope	CalcResult::constructor
TU-109	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResult()</i> con parametri nulli; • Caso: <ul style="list-style-type: none"> – Input: parametri nulli; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "invalid null parameter". • Procedura: <pre> 1 expect (() => new CalcResult (null , null)).to.throw (Error , "↵ invalid null parameter"); </pre>
TU-110	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResult()</i> con item nulli in array; • Caso: <ul style="list-style-type: none"> – Input: item nulli in array; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "invalid array parameter". • Procedura: <pre> 1 let calcItems : Array<CalcResultItem>=new Array<CalcResultItem ↵ >(); 2 calcItems.push(new CalcResultItem ("1", 0.5)); 3 calcItems.push(new CalcResultItem ("2", 0.6)); 4 calcItems.push(new CalcResultItem ("3", 0.3)); 5 calcItems.push(null); 6 expect (() => new CalcResult (null , calcItems)).to.throw (Error , "↵ invalid array parameter"); </pre>

Test	Specifica
TU-111	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>CalcResult()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>CalcResult</i>; – Risultato atteso: nessuna eccezione viene lanciata. • Procedura: <pre> 1 let calcItems:Array<CalcResultItem>=new Array<CalcResultItem < >(); 2 calcItems.push(new CalcResultItem("1",0.5)); 3 expect(() => new CalcResult("n1",calcItems)).to.not.throw(< Error); </pre>
Scope	CalcResult::getNodeName
TU-112	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getNodeName()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>CalcResultItem</i>; – Risultato atteso: "n1". • Procedura: <pre> 1 let calcItems:Array<CalcResultItem>=new Array<CalcResultItem < >(); 2 calcItems.push(new CalcResultItem("1",0.5)); 3 expect(new CalcResult("n1",calcItems).getNodeName()).to.equal(< "n1"); </pre>
Scope	CalcResult::getValueProbs

Test	Specifica
TU-113	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getValueProbs()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>calcResult</i>; – Risultato atteso: ["n5", 0.5], ["n3", 0.6]. • Procedura: <pre> 1 let calcItems: Array<CalcResultItem>=new Array<CalcResultItem>← >(); 2 calcItems.push(new CalcResultItem("n5",0.5)); 3 calcItems.push(new CalcResultItem("n3",0.6)); 4 let calcResult=new CalcResult("n1",calcItems); 5 let probs=calcResult.getValueProbs(); 6 expect(probs[0].getValueName()).to.equals("n5"); 7 expect(probs[0].getProbValue()).to.equals(0.5); 8 expect(probs[1].getValueName()).to.equals("n3"); 9 expect(probs[1].getProbValue()).to.equals(0.6); </pre>
Scope	InputResultAggregate::constructor
TU-114	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InputResultAggregate()</i> con <i>collection</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>collection</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid collection parameter"</i>. • Procedura: <pre> 1 let collection: Array<InputResult>; 2 expect(() => new InputResultAggregate(collection)).to.throw(← Error, "invalid collection parameter"); </pre>
Scope	InputResultAggregate::buildIterator

Test	Specifica
TU-115	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>buildIterator()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InputResult</i>; – Risultato atteso: "valore". • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const values: Array<AbstractValue> = new Array<AbstractValue>() >(); 4 values.push(new BoolValue(true, "prova")); 5 const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(n1, values); 6 const currentValue: string = "valore"; 7 const collection: Array<InputResult> = new Array<InputResult>() >(); 8 collection.push(new InputResult(nodeName, currentValue)); 9 expect(new InputResultAggregate(collection).buildIterator(). next().value.getCurrentValue()).toEqual("valore"); </pre>
Scope	InputResult::constructor
TU-116	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InputResult()</i> con <i>nodeName</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>nodeName</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid nodeName parameter"</i>. • Procedura: <pre> 1 let nodeName: ConcreteNodeAdapter; 2 expect(() => new InputResult(nodeName, "valore")).toThrow(Error, "invalid nodeName parameter"); </pre>

Test	Specifica
TU-117	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InputResult()</i> con <i>currentValue</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>currentValue</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid currentValue parameter"</i>. • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const values: Array<AbstractValue> = new Array<AbstractValue>() 4 values.push(new BoolValue(true, "prova")); 5 const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(n1, values); 6 let currentValue: string; 7 expect(() => new InputResult(nodeName, currentValue)).to.throw(Error, "invalid currentValue parameter"); </pre>
Scope	InputResult::getNode
TU-118	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getNode()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InputResult</i>; – Risultato atteso: "n1". • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const values: Array<AbstractValue> = new Array<AbstractValue>() 4 values.push(new BoolValue(true, "prova")); 5 const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(n1, values); 6 const currentValue: string = "valore"; 7 const name: string = new InputResult(nodeName, currentValue).getNode().getName(); 8 expect(name).to.equal("n1"); </pre>
Scope	InputResult::getCurrentValue

Test	Specifica
TU-119	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getCurrentValue()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InputResult</i>; – Risultato atteso: "valore". • Procedura: <pre> 1 const graph: JGraph = jsbayes.newGraph(); 2 const n1: JNode = graph.addNode("n1", ["true", "false"]); 3 const values: Array<AbstractValue> = new Array<AbstractValue>() >(); 4 values.push(new BoolValue(true, "prova")); 5 const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(n1, values); 6 const currentValue: string = "valore"; 7 const getCurrent: string = new InputResult(nodeName, currentValue).getCurrentValue(); 8 expect(getCurrent).to.equal("valore"); </pre>
Scope	NetUpdater::constructor
TU-120	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>NetUpdater()</i> con <i>network</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>network</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "invalid network parameter". • Procedura: <pre> 1 let network: NetworkAdapter; 2 expect(() => new NetUpdater(network)).to.throw(Error, "invalid network parameter"); </pre>
Scope	NetUpdater::updateNet

Test	Specifica
TU-121	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>updateNet()</i> con <i>InputResultAggregate</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>InputResultAggregate</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid InputResultAggregate parameter"</i>. • Procedura: <pre> 1 const network: NetworkAdapter = new ConcreteNetworkFactory().↵ parseNetwork(jsonString, jsonSchemaString); 2 let results: InputResultAggregate; 3 const networkUpdater: NetUpdater = new NetUpdater(network); 4 expect(() => networkUpdater.updateNet(results)).to.throw(Error, ↵ "invalid InputResultAggregate parameter"); </pre>
Scope	SingleNetWriter::constructor
TU-122	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>SingleNetWriter()</i> con <i>client</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>client</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid client parameter"</i>. • Procedura: <pre> 1 let client: WriteClient; 2 expect(() => new SingleNetWriter(client)).to.throw(Error, "↵ invalid client parameter"); </pre>

Test	Specifica
TU-123	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>SingleNetWriter()</i> con <i>client</i> definito; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>SingleNetWriter</i>; – Risultato atteso: diverso da "null". • Procedura: <pre> 1 new ConcreteWriteClientFactory().makeInfluxWriteClient(2 "http://localhost", "8086", "prova", ["root", "root"] 3).then(function(writeClient){ 4 const singleWriter: SingleNetWriter = new SingleNetWriter(↵ 5 writeClient); 6 expect(singleWriter).to.not.equal(null); 7 }); </pre>
Scope	SingleNetWriter::write
TU-124	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>write()</i> con <i>calcData</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>calcData</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid calcData parameter</i>". • Procedura: <pre> 1 const idb: InfluxDB = new InfluxDB(); 2 const client: WriteClient = new InfluxWriteClient("", "", idb)↵ 3 ; 4 const singleWriter: SingleNetWriter = new SingleNetWriter(↵ 5 client); 6 let calc: CalcResultAggregate; 7 singleWriter.write(calc).then(function () {}) 8 .catch(function (e){ 9 expect(<Error> e.toString()).to.equal("Error: invalid ↵ 10 calcData parameter"); 11 }); </pre>
Scope	NetManager::constructor

Test	Specifica
TU-125	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>NetManager()</i> con <i>reader</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>reader</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid reader parameter</i>". • Procedura: <pre> 1 let unReader: NetReader; 2 expect (() => new NetManager(unReader, updater, writer)).to.↔ throw(Error, "invalid reader parameter"); </pre>
TU-126	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>NetManager()</i> con <i>updater</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>updater</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid updater parameter</i>". • Procedura: <pre> 1 let unUpdater: NetUpdater; 2 expect (() => new NetManager(reader, unUpdater, writer)).to.↔ throw(Error, "invalid updater parameter"); </pre>
TU-127	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>NetManager()</i> con <i>writer</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>writer</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid writer parameter</i>". • Procedura: <pre> 1 let unWriter: NetWriter; 2 expect (() => new NetManager(reader, updater, unWriter)).to.↔ throw(Error, "invalid writer parameter"); </pre>
Scope	NetManager::updateNet

Test	Specifica
TU-128	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>updateNet()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>NetManager</i>; – Risultato atteso: TRUE.
Scope	InfluxWriteClient::constructor
TU-129	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxWriteClient()</i> con <i>dsn</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>dsn</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid dsn parameter</i>". • Procedura: <pre> 1 let dsn: string; 2 expect(() => new InfluxWriteClient(dsn, "prova", influx)).to.↵ throw(Error, "invalid dsn parameter"); </pre>
TU-130	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxWriteClient()</i> con <i>defaultDB</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>defaultDB</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>invalid defaultDB parameter</i>". • Procedura: <pre> 1 let defaultDB: string; 2 expect(() => new InfluxWriteClient("http://localhost:8086/", ↵ defaultDB, influx)).to.throw(Error, "invalid defaultDB ↵ parameter"); </pre>

Test	Specifica
TU-131	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>InfluxWriteClient()</i> con <i>influx</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>influx</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"invalid influx parameter"</i>. • Procedura: <pre> 1 let unInflux : InfluxDB ; 2 expect (() => new InfluxWriteClient ("http://localhost:8086/", "← prova", unInflux)).to.throw(Error, "invalid influx ← parameter"); </pre>
Scope	InfluxWriteClient::getAddress
TU-132	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getAddress()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InfluxWriteClient</i>; – Risultato atteso: <i>"http://localhost:8086/"</i>. • Procedura: <pre> 1 expect(new InfluxWriteClient ("http://localhost:8086/", "prova"← , influx).getAddress()).to.equal("http://localhost:8086/")← ; </pre>
Scope	InfluxWriteClient::getDefaultDB
TU-133	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>getDefaultDB()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InfluxWriteClient</i>; – Risultato atteso: <i>"prova"</i>. • Procedura: <pre> 1 expect(new InfluxWriteClient ("http://localhost:8086/", "prova"← , influx).getDefaultDB()).to.equal("prova"); </pre>
Scope	InfluxWriteClient::writeBatchData

Test	Specifica
TU-134	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>writeBatchData()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InfluxWriteClient</i>; – Risultato atteso: TRUE. • Procedura: <pre> 1 const influxWriter: InfluxWriteClient = new InfluxWriteClient(↵ "http://localhost:8086/", "prova", influx); 2 let calcArray: Array<CalcResult> = new Array<CalcResult>(); 3 let calcItemArray: Array<CalcResultItem> = new Array<↵ CalcResultItem>(); 4 calcItemArray.push(new CalcResultItem("field1", 0.5)); 5 calcArray.push(new CalcResult("burglary", calcItemArray)); 6 let toBeWritten: CalcResultAggregate = new CalcResultAggregate↵ (calcArray); 7 influxWriter.writeBatchData(toBeWritten).then(function(){ 8 expect(true).to.equal(true); 9 }); </pre>
Scope	InfluxWriteClient::writePointData
TU-135	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento di <i>writePointData()</i>; • Caso: <ul style="list-style-type: none"> – Input: oggetto <i>InfluxWriteClient</i>; – Risultato atteso: TRUE. • Procedura: <pre> 1 const influxWriter: InfluxWriteClient = new InfluxWriteClient(↵ "http://localhost:8086/", "prova", influx); 2 let calcItemArray: Array<CalcResultItem> = new Array<↵ CalcResultItem>(); 3 calcItemArray.push(new CalcResultItem("field1", 0.5)); 4 influxWriter.writePointData(new CalcResult("burglary", ↵ calcItemArray)).then(function(){ 5 expect(true).to.equal(true); 6 }); </pre>
Scope	WriteClientFactory::constructor

Test	Specifica
TU-136	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>WriteClientFactory()</i> con <i>host</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>host</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>Error: invalid host parameter</i>". • Procedura: <pre> 1 let host: string; 2 new ConcreteWriteClientFactory().makeInfluxWriteClient(3 host, "something", "something else", ["admin", "password"] 4).then(function(){}).catch(function(e){ 5 expect(<Error> e.toString()).toEqual("Error: invalid host ↵ 6 parameter"); 7 }); </pre>
TU-137	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>WriteClientFactory()</i> con <i>port</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>port</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>Error: invalid port parameter</i>". • Procedura: <pre> 1 let port: string; 2 new ConcreteWriteClientFactory().makeInfluxWriteClient(3 "something", port, "something else", ["admin", "password"] 4).then(function(){}).catch(function(e){ 5 expect(<Error> e.toString()).toEqual("Error: invalid port ↵ 6 parameter"); 7 }); </pre>

Test	Specifica
TU-138	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>WriteClientFactory()</i> con <i>defaultDB</i> non definito; • Caso: <ul style="list-style-type: none"> – Input: <i>defaultDB</i> non definito; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>Error: invalid defaultDB parameter</i>". • Procedura: <pre> 1 let defaultDB: string; 2 new ConcreteWriteClientFactory().makeInfluxWriteClient(3 "something", "something else", defaultDB, ["admin", "password"] 4).then(function(){}).catch(function(e){ 5 expect(<Error> e.toString()).to.equal("Error: invalid defaultDB parameter"); 6 }); </pre>
TU-139	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>WriteClientFactory()</i> con <i>host</i> vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>host</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio "<i>Error: invalid host parameter</i>". • Procedura: <pre> 1 new ConcreteWriteClientFactory().makeInfluxWriteClient(2 "", "something", "something else", ["admin", "password"] 3).then(function(){}).catch(function(e){ 4 expect(<Error> e.toString()).to.equal("Error: invalid host parameter"); 5 }); </pre>

Test	Specifica
TU-140	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>WriteClientFactory()</i> con <i>port</i> vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>port</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Error: invalid port parameter"</i>. • Procedura: <pre> 1 new ConcreteWriteClientFactory().makeInfluxWriteClient(2 "something", "", "something else", ["admin", "password"] 3).then(function(){}).catch(function(e){ 4 expect(<Error> e.toString()).toEqual("Error: invalid port ↵ parameter"); 5 }); </pre>
TU-141	<ul style="list-style-type: none"> • Progettazione: verifica il funzionamento del costruttore di <i>WriteClientFactory()</i> con <i>defaultDB</i> vuoto; • Caso: <ul style="list-style-type: none"> – Input: <i>defaultDB</i> vuoto; – Risultato atteso: il sistema lancia un'eccezione di tipo <i>Error</i> mostrando il messaggio <i>"Error: invalid defaultDB parameter"</i>. • Procedura: <pre> 1 new ConcreteWriteClientFactory().makeInfluxWriteClient(2 "something", "something else", "", ["admin", "password"] 3).then(function(){}).catch(function(e){ 4 expect(<Error> e.toString()).toEqual("Error: invalid ↵ defaultDB parameter"); 5 }); </pre>

Tabella 15: Specifica test di unità