SWEeneyThreads

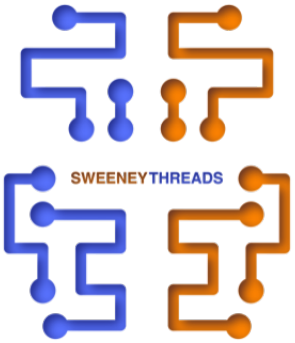
ActorBase

a NoSQL DB based on the Actor model

Piano di qualifica

Redattori:
Biggeri Mattia
Tommasin Davide

Approvazione:
Maino Elia
Verifica:
Bonato Paolo
Nicoletti Luca



Versione 2.0.2

15 maggio 2016

Indice

Diario delle modifiche

Versione	Data	Autore	Descrizione
2.1.0	2016-05-13	*Verificatore* Nicoletti Luca	Verificato il documento, inviato il documento contenente gli errori nella chat di gruppo
2.0.2	2016-05-13	*Progettisti* Paolo Bonato Tommaso Padovan	Aggiunti test di Validazione e test di Unità, calcolate le metriche sui processi.
2.0.1	2016-05-08	*Progettista* Paolo Bonato	Spostate le sezioni da 2.3 a 2.5 e 3 nelle norme di progetto. Aggiunte metriche sui processi da affiancare a SV e BV. Eliminata la metrica di annidamento dei requisiti. Ricalcolati i risultati delle metriche aggiunte nel resoconto delle attività di verifica in appendice.
2.0.0	2016-04-11	*Responsabile* Maino Elia	Documento approvato
1.6.0	2016-04-11	*Verificatore* Padovan Tommaso	Verificato il documento
1.5.1	2016-04-10	*Progettista* Biggeri Mattia	Correzioni da verifica, eliminata sezione errata 4.4.2, modificata introduzione alle metriche dei documenti.
1.5.0	2016-04-10	*Verificatore* Padovan Tommaso	Verifica documento.
1.4.5	2016-04-10	*Progettista* Nicoletti Luca	Separata tabella del diario della modifiche in file esterno.
1.4.4	2016-04-21	*Progettisti* Biggeri Mattia	Inserita tabella requisito test
1.4.3	2016-03-21	*Progettisti* Biggeri Mattia Bonato Paolo	Aggiunta introduzione e tabella valore metriche dei documenti in sezione Documenti 4.4.2, inserita completamente la sezione della pianificazione dei test, sezione Pianificazione test 4.5
1.4.2	2016-03-21	*Progettista* Padovan Tommaso	Incremento sezione 2.3. Aggiunta dei seguenti strumenti per l'applicazione delle metriche: Repo's Outpost, Camel Calculator e Gloss Buddy. Correzione errore nella data della versione 1.1.3 nel Diario delle modifiche.
1.4.1	2016-03-21	*Analista* Tommasin Davide	Controllo e sostituzione di termini all'interno della sezione di introduzione del documento
1.4.0	2016-03-19	*Verificatore* Nicoletti Luca	Verificate sezione 2.2.2(metriche per i documenti), 2.2.3 (metriche per il software)
1.3.2	2016-03-18	*Progettista* Padovan Tommaso	Aggiunte le seguenti metriche nelle sezioni 2.2.2 e 2.2.3: Numero di errori ortografici, Livelli di annidamento dei requisiti, Complessità ciclomatica, SLOC, Righe per ogni metodo. Corretto la forma tipografica di alcuni range di accettazione che non rispettavano lo standard.
1.3.1	2016-03-09	*Amministratore* Biggeri Mattia	Correzioni post verifica ed aggiunta immagini
1.3.0	2016-03-05	*Amministratore* Biggeri Mattia	Spostamento sezioni qualità di processo, qualità di prodotto e resoconto attività di verifica in appendice, sezioni misure e metriche e strumenti spostate sotto definizione obiettivi
1.2.0	2016-01-18	*Responsabile* Padovan Tommasin	Documento approvato, pronto alla consegna

Versione	Data	Autore	Descrizione
1.1.0	2016-01-17	*Verificatore* Biggeri Mattia	Verificato il documento e comunicato errori
1.0.4	2016-01-17	*Amministratori* Nicoletti Luca Tommaso Padovan	Miglioramento sezione Metriche e Analisi
1.0.2	2016-01-17	*Amministratore* Tommaso Padovan	Aggiunta qualità di prodotto, Analisi e Metriche
1.0.1	2016-01-17	*Amministratori* Nicoletti Luca Tommaso Padovan	Visione generale della strategia di verifica e Gestione amministrativa della revisione
1.0.0	2016-01-17	*Amministratore* Nicoletti Luca	Scrittura scheletro logico del documento

Tabella 1: Diario delle modifiche

1 Introduzione

1.1 Scopo del documento

Questo documento descrive le scelte effettuate in merito alle strategie che il gruppo ha deciso di adottare per raggiungere obiettivi qualitativi e misurabili da applicare al proprio prodotto. Per soddisfare questi obiettivi sarà necessario attuare un processo di verifica continuo sulle attività svolte in modo da poter rilevare ed eventualmente correggere anomalie e incongruenze in modo tempestivo per evitare danni e sprechi di risorse.

1.2 Scopo del prodotto

Il progetto consiste nella la realizzazione di un DataBase NoSQL key-value basato sul modello ad Attori$_G$ con l'obiettivo di fornire una tecnologia adatta allo sviluppo di moderne applicazioni che richiedono brevissimi tempi di risposta e che elaborano enormi quantità di dati. Lo sviluppo porterà al rilascio del software sotto licenza MIT.

1.3 Glossario

Al fine di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v2.0.0*. In esso saranno definiti, in modo chiaro e conciso i termini che possono causare ambiguità o incomprensione del testo.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di progetto:**
  Norme di progetto v2.0.0;
- **Capitolato d'appalto:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1p.pdf`.

1.4.2 Informativi

- **Piano di progetto:**
  Piano di progetto v2.0.0;
- **Slide del corso:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/`;
- **SWEBOK - Version 3:**
  `http://www.computer.org/web/swebok/v3`;
- ***TexMaker*:**
  `http://www.xm1math.net/texmaker/`;
- ***Scalastyle*:**
  `https://github.com/scalastyle/scalastyle`;
- ***Scapegoat*:**
  `https://github.com/sksamuel/scapegoat`;
- ***CLOC (Count Lines Of Code)*:**
  `cloc.sourceforge.net`;
- ***ScalaTest*:**
  `http://www.scalatest.org/`.

2 Visione generale della strategia di verifica

2.1 Definizione obiettivi

In questa sezione verranno descritti gli obiettivi di qualità relativi al prodotto che il gruppo ha deciso di raggiungere e gli obiettivi relativi ai processi che saranno svolti per il completamento del progetto. Per la stesura di questo documento e per la definizione delle metriche e degli strumenti atti a garantire la qualità del prodotto abbiamo seguito quanto definito in Appendice 4.5.1 e 4.5.2.

2.2 Misure e metriche

Il processo di verifica, per avere un valore informativo, deve essere quantificabile e le misure rilevate devono essere basate su metriche stabilite a priori. A causa della scarsa esperienza del gruppo, alcune metriche stabilite all'inizio potrebbero risultare approssimative ma seguendo il modello incrementale esposto nel *Piano di progetto* si potrà migliorarne la precisione e l'accuratezza.

Le tipologie di range ammesse sono due:

- **Accettabile:** Superiore al minimo valore richiesto affinché il prodotto sia accettato.
- **Ottimale:** Valori entro cui dovrebbe collocarsi la misurazione. Non sono vincolanti, ma fortemente consigliati. Misurazioni al di fuori di questi valori necessitano una verifica approfondita e nel caso non si trovi una maniera immediata per farli rientrare le cause di tale scostamento dovranno essere discusse nella successiva riunione.

2.2.1 Metriche per i processi

Gli indici scelti per la quantificazione dei processi prendono in considerazione principalmente costi e tempi; hanno lo scopo di mantenere il controllo sui processi e che il progetto segua quanto descritto nel *Piano di progetto*.

Gli indici scelti sono:

- **PPC (Partial Planned Cost):** Indica il costo pianificato per lo svolgimento di un sottoinsieme di attività. Si misura in euro e in ore.
- **PV (Planned Value):** Indica il valore che si prevede ottenere dal completamento delle attività pianificate. Per questo progetto tale valore corrisponde alla spesa richiesta per il completamento delle attività. Si misura in euro e in ore.
- **EV (Earned Value):** Indica il valore ottenuto tramite le attività completate alla data corrente. Per questo progetto tale valore corrisponde alla spesa richiesta per il completamento delle attività. Si misura in euro e in ore.
- **AC (Actual Cost):** Indica il costo effettivamente sostenuto alla data corrente. Si misura in euro e in ore. Aiuta a calcolare altre metriche.
- **BAC (Budget at Completion):** Costo previsto per portare a termine il progetto. Si misura in euro e in ore. Mantiene traccia della spesa totale preventivata all'inizio del progetto.
- **ETC (Estimate to Complete):** Indica i costi pianificati per portare a termine le attività di progetto rimanenti alla data corrente. Corrisponde al PV riesaminato allo stato corrente del progetto ma senza tenere conto delle attività completate. Si misura in euro e in ore.
- **EAC (Estimated at Completion):** Revisione del costo stimato per la realizzazione del progetto, ossia il BAC rivisto allo stato corrente del progetto. Si misura in euro e in ore e si ottiene dalla formula: EAC = AC + ETC.
- **SV (Schedule Variance):** É un indicatore di efficacia, mostra se si è o meno in linea con la pianificazione temporale rispetto alle attività nella baseline. Una schedule variance positiva indica che il gruppo è in anticipo rispetto al Piano di progetto, che è in ritardo altrimenti. Si ottiene dalla formula: SV = EV – PV.
- **BV (Budget Variance):**Indica se la spesa sostenuta alla data corrente è superiore o inferiore a quella preventivata. Una budget variance positiva indica che si è speso meno di quanto inizialmente previsto, viceversa altrimenti. Si ottiene dalla formula: BV = PV – AC.

2.2.2 Metriche per i documenti

Per i documenti abbiamo scelto di adottare le seguenti metriche:

- **Gulpease:** Lo abbiamo scelto come *indice di leggibilità* in quanto:
  - È l'unico tarato appositamente per la lingua italiana.
  - Utilizza la lunghezza delle parole in lettere anziché in sillabe, quindi è più semplice da automatizzare
  - Considera la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere
  - Permette di misurare la complessità dello stile di un documento

  La formula per il suo calcolo è la seguente:

$$89 + \frac{300 \cdot (numero\ delle\ frasi) - 10 \cdot (numero\ delle\ lettere)}{numero\ delle\ parole} \tag{1}$$

  I risultati sono compresi tra 0 e 100, dove il valore "100" indica la leggibilità più alta e "0" la leggibilità più bassa. In generale risulta che testi con un indice
  - inferiore a 80 sono difficili da leggere per chi ha la licenza elementare
  - inferiore a 60 sono difficili da leggere per chi ha la licenza media
  - inferiore a 40 sono difficili da leggere per chi ha un diploma superiore

  Il gruppo si atterrà ai seguenti parametri:
  - Range di accettazione: [35|100]
  - Range di ottimale: [45|100]
- **Numero parole:** Questa metrica servirà principalmente per misurare l'incremento del documento durante il completamento delle varie fasi.
- **Numero figure e tabelle su numero pagine:** Questa metrica verrà valutata solamente su i documenti esterni per verificarne la difficoltà di lettura. Il gruppo si atterrà ai seguenti parametri:
  - Range di accettazione: [0,25|2]
  - Range di ottimale: [0,5|2]
- **Media parole per section:** Servirà ad evitare la costruzione di sezioni troppo grandi, in modo che tutti gli argomenti siano facilmente trovabili sull'indice.
  - Valori accettati: [100|250]
  - Valori ottimali: [0|100]
- **Numero di errori ortografici:** Garantisce l'assenza di errori ortografici rilevati dal correttore automatico, a meno di falsi positivi.
  - Range di accettazione: [0|0]
  - Range di ottimale: [0|0]

2.2.3 Metriche per il software

Questa sezione è da intendere come una dichiarazione di intenti e probabilmente verrà rivista quando inizieremo realmente la fase di programmazione.

Al fine di perseguire gli obbiettivi sulla qualità del software, applicheremo le seguenti metriche:

- **Attributi per classe:** Un grande numero di attributi interni ad una classe mostra probabilmente la necessità di suddividere la classe in più classi relazionate tra loro.
  - Valori accettati: [0|18]
  - Valori ottimali: [2|9]

- **Numero livelli annidamento:** Mostra il livello di annidamento dei metodi, un numero alto implica una bassa astrazione del codice ed un' elevata complessità.
  - Valori accettati: [1|8]
  - Valori ottimali: [1|4]
- **Numero parametri per metodo:** Un valore elevato indica che probabilmente il metodo ha un sovraccarico di funzionalità.
  - Valori accettati: [0|8]
  - Valori ottimali: [0|5]
- **Linee di codice per linee di commento:** È la percentuale di righe di commento rispetto al totale delle righe di codice. Utile per la manutenibilità.
  - Valori accettati: [0,2|0,7]
  - Valori ottimali: [0,3|0,5]
- **Accoppiamento afferente:** Indica il numero di classi esterne ad un package che dipendono da esso, un grande valore indica una forte dipendenza del software per il package in questione, un valore basso invece indica una bassa utilità del package per il resto del software.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Accoppiamento efferente:** Il numero di classi di un package che dipendono da package esterni, un valore basso indica che il package ha numerose funzionalità indipendenti dal resto del software.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Linee di codice per metodo:** È una misura della leggibilità del codice: valori troppo elevanti indicano che il corpo di un metodo potrebbe essere scarsamente comprensibile.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Source Line Of Code (SLOC):** Il numero di istruzioni presenti nel codice. Questa metrica fornisce una stima della complessità del programma. È utile anche per dare una stima di quanto il codice incrementerà nel tempo, semplificando così la pianificazione.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Complessità Ciclomatica:** Una metrica sviluppata da Thomas J. McCabe che consente di stimare la complessità di un programma misurando il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere

2.3 Pianificazione strategica e temporale

Avendo lo scopo di rispettare le scadenze riportate nel *Piano di progetto*, è necessario che l'attività di verifica, sia del codice che della documentazione, sia sistematica e ben organizzata; in questo modo l'individuazione, e quindi la correzione degli errori avverrà il prima possibile, limitando la diffusione degli stessi.
Per cercare di ridurre il numero degli errori, e quindi semplificare l'attività di verifica, ogni fase di codifica o documentazione sarà preceduta da una fase di studio preliminare. Evitando le imprecisioni di natura concettuale si ridurranno le correzioni necessarie.

Di seguito vengono riportate le scadenze previste:

- **Revisione dei requisiti:** 2016-01-22;
- **Revisione di progettazione:** 2016-04-18;
- **Revisione di qualifica:** 2016-05-23;
- **Revisione di accettazione:** 2016-06-17.

2.4 Responsabilità

Il *Responsabile di progetto* ha il compito di:

- Accertarsi che le attività di verifica vengano svolte sistematicamente secondo quanto riportato nelle *Norme di progetto*;
- Accertarsi che vengano rispettati ruoli e competenze assegnate nel *Piano di progetto*;
- Verificare che non ci siano conflitti di interesse tra redattori e *Verificatori*;
- Aprire ed assegnare i ticket principali e le task-list;
- Approvare un documento e sancirne la distribuzione.

I *Verificatori* hanno il compito di:

- Effettuare la verifica dei documenti con strumenti e metodi proposti nel *Piano di Qualifica*;
- Attenersi rigidamente a quanto sancito nelle *Norme di progetto*;
- Segnalare tempestivamente un errore, qualora riscontrato;
- Sottoporre i documenti all'approvazione del *Responsabile*, una volta giunti ad uno stadio finale.

2.5 Risorse

Per la realizzazione del progetto sono necessarie risorse sia umane che tecnologiche.

2.5.1 Risorse umane

Vengono descritte nel dettaglio nel *Piano di progetto* e sono:

- *Responsabile di progetto*;
- *Amministratore*;
- *Analista*;
- *Progettista*;
- *Programmatore*;
- *Verificatore*.

2.5.2 Risorse software

Sono necessari tutti i software utili

- alla gestione di documentazione in LaTeX;
- alla creazione di diagrammi UML;
- allo sviluppo di codice *Scala*;
- a semplificare ed automatizzare la verifica;
- a semplificare ed automatizzare la pianificazione e la documentazione della stessa;
- a semplificare ed automatizzare la comunicazione interna tra i membri del gruppo;
- a gestire test ed analisi sul codice.

2.5.3 Risorse hardware

- computer dotati di tutti i software descritti nel *Piano di qualifica* e nelle *Norme di progetto*;
- luoghi dove effettuare le riunioni del gruppo.

Tutti i membri del gruppo hanno a disposizione almeno un computer personale dotato di tutti gli strumenti necessari per il progetto; tutte le macchine in questione sono portatili. Inoltre in caso di rottura o guasto è messo a disposizione un computer di riserva. Sono a disposizione quattro appartamenti a Padova dove effettuare le riunioni. La scelta di quale viene presa di volta in volta a seconda delle disponibilità. Tutti gli appartamenti sono dotati di connessione internet a banda larga.

2.6 Analisi

2.6.1 Tecniche per l'analisi statica

L'analisi statica non richiede l'esecuzione del codice in oggetto, ed è quindi applicabile sia alla documentazione che al codice. Permette di individuare errori ed anomalie al più presto possibile, scongiurandone la diffusione.
Essa può essere svolta in due modi distinti.

2.6.2 Walkthrough

Si svolge effettuando una lettura critica a pettine. Questa tecnica viene utilizzata prevalentemente nelle prime fasi del progetto, in cui non si ha ne una adeguata esperienza, ne uno storico degli errori più comuni che permetta una indagine più mirata. I Verificatori, tramite questa tecnica, saranno in grado di stilare una lista di errori più frequenti, potendo così applicare successivamente la tecnica *Inspection*.
Il *Walkthrough* è una tecnica onerosa e richiede l'intervento di più persone. Dopo una fase iniziale in cui i Verificatori leggono il documento ed individuano potenziali errori essi devono essere discussi in una riunione con altri componenti del gruppo per accertare che non siano dei falsi positivi.

2.6.3 Inspection

È una tecnica molto meno onerosa. Consiste nel controllare alcune parti dei documenti che si sono rivelate maggiormente prone ad errori. Per ottenere questo risultato è necessario avere una lista di controllo che indichi quali sono le parti da controllare in maniera mirata. Essa viene stilata durante le fasi di *Walkthrough*. Un altro motivo per cui la *Inspection* è preferibile è il fatto che essa richiede l'intervento dei soli verificatori, che poi possono procedere alla correzione della maggior parte degli errori, oppure ad aprire un ticket riguardante quelli che non sono di immediata risoluzione.

Durante l'applicazione del *Walkthrough* ai documenti sono state riportate le tipologie di errori più frequenti, esse costituiscono quindi la lista di controllo per le verifiche ad *Inspection*, l'attuale lista si trova in appendice sezione 4.6 e una versione sempre aggiornata della stessa è presente sul drive del gruppo.

2.6.4 Tecniche per l'analisi dinamica

Questo tipo di analisi richiede una esecuzione di parte del programma, quindi ovviamente non applica ai documenti ma solo al codice. Il suo obiettivo è rilevare errori o difetti di implementazione mediante l'uso di test che devono essere necessariamente ripetibili: solo un test che produca lo stesso output partendo dallo stesso ambiente e lo stesso input può è capace di riscontrare problemi. L'attore che esegue un test deve definire a priori ed avere il pieno controllo su:

- **Ambiente:** insieme di hardware a software come sistema operativo e altri programmi o processi in esecuzione;
- **Specifiche:** definizione degli input e dei relativi output attesi, che sono ripetibili in quanto si postula di essere in un ambiente deterministico;
- **Procedure:** descrizione delle azioni compiute dall'attore (umano o computer che sia) per arrivare allo stato iniziale, far partire l'esecuzione, inserire gli input specificati e verificare che l'output sia uguale a quello atteso.

Sono definiti 5 tipi di test:

- **Test di unità:** Una unità viene definita come la più piccola quantità di software che conviene testare singolarmente. Il fine di questi test è cercare di individuare eventuali errori presenti nelle singole unità che compongono l'intero sistema. Essi vengono testati attraverso l'uso di stub, driver e logger. Queste verifiche sono spesso le più onerose, ma anche quelle che portano alla luce il maggior numero di errori, quindi quelle che producono il maggior valore.
- **Test di integrazione:** Consiste nella verifica di componenti del sistema che vengono aggiunti incrementalmente, è necessario dunque analizzare combinazioni di due o più unità di software. Hanno lo scopo di individuare errori residui nella realizzazione dei singoli moduli, modifiche delle interfacce e comportamenti inaspettati di componenti software preesistenti forniti da terze parti che non si conoscono a fondo. Per la loro realizzazione è necessario usare spesso componenti fittizie non ancora sviluppate, ma che emulano il comportamento atteso.
- **Test di Sistema:** Consiste nella validazione del prodotto software una volta che siano stati aggiunti tutti i componenti e lo si ritiene giunto ad una versione definitiva. Lo scopo principale è verificare che ci sia totale copertura dei requisiti stabiliti nella fase di Analisi di dettaglio. È obiettivo fondamentale della qualità del processo fare in modo che giunti a questo punto l'esito del test sia comunque positivo, in quanto garantito dal tracciamento dei requisiti.
- **Test di regressione:** Consiste nell'eseguire nuovamente i test di unità e integrazione in porzioni di software che hanno subito modifiche in maniera da accertare che questi cambiamenti non pregiudichino il funzionamento dei componenti non toccati da questa modifica.
- **Test di accettazione:** Consiste nel collaudo del prodotto che viene eseguito in presenza del proponente. Un esito positivo di questo test permette il rilascio ufficiale del software.

3 Appendice

3.1 Qualità

Riportiamo gli standard di riferimento per sviluppare le metriche e i metodi atti a garantire la qualità del prodotto.

Figura 1: Influenze e dipendenze delle varie misure di qualità.

3.1.1 Qualità di processo

Per garantire la qualità del prodotto è necessario garantire anche quella dei processi necessari al suo completamento. A questo scopo si è deciso di adottare lo standard ISO/IEC 15504 denominato SPICE.

Questo modello descrive come ogni processo debba essere controllato costantemente in maniera da rilevare possibili errori o debolezze e correggerli prima che essi si diffondano, facendo aumentare esponenzialmente il carico di lavoro. Affinché le singole valutazioni contribuiscano all'effettivo miglioramento dei processi devono essere sempre ripetibili, oggettivi e comparabili. SPICE definisce 6 livelli di maturità del processo:

0 - Incomplete

1 - Performed

2 - Managed

3 - Established

4 - Predictable

5 - Optimizing

Al fine di applicare correttamente questo modello è evidentemente indispensabile adottare il principio PDCA il quale definisce una metodologia di controllo dei processi durante il loro ciclo di vita che consente di migliorarne in modo continuativo la qualità.
Esso di compone di 4 fasi:

- **Plan:** definire dettagliatamente cosa deve essere realizzato rispetto agli obiettivi di miglioramento, e come questi controlli saranno effettuati;
- **Do:** fase di esecuzione delle attività pianificate;
- **Check:** vengono confrontati i dati in uscita dalla fase *Do* con quelli pianificati nella fase *Plan*, per intervenire in tempo e migliorare i risultati;
- **Act:** fase in cui si mette in pratica il miglioramento continuo dei processi utilizzando i risultati della verifica per modificare gli aspetti critici dei processi in esame.

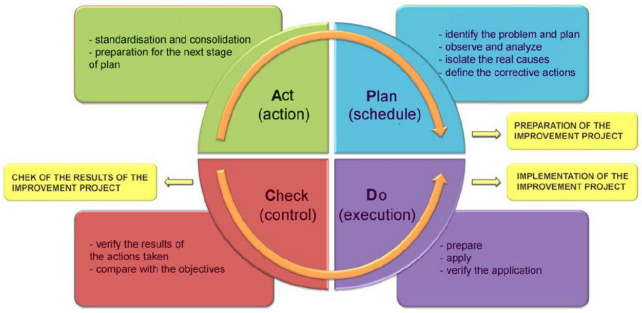


Figura 2: Fasi del principio PDCA.

3.1.2 Qualità di prodotto

Al fine di aumentare il valore del prodotto e di migliorarne il funzionamento è necessario fissare degli obbiettivi qualitativi e garantire che saranno raggiunti.
Questi obbiettivi sono descritti nell'ISO/IEC 9126 dove sono anche descritte le metriche per misurare gli stessi.
I criteri valutativi sono suddivisi in 3 aree:

- **Qualità esterna:** Le metriche esterne, specificate nella norma ISO/IEC 9126-2, valutano i comportamenti del prodotto sulla base di prove, dall'operatività e dall'osservazione durante la sua esecuzione, in funzione degli obiettivi stabiliti.
- **Qualità interna:** È specificata nella norma ISO/IEC 9126-3 e si applica al software non eseguibile durante la progettazione e la codifica dello stesso, le misure effettuate consentono di prevedere il livello di qualità esterna ed interna, in quanto gli attributi interni influenzano quelli esterni e di uso
- **Qualità d'uso:** Rappresenta la qualità dal punto di vista dell'utente finale, viene raggiunto quando sono raggiunte la qualità esterna e quella interna, le metriche di valutazione sono fornite nella norma ISO/IEC 9126-4.

lo standard ISO/IEC 9126 prevede di suddividere la qualità esterna ed interna in 6 caratteristiche principali tra le quali la funzionalità è l'unico ?requisito funzionale? mentre le altre 5 sono ?requisiti di qualità?, ciascuna caratteristica si suddivide in altre sotto caratteristiche che possono essere misurate qualitativamente:

- **Funzionalità:** capacità del prodotto software di fornire funzioni che rispondano a esigenze stabilite
  - **Idoneità:** capacità del prodotto software di fornire un insieme di funzioni per attività specifiche già conosciute dall'utente;
  - **Accuratezza:** capacità del prodotto software di fornire risultati esatti o concordi al grado di precisione necessario;
  - **Interoperabilità:** capacità del prodotto software di interagire con uno o più sistemi precedentemente specificati;
  - **Sicurezza:** capacità del prodotto software di proteggere dati e informazioni;
  - **Conformità funzionale:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni e prescrizioni in materia di funzionalità.
- **Affidabilità:** capacità del prodotto software di mantenere uno specifico livello di prestazioni quando usato
  - **Maturità:** capacità del prodotto software di non fallire a causa di errori nel software;
  - **Tolleranza agli errori:** capacità del prodotto software di mantenere un adeguato livello di prestazioni e funzioni in caso di errori software o di violazioni;
  - **Capacità di recupero:** capacità del prodotto software di ristabilire un adeguato livello di performance e di recuperare i dati in caso di errori;
  - **Conformità di affidabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di affidabilità.
- **Usabilità:** capacità del software di essere capito, imparato, usato e apprezzato dall'utente quando usato
  - **Comprensibilità:** capacità del prodotto software di far comprendere all'utente se il prodotto è adatto ad uno specifico scopo;
  - **Apprendibilità:** capacità del prodotto software di ridurre all'utente il tempo necessario per apprendere le sue funzioni;
  - **Operabilità:** capacità del prodotto software di essere utilizzato dall'utente in modo controllato
  - **Attrattiva:** capacità del prodotto software di creare interesse nell'utente;
  - **Conformità di usabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di usabilità.
- **Efficienza:** capacità del software di fornire prestazioni appropriate in relazione alla quantità di risorse in utilizzo
  - **Comportamento temporale:** capacità del software di fornire tempi di risposta e di elaborazione adeguati sotto condizioni determinate;
  - **Utilizzo di risorse:** capacità del prodotto software di utilizzare quantità e tipo di risorse adeguate durante la sua esecuzione;
  - **Conformità di efficienza:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di efficienza.
- **Manutenibilità:** capacità del prodotto software di essere modificato e ampliato.
  - **Analizzabilità:** rappresenta la facilità con la quale è possibile analizzare il software alla ricerca di carenze e difetti;
  - **Modificabilità:** capacità del prodotto software di permettere l'implementazione di una specifica modifica o di un aggiornamento;
  - **Stabilità:** capacità del prodotto software di evitare effetti indesiderati causati da uno o più aggiornamenti o modifiche;
  - **Testabilità:** capacità del prodotto software di consentire una facile validazione di una versione modificata del software;

- **Conformità di manutenibilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di manutenibilità.

- **Portabilità:** capacità del prodotto software di poter essere trasferito da un ambiente di lavoro ad un altro sia dal punto di vista hardware che per quanto riguarda il sistema operativo
  - **Adattabilità:** capacità del prodotto software di essere adattato a diversi ambienti di lavoro senza la necessità di effettuare modifiche aggiuntive;
  - **Installabilità:** capacità del prodotto software di poter essere installato in specifici ambienti;
  - **Coesistenza:** capacità del prodotto software di coesistere in ambienti comuni con altri software indipendenti condividendo risorse comuni;
  - **Sostituibilità:** capacità del prodotto software di poter sostituire un software analogo o simile nello stesso ambiente;
  - **Conformità di portabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di portabilità.

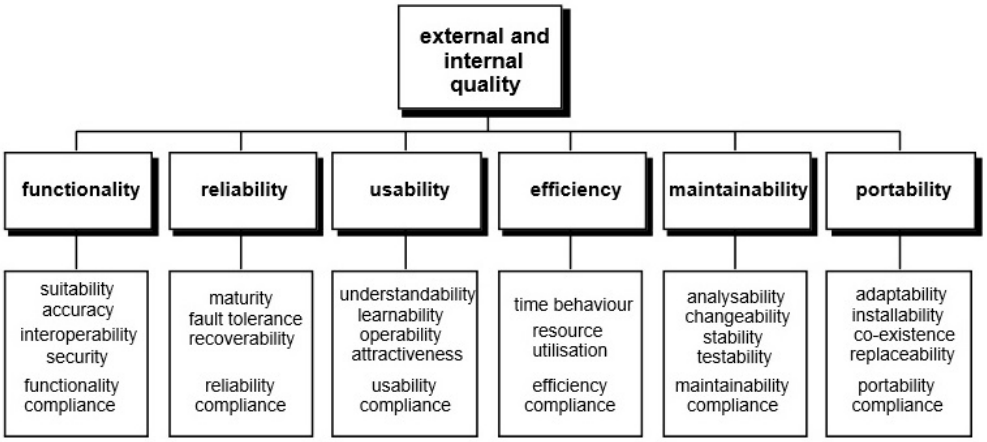


Figura 3: Caratteristiche della qualità esterna e interna.

Per la qualità d'uso invece vengono invece definite le seguenti caratteristiche:

- **Efficacia:** la capacità del prodotto di consentire agli utenti di raggiungere gli obiettivi specificati con precisione sufficiente e completezza.
- **Produttività:** la capacità di consentire agli utenti di utilizzare una quantità di risorse appropriate in relazione all'efficacia ottenuta in contesto d'uso definito.
- **Soddisfazione:** è la capacità del prodotto di soddisfare gli utenti.
- **Sicurezza:** rappresenta la capacità del prodotto di avere accettabili livelli di rischio per quanto riguarda i danni alle persone, al software, ad apparecchiature o all'ambiente operativo d'uso.

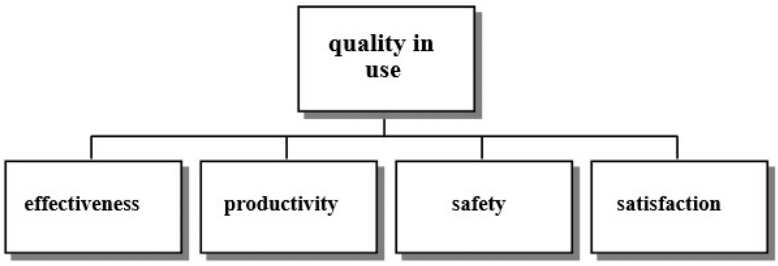


Figura 4: Caratteristiche della qualità d'uso.

3.2 Resoconto delle attività di verifica

3.2.1 Riassunto delle attività di verifica

Durante la stesura della documentazione sono stati verificati ad ogni modifica sostanziale la documentazione stessa, i casi d'uso e i requisiti; è stato in oltre verificato se i processi si sono svolti in maniera corretta.

3.2.2 Documentazione

I documenti sono stati verificati tramite walkthrough da due persone differenti seguendo il seguente protocollo:

- Verifica della sintassi e della correzione ortografica
- Verifica della chiarezza espositiva
- Verifica del rispetto delle *Norme Di Progetto v2.0.0* capitolo 3.1
- Verifica dell'uniformità dei termini rispetto allo stesso documento
- Verifica dell'uniformità dei termini rispetto agli altri documenti
- Verifica che i termini che lo necessitano siano stati inseriti nel glossario
- Produzione dei un file .txt contenente tutti gli errori al fine di mostrare le correzioni necessarie e facilitare la stesura della checklist degli errori più frequenti

In seguito i documenti sono stati nuovamente verificati tramite inspection utilizzando la checklist precedentemente stilata.

3.2.3 Casi d'uso

I casi d'uso sono stati verificati sempre tramite walkthrough ponendo l'attenzione sui seguenti punti:

- Uniformità dei termini usati tra i vari UC
- Uniformità tra l'immagine dei diagrammi e la spiegazione della suddetta
- Correttezza del codice utilizzato rispetto a quanto definito nelle *Norme Di Progetto v2.0.0* sezione 2.1.3
- Uniformità dei casi d'uso rispetto al capitolato
- Rispetto della struttura definita nelle *Norme Di Progetto v2.0.0* sezione 2.3.1

I Casi d'uso sono stati successivamente ricontrollati all'interno del documento di *Analisi Dei Requisiti* come descritto nella sezione 4.2.2

3.2.4 Requisiti

I requisiti sono stati controllati tramite walkthrough seguendo il seguente protocollo:

- Verifica che la fonte del requisito sia corretta.
- Nel caso in cui la fonte sia un UC, verifica che sia l?UC corretto e valutazione sulla possibilità di dedurre altri requisiti dallo stesso.
- Verifica uniformità dei termini tra i requisiti.
- Rispetto del codice e della struttura definiti nelle *Norme di Progetto v2.0.0* sezione 2.1.2.
- I Casi d'uso sono stati successivamente ricontrollati all'interno del documento di *Analisi Dei Requisiti* come descritto in sezione 4.1.

3.2.5 Processi

I processi sono stati verificati dal responsabile grazie all'utilizzo di teamwork controllando le seguenti condizioni:

- Che la verifica non sia stata effettuata dallo stesso che ha prodotto il materiale da verificare.
- Che i documenti siano stati prodotti nell'ordine e nei tempi corretti.
- Che le ore di verifica siano almeno 30% delle ore totali.
- Che le riunioni e i brainstorming si siano svolti come definito nelle *Norme di Progetto v2.0.0* sezione 4.2.3.

3.3 Dettaglio delle verifiche tramite analisi

3.3.1 Processi

Vengono qui di seguito riportati i valori delle metriche calcolati sulle fasi di progetto portate a termine.

Valore di BAC = 17457 €, 901 ore.
Valore totale di SV= 0 €, 0 ore.
Valore totale di BV= -5 €, -1 ore.

Tabella con valori calcolati in euro:

Fase	PPC	PV	EV	AC	ETC	EAC
Scelta ed approccio al capitolato	3012	3012	3012	2974	14277	17251
Analisi di dettaglio	945	3957	3957	3919	13332	17251
Progettazione e sviluppo:	4068	8025	8025	8038	9264	17302
Sviluppo ulteriore ed incremento:	6707	14732	14732	14737	2577	17294
Conclusione:	2557	17289				

Tabella 2: Tabella delle fasi e relativi valori in euro delle metriche di processo

Tabella con valori calcolati in ore:

Fase	PPC	PV	EV	AC	ETC	EAC
Scelta ed approccio al capitolato:	136	136	136	136	765	901
Analisi di dettaglio:	42	178	178	178	723	901
Progettazione e sviluppo:	197	375	375	377	526	903
Sviluppo ulteriore ed incremento:	382	757	757	758	144	902
Conclusione:	144	901				

Tabella 3: Tabella delle fasi e relativi valori in ore delle metriche di processo

Vengono inoltre presentati in dettaglio per ogni fase di progetto i valori di SV in ore e BV in euro, esaminando le macro-attività che ne fanno parte.

Macro-attività	SV	BV
Norme di progetto	ore 1	€ 30
Studio di fattibilità	ore 2	€ 50
Analisi dei requisiti	ore -2	€ -20
Piano di progetto	ore -1	€ -22
Piano di qualifica	ore 0	€ 0
Totale	ore 0	€ 38

Tabella 4: Tabella delle attività con SV e BV della fase di scelta ed approccio al capitolato

Da questi dati si può dedurre che:

- I tempi impiegati hanno subito delle leggere variazioni e grazie agli slack inseriti nel *Piano di progetto* non ci sono stati ritardi significativi.
- Il costo effettivo è molto vicino a quanto preventivato nel *Piano di progetto* anche se alcuni picchi, specie nell'analisi dei requisiti, significano che è possibile migliorare la pianificazione dei processi.

Durante la prima fase di analisi le problematiche riscontrate sui processi sono dovute per la maggior parte all'inesperienza del gruppo. A questa fase seguirà una pianificazione su come migliorare i processi che sono risultati più critici.

Macro-attività	SV	BV
Glossario	ore 0	€ 0
Norme di progetto	ore 0	€ 0
Piano di progetto	ore 0	€ 0
Analisi dei requisiti	ore 0	€ 0
Piano di qualifica	ore 0	€ 0
Totale	ore 0	€ 0

Tabella 5: Tabella delle attività con SV e BV della fase di analisi di dettaglio

Da questi dati si può dedurre che:

- La pianificazione è risultata essere accurata. Non ci sono state variazioni rispetto a quanto pianificato.

Il periodo di tempo e le ore impiegate in questa fase sono stati ridotti. Questo ha reso molto più semplice fare una pianificazione accurata delle attività. Inoltre i processi sono stati affinati, soprattutto grazie alla piena adozione di metodi automatici per la gestione dei requisiti.

Macro-attività	SV	BV
Glossario	ore 2	€ 30
Norme di progetto	ore 1	€ 20
Piano di progetto	ore 2	€ 45
Analisi dei requisiti	ore -2	€ -50
Specifica tecnica	ore -5	€ -96
Piano di qualifica	ore 0	€ 0
Totale	ore -2	€ -51

Tabella 6: Tabella delle attività con SV e BV della fase di progettazione e sviluppo

Da questi dati si può dedurre che:

- I tempi impiegati hanno subito delle variazioni ma grazie agli slack inseriti nel *Piano di progetto* non ci sono stati ritardi significativi.

- Il costo effettivo ha subito un certo incremento. Grazie al risparmio ottenuto in fase di approccio al capitolato e alle precauzioni prese nel preventivo il costo del progetto risulta essere ancora al di sotto del budget prefissato.

Durante la fase di progettazione e sviluppo le problematiche riscontrate nei processi sono dovute per la maggior parte alla pianificazione. I processi per la produzione dei documenti sono stati migliorati, ma le attività di analisi dei requisiti e lo studio di fattibilità sono state sottovalutate.

Macro-attività	SV	BV
Glossario	ore 0	€ 0
Norme di progetto	ore 0	€ 0
Piano di progetto	ore 0	€ 0
Analisi dei requisiti	ore 0	€ 0
Specifica tecnica	ore 0	€ 0
Piano di qualifica	ore -1	€ -15
Resoconto attività di verifica	ore 0	€ 0
Sviluppo	ore 3	€ 45
Definizione di prodotto	ore -1	€ -22
Manuale utente	ore 0	€ 0
Totale	ore 1	€ 8

Tabella 7: Tabella delle attività con SV e BV della fase di sviluppo ulteriore ed incremento

Da questi dati si può dedurre che:

- I tempi impiegati hanno subito delle variazioni ma grazie agli slack inseriti nel *Piano di progetto* non ci sono stati ritardi significativi.
- Il costo effettivo ha subito un lieve decremento, che lenisce leggermente l'incremento verificatosi nella fase precedente.

Durante la fase di sviluppo ulteriore ed incremento le attività non hanno subito cambiamenti significativi, eccezione fatta per lo sviluppo. Il team si è concentrato maggiormente nello studio del linguaggio di programmazione *Scala*, il che ha portato ad uno sviluppo significativamente più rapido di quanto previsto. Al contrario sono emerse difficoltà maggiori del previsto nella definizione e nella realizzazione dei test per il software.

3.3.2 Documenti

Segue la tabella contenente il valore delle metriche di tutti i documenti presentati. Questa tabella viene aggiornata ad ogni cambiamento di versione. La maggior parte dei valori è in range di ottimalità; eccetto i seguenti che sono in ogni caso in range di accettazione:

- **Norme di Progetto vX.Y.Z:**
  - **Gulpease:** L'indice Gulpease è comunque sufficientemente alto e, trattandosi di un documento interno, il gruppo ha ritenuto ciò non problematico.
- **Piano di Qualifica vX.Y.Z:**
  - **Gulpease:** Trattandosi di un documento tecnico contiene lunghi periodi e termini tecnici che ne diminuiscono la leggibilità.
  - **Lunghezza media sezioni:** Sono presenti molte sezioni descrittive che aumentano questa metrica.
- **Studio di Fattibilità vX.Y.Z:**
  - **Gulpease:** Trattandosi di un documento tecnico contiene lunghi periodi e termini tecnici che ne diminuiscono la leggibilità.

documenti	Gulpease	parole	figure per pagina	parole per section	errori ortografici
Analisi dei requisiti	50,11	4405	1,86	37,02	0
Norme di progetto	43,78	4251	0,95	88,56	0
Piano di progetto	45,28	4724	0,51	59,05	0
Piano di qualifica	43,70	5409	0,58	115,09	0
Studio di fattibilità	39,84	1471	0,25	49,03	0
Specifica tecnica	89,35	5699	0,63	15,00	0

Tabella 8: Tabella metriche dei documenti

3.4 Test di validazione

In questa sezione vengono descritti i test di validazione che servono per accertare che il prodotto realizzato sia conforme alle attese. Per ogni test vengono descritti i vari passi che un utente deve eseguire per testare i requisiti ad esso associati. Il client di Actorbase funziona tramite riga di comando, per questo motivo viene sempre riportata la sintassi dei comandi che si intendono verificare. Viene richiesto di porre attenzione al rispetto degli spazi e all'utilizzo degli apici singoli, che identificano le chiavi. Le parentesi uncinate "<>" invece non vanno riportate e servono unicamente a contraddistinguere i parametri scelti dall'utente dai comandi. I comandi non sono case sensitive e possono essere scritti in maiuscolo o in minuscolo indifferentemente. Al contrario i parametri inseriti dall'utente vengono interpretati tenendo conto delle lettere maiuscole e minuscole.

3.4.1 Test TV1: Connessione

L'utente vuole verificare che sia possibile connettersi al server di Actorbase tramite l'interfaccia del client.
Sono richiesti i seguenti passaggi:

- Accertarsi che il server sia in esecuzione all'indirizzo <host> desiderato. "localhost" di default.
- Accertarsi che il server sia in ascolto sulla porta <port> desiderata. "8181" di default.
- Accertarsi che sia disponibile nel database un utente con l'username <username> desiderato. "admin" di default.
- Accertarsi che la password <password> inserita sia propria dell'utente <username> desiderato. "admin" di default.
- Digitare nella console del client il comando di connessione utilizzando la sintassi: connect <host>:<port> <username> <password>
- Eseguire il comando tramite il tasto invio.

3.4.2 Test TV2: Aiuto

L'utente vuole verificare la funzionalità di aiuto, sia generico che per un dato comando.
Sono richiesti i seguenti passaggi:

- Digitare nella console del client il comando di aiuto utilizzando la sintassi: help
- Eseguire il comando tramite il tasto invio.
- Scegliere un comando <command> dalla lista dei comandi ritornati.
- Digitare nella console del client il comando di help utilizzando la sintassi: help <command>
- Eseguire il comando tramite il tasto invio.

3.4.3 Test TV3: Database disponibili

L'utente vuole verificare la funzionalità di visualizzazione dei nomi dei database disponibili. Sono richiesti i seguenti passaggi:

- Digitare nella console del client il comando listdb utilizzando la sintassi: listdb
- Eseguire il comando tramite il tasto invio.

3.4.4 Test TV4: Creazione database

L'utente vuole verificare la funzionalità di creazione di un nuovo database. Sono richiesti i seguenti passaggi:

- Scegliere un nome <dbname> per il database che si vuole creare.
- Digitare nella console del client il comando di creazione database utilizzando la sintassi: createdb <dbname>
- Eseguire il comando tramite il tasto invio.

3.4.5 Test TV5: Rimozione database

L'utente vuole verificare la funzionalità di rimozione di un database. Sono richiesti i seguenti passaggi:

- Scegliere un database da eliminare tramite il suo nome <dbname>.
- Digitare nella console del client il comando di rimozione database utilizzando la sintassi: deletedb <dbname>
- Eseguire il comando tramite il tasto invio.

3.4.6 Test TV6: Selezione database

L'utente vuole verificare la funzionalità di selezionare di un database. Sono richiesti i seguenti passaggi:

- Scegliere un database da selezionare tramite il suo nome <dbname>.
- Digitare nella console del client il comando di selezione database utilizzando la sintassi: selectdb <dbname>
- Eseguire il comando tramite il tasto invio.

3.4.7 Test TV7: Mappe disponibili

L'utente vuole verificare la funzionalità di visualizzazione delle mappe del database selezionato. Sono richiesti i seguenti passaggi:

- Digitare nella console del client il comando listmap utilizzando la sintassi: listmap
- Eseguire il comando tramite il tasto invio.

3.4.8 Test TV8: Creazione mappa

L'utente vuole verificare la funzionalità di creazione di una nuova mappa all'interno del database selezionato. Sono richiesti i seguenti passaggi:

- Scegliere un nome <mapname> per la mappa che si vuole creare.
- Digitare nella console del client il comando di creazione mappa utilizzando la sintassi: createmap <mapname>
- Eseguire il comando tramite il tasto invio.

3.4.9 Test TV9: Rimozione mappa

L'utente vuole verificare la funzionalità di rimozione di una mappa dal database selezionato. Sono richiesti i seguenti passaggi:

- Scegliere una mappa da eliminare tramite il suo nome <mapname>.
- Digitare nella console del client il comando di rimozione mappa utilizzando la sintassi: deletemap <mapname>
- Eseguire il comando tramite il tasto invio.

3.4.10 Test TV10: Selezione mappa

L'utente vuole verificare la funzionalità di selezione di una mappa del database selezionato. Sono richiesti i seguenti passaggi:

- Scegliere una mappa da selezionare tramite il suo nome <mapname>.
- Digitare nella console del client il comando di selezione mappa utilizzando la sintassi: selectmap <mapname>
- Eseguire il comando tramite il tasto invio.

3.4.11 Test TV11: Chiavi disponibili

L'utente vuole verificare la funzionalità di visualizzazione dell'elenco delle chiavi della mappa selezionata. Sono richiesti i seguenti passaggi:

- Digitare nella console del client il comando keys utilizzando la sintassi: keys
- Eseguire il comando tramite il tasto invio.

3.4.12 Test TV12: Inserimento item

L'utente vuole verificare la funzionalità di inserimento di un item nella mappa selezionata. Sono richiesti i seguenti passaggi:

- Scegliere una chiave <key> per l'item.
- Scegliere un valore <value> da inserire.
- Digitare nella console del client il comando di inserimento item utilizzando la sintassi: insert '<key>' <value>
- Eseguire il comando tramite il tasto invio.

3.4.13 Test TV13: Ricerca item

L'utente vuole verificare la funzionalità che permette di recuperare il valore associato ad una chiave. Sono richiesti i seguenti passaggi:

- Scegliere la chiave <key> del valore che si vuole recuperare.
- Digitare nella console del client il comando di ricerca item utilizzando la sintassi: find '<key>'
- Eseguire il comando tramite il tasto invio.

3.4.14 Test TV14: Aggiornamento item

L'utente vuole verificare la funzionalità di aggiornamento di un item della mappa selezionata. Sono richiesti i seguenti passaggi:

- Scegliere la chiave <key> del valore che si vuole aggiornare.
- Scegliere il nuovo valore <value> che si vuole inserire.
- Digitare nella console del client il comando di aggiornamento item utilizzando la sintassi: update '<key>' <value>
- Eseguire il comando tramite il tasto invio.

3.4.15 Test TV15: Rimozione item

L'utente vuole verificare la funzionalità di rimozione di un item dalla mappa selezionata. Sono richiesti i seguenti passaggi:

- Scegliere la chiave <key> dell'item che si vuole rimuovere.
- Digitare nella console del client il comando di rimozione di un item utilizzando la sintassi: remove '<key>'
- Eseguire il comando tramite il tasto invio.

3.4.16 Test TV16: Disconnessione

L'utente vuole verificare la funzionalità di disconnessione dal server. Sono richiesti i seguenti passaggi:

- Digitare nella console del client il comando di disconnessione utilizzando la sintassi: disconnect
- Eseguire il comando tramite il tasto invio.

3.4.17 Test TV17: Visualizzazione utenti

L'utente vuole verificare la funzionalità dell'amministratore per visualizzare la lista degli utenti. Sono richiesti i seguenti passaggi:

- Digitare nella console del client il comando per visualizzare la lista degli utenti utilizzando la sintassi: listusers
- Eseguire il comando tramite il tasto invio.

3.4.18 Test TV18: Visualizzazione permessi

L'utente vuole verificare la funzionalità dell'amministratore per visualizzare la lista dei permessi di un utente. Sono richiesti i seguenti passaggi:

- Scegliere un utente tramite il suo usename <username> di cui si vogliono visualizzare i permessi.
- Digitare nella console del client il comando per visualizzare la lista dei permessi utilizzando la sintassi: userpermissions <username>
- Eseguire il comando tramite il tasto invio.

3.4.19 Test TV19: Aggiunta utente

L'utente vuole verificare la funzionalità dell'amministratore per aggiungere un utente. Sono richiesti i seguenti passaggi:

- Scegliere un nome <username> per l'utente che si vuole aggiungere.
- Scegliere una password <password> per l'utente che si vuole aggiungere.
- Digitare nella console del client il comando per aggiungere un utente utilizzando la sintassi: adduser <username> <password>
- Eseguire il comando tramite il tasto invio.

3.4.20 Test TV20: Aggiunta permessi

L'utente vuole verificare la funzionalità dell'amministratore per aggiungere permessi ad un utente. Sono richiesti i seguenti passaggi:

- Scegliere il nome <username> dell'utente a cui si vogliono aggiungere permessi.
- Scegliere un database <database> sul quale aggiungere i permessi all'utente.
- Scegliere il tipo di permesso <permission> che si vuole aggiungere tra: "read" o "readwrite".
- Digitare nella console del client il comando per aggiungere permessi ad un utente utilizzando la sintassi: addpermission <username> <database> <permission>
- Eseguire il comando tramite il tasto invio.

3.4.21 Test TV21: Rimozione permessi

L'utente vuole verificare la funzionalità dell'amministratore per rimuovere permessi ad un utente. Sono richiesti i seguenti passaggi:

- Scegliere il nome <username> dell'utente a cui si vogliono rimuovere i permessi.
- Scegliere un database <database> sul quale rimuovere i permessi all'utente.
- Digitare nella console del client il comando per rimuovere i permessi ad un utente utilizzando la sintassi: removepermission <username> <database>
- Eseguire il comando tramite il tasto invio.

3.4.22 Test TV22: Rimozione utente

L'utente vuole verificare la funzionalità dell'amministratore per rimuovere un utente. Sono richiesti i seguenti passaggi:

- Scegliere il nome <username> per l'utente che si vuole aggiungere.
- Digitare nella console del client il comando per rimuovere un utente utilizzando la sintassi: removeuser <username>
- Eseguire il comando tramite il tasto invio.

3.4.23 Test TV23: Impostazione Ninja

L'utente vuole verificare la funzionalità dell'amministratore per impostare il numero di attori di tipo Ninja per ogni attore di tipo Storekeeper. Sono richiesti i seguenti passaggi:

- Scegliere il numero <number> di attori Ninja per ogni attore di tipo Storekeeper.
- Digitare nella console del client il comando per impostare il numero di ninja utilizzando la sintassi: setninja <number>
- Eseguire il comando tramite il tasto invio.

3.4.24 Test TV24: Impostazione Warehouseman

L'utente vuole verificare la funzionalità dell'amministratore per impostare il numero di attori di tipo Warehouseman per ogni attore di tipo Storekeeper Sono richiesti i seguenti passaggi:

- Scegliere il numero <number> di attori Warehouseman per ogni attore di tipo Storekeeper.
- Digitare nella console del client il comando per impostare il numero di warehouseman utilizzando la sintassi: setwarehouseman <number>
- Eseguire il comando tramite il tasto invio.

3.4.25 Test TV25: Impostazione item

L'utente vuole verificare la funzionalità dell'amministratore per impostare il numero massimo di item per le mappe degli attori Storekeeper. Sono richiesti i seguenti passaggi:

- Scegliere il numero <number> di item massimo per la mappa gestita da un attore di tipo Storekeeper.
- Digitare nella console del client il comando per impostare il numero di item utilizzando la sintassi: setmaxrow <number>
- Eseguire il comando tramite il tasto invio.

3.4.26 Test TV26: Impostazione Storekeeper

L'utente vuole verificare la funzionalità per impostare il numero massimo di attori di tipo Storekeeper per un attore di tipo Storefinder. Sono richiesti i seguenti passaggi:

- Scegliere il numero <number> di Storekeeper massimo per ogni Storefinder.
- Digitare nella console del client il comando per impostare il numero di Storekeeper utilizzando la sintassi: setmaxstorekeeper <number>
- Eseguire il comando tramite il tasto invio.

3.4.27 Test TV27: Impostazione Storefinder

L'utente vuole verificare la funzionalità per impostare il numero massimo di attori di tipo Storefinder per un attore di tipo Storemanager. Sono richiesti i seguenti passaggi:

- Scegliere il numero <number> di Storefinder massimo per ogni Storemanager.
- Digitare nella console del client il comando per impostare il numero di Storefinder utilizzando la sintassi: setmaxstorefinder <number>
- Eseguire il comando tramite il tasto invio.

3.5 Pianificazione dei test

Di seguito saranno descritti i test di sistema, integrazione e unità che sono previsti. Nelle tabelle sottostanti la sigla N.E sta per non eseguito.

3.5.1 Test di sistema

Viene mostrata di seguito la tabella con tutti i test di sistema che consentono di verificare il sistema rispetto ai requisiti descritti nell'*Analisi dei Requisiti v1.3.11*. Per ogni comando testato sarà anche verificato il corretto riconoscimento e la corretta gestione degli errori. Ogni test di sistema è identificato da TS ed il numero del Caso d'uso ad esso associato.

Test	Descrizione	Stato	Requisiti
TS1	Viene verificato che il sistema consenta di connettersi al server tramite l'utilizzo del client	N.E	R[1.3][N][F], R[2.1][N][F], R[3.1][N][F]
TS2	Viene verificato il corretto funzionamento del comando di `HELP` generico e di `HELP` specifico	N.E	R[2.3][D][F]
TS3.1	Viene verificato che il comando `listmap` restituisca correttamente la lista dei database	N.E	R[1.4.1][D][F], R[2.4][D][F], R[3.3][D][F]
TS3.4	Viene verificato che il comando di creazione di un nuovo database funzioni correttamente	N.E	R[1.4.4][N][F], R[2.7][N][F], R[3.4][N][F]
TS3.5	Viene verificato che il comando di eliminazione di un database funzioni correttamente	N.E	R[1.4.5][N][F], R[2.8][N][F], R[3.5][N][F]
TS3.6	Viene verificato che il comando di rinomina rinomini correttamente il database scelto	N.E	R[1.4.6][D][F], R[2.9][D][F], R[3.6][D][F]
TS3.7	Viene verificato che il comando di selezione di un database effettui correttamente la selezione del database scelto	N.E	R[1.4.7][N][F], R[2.10][N][F],R[3.7][N][F]
TS4.1	Viene verificato che il comando `SHOW` mostri correttamente la lista delle mappe	N.E	R[1.5.1][D][F], R[2.11][D][F], R[3.8][D][F]

Test	Descrizione	Stato	Requisiti
TS4.2	Viene verificato che il comando di creazione mappa crei correttamente una mappa all'interno del database selezionato	N.E	R[1.5.2][N][F], R[2.12][N][F], R[3.9][N][F]
TS4.3	Viene verificato che il comando di eliminazione mappa rimuova correttamente la mappa scelta dal database selezionato	N.E	R[1.5.3][N][F], R[2.13][N][F], R[3.10][N][F]
TS4.4	Viene verificato che il comando di rinomina mappa rinomini correttamente la mappa scelta nel database selezionato	N.E	R[1.5.4][D][F], R[2.14][D][F], R[3.11][D][F]
TS4.5	Viene verificato che il comando di selezione di una mappa selezioni correttamente la mappa scelta	N.E	R[1.5.5][N][F], R[2.15][N][F], R[3.12][N][F]
TS5.1	Viene verificato che il comando di visualizzazione chiavi mostri correttamente le chiavi della mappa selezionata	N.E	R[1.6.1][D][F], R[2.18][D][F], R[3.13][D][F]
TS5.2	Viene verificato che il comando di ricerca per chiave restituisca il valore associato corretto	N.E	R[1.6.2][N][F], R[2.19][N][F], R[3.14][N][F]
TS5.3	Viene verificato che il comando di inserimento di un item inserisca la coppia chiave-valore corretta nella mappa selezionata	N.E	R[1.6.3][N][F], R[2.20][N][F], R[3.15][N][F]
TS5.4	Viene verificato che il comando di aggiornamento di un item modifichi correttamente il valore associato alla chiave scelta	N.E	R[1.6.4][N][F], R[2.21][N][F], R[3.16][N][F]
TS5.6	Viene verificato che il comando di rimozione di un item elimini correttamente la coppia chiave-valore scelta dalla mappa selezionata	N.E	R[1.6.6][N][F], R[2.23][N][F], R[3.18][N][F]
TS6	Viene verificato che il comando di disconnessione del client dal server funzioni correttamente	N.E	R[2.2][N][F], R[3.2][N][F]
TS8	Viene verificato l'interfaccia client si chiuda correttamente	N.E	R[2.24][N][F]
TS9	Viene verificato che il server si configuri correttamente	N.E	R[1.1], R[1.2.1][N][F], R[1.2.2][N][F], R[1.2.3][D][F], R[1.2.4][D][F], R[1.2.6][N][F], R[1.2.7][N][F]
TS14	Viene verificato che l'arresto del server funzioni correttamente senza alcuna perdita di dati o strutture	N.E	R[1.2.5][N][F]
TS17	Viene verificato che il sistema funzioni correttamente sulle macchine che dispongono la JVM aggiornata alla versione 8 o successive	N.E	R[5][N][V]

Tabella 9: Test di sistema con requisiti associati

Requisito	Test
R[1.1][N][F]	TS9
R[1.2.1][N][F]	TS9
R[1.2.2][N][F]	TS9
R[1.2.3][N][F]	TS9
R[1.2.4][D][F]	TS9
R[1.2.5][N][F]	TS14
R[1.2.6][N][F]	TS9

Requisito	Test
R[1.2.7][N][F]	TS9
R[1.3][N][F]	TS1
R[1.4.1][D][F]	TS3.1
R[1.4.4][N][F]	TS3.4
R[1.4.5][N][F]	TS3.5
R[1.4.6][D][F]	TS3.6
R[1.4.7][N][F]	TS3.7
R[1.5.1][D][F]	TS4.1
R[1.5.2][N][F]	TS4.2
R[1.5.3][N][F]	TS4.3
R[1.5.4][D][F]	TS4.4
R[1.5.5][N][F]	TS4.5
R[1.6.1][D][F]	TS5.1
R[1.6.2][N][F]	TS5.2
R[1.6.3][N][F]	TS5.3
R[1.6.4][N][F]	TS5.4
R[1.6.6][N][F]	TS5.6
R[2.1][N][F]	TS1
R[2.2][N][F]	TS6
R[2.3][D][F]	TS2
R[2.4][D][F]	TS3.1
R[2.7][N][F]	TS3.4
R[2.8][N][F]	TS3.5
R[2.9][D][F]	TS3.6
R[2.10][N][F]	TS3.7
R[2.11][D][F]	TS4.1
R[2.12][N][F]	TS4.2
R[2.13][N][F]	TS4.3
R[2.14][D][F]	TS4.4
R[2.15][N][F]	TS4.5
R[2.18][D][F]	TS5.1
R[2.19][N][F]	TS5.2
R[2.20][N][F]	TS5.3
R[2.21][N][F]	TS5.4
R[2.23][N][F]	TS5.6
R[2.24][N][F]	TS8
R[3.1][N][F]	TS1
R[3.2][N][F]	TS6
R[3.3][D][F]	TS3.1
R[3.4][N][F]	TS3.4
R[3.5][N][F]	TS3.5
R[3.6][D][F]	TS3.6
R[3.7][N][F]	TS3.7
R[3.8][D][F]	TS4.1
R[3.9][N][F]	TS4.2
R[3.10][N][F]	TS4.3
R[3.11][D][F]	TS4.4
R[3.12][N][F]	TS4.5
R[3.13][D][F]	TS5.1
R[3.14][N][F]	TS5.2
R[3.15][N][F]	TS5.3
R[3.16][N][F]	TS5.4
R[3.18][N][F]	TS5.6
R[5][N][V]	TS17

Requisito	Test

Tabella 10: Requisiti associati ai test

3.5.2 Test di integrazione

Viene mostrata di seguito la tabella contenente tutti i test di integrazione, atti a verificare il corretto funzionamento delle singole componenti del sistema descritte nella *Specifica Tecnica v2.0.0*.

Test	Descrizione	Stato	Componente
TI.server	Viene verificato che il Server riceva i comandi e restituisca output corretti. Vengono anche verificati l'avvio e l'arresto	Success	server
TI.utils	Viene verificato che le le classi di utilità interagiscano correttamente con il resto del sistema	Success	utils
TI.actors	Viene verificato che all'interno del sistema di attori i messaggi vengano scambiati ed elaborati correttamente	Success	actors
TI.driver	Viene verificato che il driver si connetta correttamente al server e che invii le stringe di query composte secondo la grammatica corretta	Success	driver
TI.client	Viene verificato che all'apertura del client venga mostrato correttamente il messaggio di benvenuto, e che sia messa a disposizione una shell che permetta di inserire i comandi da inviare al Driver e ne stampi la risposta corrispondente	Success	client

Tabella 11: Test di integrazione con componente associata

Viene mostrata la tabella che traccia ogni requisito al test.

3.5.3 Test di unità

Viene mostrata di seguito la tabella contenente tutti i test di unità.

Test	Descrizione	Metodi	Stato

TU1	Si verifica che il Server carichi correttamente l'albero degli attori da disco.	server.Server.{
		loadUsers()
		loadUsersPermissions()
		loadDatabases()
		}
			Success

TU2	Si verifica che il Server all'accensione faccia partire l'ActorSystem, il sistema di logging e che instanzi i Doorkeeper in ascolto sulle porte impostate	server.Server.main()	Success

TU3	Si verifica che il parser riconosca correttamente il comando di login (login <username> <password>) e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		getMatch()
		}
			Success

TU4	Si verifica che il parser riconosca correttamente i comandi senza parametri (ovvero: listdb, listmap, keys, help, listuser) e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		parseCommandWithoutParam()
		getMatch()
		}
			Success

Test	Descrizione	Metodi	Stato

TU5	Si verifica che il parser riconosca correttamente i comandi con un parametro (ovvero: selectdb <DBName>, createdb <DBName>, deletedb <DBName>, selectmap <mapName>, createmap <mapName>, deletemap <mapName>, help <commandName>, removeuser <userName>, listpermissions <userName>, setninja <number>, setwarehousemen <number>, setmaxrow <number>, setmaxstorekeeper <number>, setmaxstorefinder <number>) e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		parseCommandWithParam()
		getMatch()
		}
			Success

TU6	Si verifica che il parser riconosca correttamente i comandi con un parametro che può essere composto da più parole separate da spazio (ovvero: find '<key>', remove '<key>') e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		parseRowCommandOneParam()
		getMatch()
		}
			Success

TU7	Si verifica che il parser riconosca correttamente i comandi a due parametri di cui il primo possono essere formati da più di una parola (ovvero: insert '<key>' <value>, update '<key>' <value>) e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		parseRowCommandTwoParams()
		getMatch()
		}
			Success

TU8	Si verifica che il parser riconosca correttamente i comandi con due parametri (ovvero: adduser <username> <password>, removepermission <username> <DBName>) e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		parseCommandsWithTwoParams()
		getMatch()
		}
			Success

TU9	Si verifica che il parser riconosca correttamente i comandi con tre parametri (ovvero: addpermission <userName> <DBName> <permissionType>) e ritorni il corretto messaggio	server.ultil.Parser.{
		parseQuery()
		parseCommandWithThreeParams()
		getMatch()
		}
			Success

TU10	Si verifica che l'attore Doorkeeper effettui correttamente il Bind con l'attore IO.		Success

TU11	Si verifica che l'attore Doorkeeper riceva correttamente il messaggio di bonud è produca il log corretto.	server.actors.Doorkeeper{
		receive()
		}
			Success

TU12	Si verifica che l'attore Doorkeeper riceva il messaggio di Connected, e che crei correttamente un attore Usermanager che gestisca la connessione.	server.actors.Doorkeeper{
		recive()
		}
			Success

Test	Descrizione	Metodi	Stato

TU13	Si verifica che l'attore Main riceva i messaggi di tipo ListDatabaseMessage e ritorni la corretta lista dei DataBase, inoltre si verifica che in caso di DataBase inesistente ritorni un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleDatabaseMessage()
		isValidStoremanager()
		}
			Success

TU14	Si verifica che l'attore Main riceva i messaggi di tipo SelectDatabaseMessage e imposti correttamente nelle sue proprietà il nuovo database selezionato, inoltre si verifica che in caso di DataBase inesistente lasci le sue proprietà invariate e ritorni un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleDatabaseMessage()
		isValidStoremanager()
		}
			Success

TU15	Si verifica che l'attore Main riceva i messaggi di tipo CreateDatabaseMessage, crei una nuova coppia nomeDatabase-Storemanager e la inserisca correttamente nella mappa presente nel server, inoltre si verifica che in caso di DataBase già esistente non ne crei uno nuovo con lo stesso nome e ritorni un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleDatabaseMessage()
		isValidStoremanager()
		}
			Success

TU16	Si verifica che l'attore Main riceva i messaggi di tipo DeleteDatabaseMessage e che effettivamente il DataBase venga rimosso dalla mappa presente nel server, inoltre si verifica che nel caso il database non esista ritorni un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleDatabaseMessage()
		isValidStoremanager()
		}
			Success

TU17	Si verifica che l'attore Main riceva i messaggi di tipo SelectMapMessage e imposti correttamente nelle sue proprietà la nuova mappa selezionata, inoltre si verifica che nel caso non sia stato precedentemente selezionato un DataBase o la mappa sia inesistente lasci le sue proprietà invariate e ritorni un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleMapMessage()
		}
			Success

TU18	Si verifica che l'attore Main riceva i messaggi di tipo CompleteHelp e SpecificHelp e ritorni la corretta descrizione del comando.	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleHelpMessage()
		}
			Success

TU19	Si verifica che l'attore Main riceva i messaggi di tipo RowMessage (ovvero: InsertRowMessage, UpdateRowMessage, RemoveRowMessage, FindRowMessagee e ListKeysMessage) e di tipo MapMessage (ovvero: CreateMapMessage, DeleteMapMessage, SelectMapMessage, ListMapMessage)e li inoltri al corretto Storefinder.	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleRowMessage()
		}
			Success

Test	Descrizione	Metodi	Stato

TU20	Si verifica, per ogni possibile messaggio utente che richiede i permessi di scrittura, inviato all'attore Main da un utente senza tali permessi non venga eseguito e venga ritornato un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleHelpMessage()
		handleDatabaseMessage()
		handleMapMessage()
		handleRowMessage()
		checkPermissions()
		}
			Success

TU21	Si verifica, per ogni possibile messaggio utente che richiede i permessi di lettura, inviato all'attore Main da un utente senza tali permessi non venga eseguito e venga ritornato un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleHelpMessage()
		handleDatabaseMessage()
		handleMapMessage()
		handleRowMessage()
		checkPermissions()
		}
			Success

TU22	Si verifica, per ogni possibile messaggio utente che richiede i permessi di amministratore, inviato all'attore Main da un utente senza tali permessi non venga eseguito e venga ritornato un messaggio d'errore	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleHelpMessage()
		handleDatabaseMessage()
		handleMapMessage()
		handleRowMessage()
		checkPermissions()
		}
			Success

TU23	Si verifica, per ogni possibile messaggio utente che non richiede permessi, inviato all'attore Main effettivamente venga eseguito anche da un utente senza permessi.	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleUserMessage()
		handleHelpMessage()
		handleDatabaseMessage()
		handleMapMessage()
		handleRowMessage()
		checkPermissions()
		}
			Success

TU24	Si verifica che l'attore Main riceva i messaggi di tipo ListUserMessage e che ritorni la corretta lista di utenti	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleUserManagementMessage()
		handleRowMessage()
		}
			Success

TU25	Si verifica che l'attore Main riceva i messaggi di tipo AddUserMessage e che effettivamente aggiunga il nuovo utente	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleUserManagementMessage()
		handleRowMessage()
		}
			Success

Test	Descrizione	Metodi	Stato

TU26	Si verifica che l'attore Main riceva i messaggi di tipo RemoveUserMessage e che effettivamente rimuova l'utente indicato	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleUserManagementMessage()
		handleRowMessage()
		}
			Success

TU27	Si verifica che l'attore Main riceva i messaggi di tipo ListPermissionMessage e che ritorni la corretta lista di permessi	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handlePermissionsManagementMessage()
		}
			Success

TU28	Si verifica che l'attore Main riceva i messaggi di tipo AddPermissionMessage e che effettivamente aggiunga i permessi indicati per l'utente e la mappa specificati.	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handlePermissionsManagementMessage()
		}
			Success

TU29	Si verifica che l'attore Main riceva i messaggi di tipo RemovePermissionMessage e che effettivamente rimuova i permessi indicati per l'utente e la mappa specificati.	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handlePermissionsManagementMessage()
		}
			Success

TU30	Si verifica che l'attore Main riceva i messaggi di tipo MaxRowsMessage e che imposti correttamente il massimo numero di righe per ogni Storekeeper.	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleActorPropertiesMessageMessage()
		}
			Success

TU31	Si verifica che l'attore Main riceva i messaggi di tipo MaxStorekeeperMessage e che imposti il massimo numero di Storekeeper per ogni Storefinder	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleActorPropertiesMessageMessage()
		}
			Success

TU32	Si verifica che l'attore Main riceva i messaggi di tipo MaxStorefinderMessage e che imposti il massimo numero di Storefinder per ogni Storemanager	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleActorPropertiesMessageMessage()
		}
			Success

TU33	Si verifica che l'attore Main riceva i messaggi di tipo SetNinjaMessage e che imposti il massimo numero di Ninja associati ad ogni Storekeeper	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleActorPropertiesMessageMessage()
		}
			Success

TU34	Si verifica che l'attore Main riceva i messaggi di tipo SetWarehousemanMessage e che imposti il massimo numero di Warehouseman associati ad ogni Storekeeper	sever.actors.Main.{
		receive()
		handleQueryMessage()
		handleAdminMessage()
		handleActorPropertiesMessageMessage()
		}
			Success

Test	Descrizione	Metodi	Stato
TU35	Si verifica che l'attore Storefinder riceva solo messaggi di tipo RowMessage e, nel caso ne riceva di altro tipo, inserisca nel log un opportuno messaggio d'errore	server.actors.Storefinder.receive()	Success

TU36	Si verifica che l'attore Storefinder riceva i messaggi di tipo ListKeysMessage, inoltri lo stesso messaggio ad ogni suo Storekeeper e ritorni la concatenazione delle risposte, si verifica inoltre che ritorni un messaggio speciale nel caso in cui la mappa sia vuota	server.actors.Storefinder.{
		receive()
		handleRowMessage()
		reply()
		}
			Success

TU37	Si verifica che l'attore Storefinder riceva i messaggi di tipo InsertRowMessage, UpdateRowMessage, RemoveRowMessage, FindRowMessage e li inoltri al corretto Storekeeper	server.actors.Storefinder.{
		receive()
		sendToStorekeeper()
		findActor()
		reply()
		}
			Success

TU38	Si verifica che l'attore Storefinder riceva i messaggi di tipo SetNinjaMessage, SetWarehousemanMessage, MaxRowsMessage, MaxStorekeeperMessage ed imposti coerentemente le sue proprietà.	server.actors.Storefinder.{
		receive()
		sendToStorekeeper()
		findActor()
		reply()
		}
			Success

TU39	Si verifica che l'attore Storekeeper riceva i messaggi di tipo InsertRowMessage e che effettivamente inserisca la coppia chiave-valore all'interno della mappa, inoltre si verifica che se la chiave è già presente non inserisca nulla e ritorni un messaggio d'errore	server.actors.Storekeeper.{
		receive()
		reply()
		}
			Success

TU40	Si verifica che l'attore Storekeeper riceva i messaggi di tipo UpdateRowMessage e che effettivamente aggiorni il valore della chiave indicata, inoltre si verifica che se la chiave non è presente nella mappa ritorni un messaggio d'errore	server.actors.Storekeeper.{
		receive()
		logAndReply()
		exists()
		}
			Success

TU41	Si verifica che l'attore Storekeeper riceva i messaggi di tipo RemoveRowMessage e che effettivamente rimuova la riga di mappa indicizzata dalla chiave indicata, inoltre si verifica che se la chiave non è presente non rimuova nulla e ritorni un messaggio d'errore	server.actors.Storekeeper.{
		receive()
		logAndReply()
		exists()
		}
			Success

TU42	Si verifica che l'attore Storekeeper riceva i messaggi di tipo FindRowMessage e che ritorni il valore indicizzato dalla chiave indicata	server.actors.Storekeeper.{
		receive()
		reply()
		exists()
		}
			Success

Test	Descrizione	Metodi	Stato

TU43	Si verifica che l'attore Storekeeper riceva i messaggi di tipo ListKeysMessage e che effettivamente ritorni la lista della chiavi presenti nella mappa selezionata	server.actors.Storekeeper.{
		receive()
		reply()
		}
			Success

TU44	Si verifica che l'attore Storekeeper riceva i messaggi di tipo AddNinjaMessage e che aggiunga alla sua lista di Ninja quello contenuto nel messaggio.	server.actors.Storekeeper.{
		receive()
		handleLinkMessage()
		reply()
		}
			Success

TU45	Si verifica che l'attore Storekeeper riceva i messaggi di tipo RemoveNinjaMessage e che rimuova dalla sua lista di Ninja quello contenuto nel messaggio.	server.actors.Storekeeper.{
		receive()
		handleLinkMessage()
		reply()
		}
			Success

TU46	Si verifica che l'attore Storekeeper riceva i messaggi di tipo AddWarehousemanMessage e che aggiunga alla sua lista di Warehouseman quello contenuto nel messaggio.	server.actors.Storekeeper.{
		receive()
		handleLinkMessage()
		reply()
		}
			Success

TU47	Si verifica che l'attore Storekeeper riceva i messaggi di tipo RemoveWarehousemanMessage e che rimuova dalla sua lista di Warehouseman quello contenuto nel messaggio.	server.actors.Storekeeper.{
		receive()
		handleLinkMessage()
		reply()
		}
			Success

TU48	Si verifica che l'attore Storekeeper (che agisce da Ninja) riceva i messaggi di tipo BecomeStorekeeperMessage e che effettivamente diventi uno Storekeeper.	server.actors.Storekeeper.{
		receive()
		receiveAsStorekeeper()
		}
			Success

TU49	Si verifica che l'attore Storekeeper (che agisce da Ninja) riceva i messaggi di tipo InsertRowMessage e che effettivamente inserisca la coppia chiave-valore all'interno della mappa, inoltre si verifica che se la chiave è già presente non inserisca nulla e ritorni un messaggio d'errore	server.actors.Storekeeper.{
		receive()
		reply()
		}
			Success

TU50	Si verifica che l'attore Storekeeper (che agisce da Ninja) riceva i messaggi di tipo UpdateRowMessage e che effettivamente aggiorni il valore della chiave indicata, inoltre si verifica che se la chiave non è presente nella mappa ritorni un messaggio d'errore	server.actors.Storekeeper.{
		receive()
		logAndReply()
		exists()
		}
			Success

TU51	Si verifica che l'attore Storekeeper (che agisce da Ninja) riceva i messaggi di tipo RemoveRowMessage e che effettivamente rimuova la riga di mappa indicizzata dalla chiave indicata, inoltre si verifica che se la chiave non è presente non rimuova nulla e ritorni un messaggio d'errore	server.actors.Storekeeper.{
		receive()
		logAndReply()
		exists()
		}
			Success

Test	Descrizione	Metodi	Stato

TU52	Si verifica che l'attore Storemanager riceva i messaggi di tipo AskMapMessage e risponda al corretto storefinder con il nome della mappa	server.actors.Storemanager.{
		receive()
		}
			Success

TU53	Si verifica che l'attore Storemanager riceva i messaggi di tipo ListMapMessage e che ritorni la lista delle mappe presenti, inoltre si verifica che ritorni un messaggio speciale in cui non sia presente alcuna mappa	server.actors.Storemanager.{
		receive()
		handleMapMessage()
		reply()
		}
			Success

TU54	Si verifica che l'attore Storemanager riceva i messaggi di tipo CreateMapMessage e che effettivamente aggiunga la mappa indicata, inoltre si verifica che nel caso la mappa esista già non aggiunga la mappa indicata e ritorni un messaggio d'errore	server.actors.Storemanager.{
		receive()
		handleMapMessage()
		logAndReply()
		reply()
		}
			Success

TU55	Si verifica che l'attore Storemanager riceva i messaggi di tipo DeleteMapMessage e che effettivamente rimuova la mappa indicata, inoltre si verifica che nel caso la mappa non esista non la rimuova ritorni un messaggio d'errore	server.actors.Storemanager.{
		receive()
		handleMapMessage()
		logAndReply()
		reply()
		}
			Success

TU56	Si verifica che l'attore Storemanager riceva i messaggi di tipo StorefinderRowMessage, che lo inoltri al corretto Storefinder e che una volta ricevuta una risposta ne ritorni il risultato, si verifica inoltre che se la mappa indicata non esiste non faccia nulla e ritorni un messaggio d'errore	server.actors.Storemanager.{
		receive()
		handleRowMessage()
		reply()
		}
			Success

TU57	Si verifica che l'attore Usermanager riceva stringhe di byte e le invii al parser se il primo byte è uguale ad 1, ritorni un messaggio d'errore altrimenti	server.actors.Usermanager.{
		receive()
		}
			Success

TU58	Si verifica che l'attore Usermanager riceva i messaggi di tipo InvalidQueryMessage e risponda con un messaggio d'errore	server.actors.Usermanager.{
		receive()
		parseQuery()
		reply()
		}
			Success

TU59	Si verifica che l'attore Usermanager riceva i messaggi di tipo LoginMessage ed inserisca nell'ActorSystem un nuovo attore Main con i permessi associati all'username e password indicati	server.actors.Usermanager.{
		receive()
		parseQuery()
		handleQueryMessage()
		reply()
		handleLogin()
		}
			Success

Test	Descrizione	Metodi	Stato

TU60	Si verifica che l'attore Usermanager riceva i messaggi di tipo QueryMessage diversi da LoginMessage e li inoltri al proprio attore Main, inoltre si verifica che nel caso l'utente a questo punto non sia connesso l'Usermanager ritorni un messaggio di errore	server.actors.Usermanager.{
		receive()
		parseQuery()
		handleQueryMessage()
		reply()
		}
			Success

TU61	Si verifica che l'attore Warehouseman riceva i messaggi di tipo RowMessage ed aggiorni coerentemente i dati sul disco.	server.actors.Warehouseman.{
		receive()
		handleRowMessage()
		replyAndLog()
		reply()
		}
			Success

TU62	Si verifica che il Driver crei correttamente una connessione e la ritorni	driver.Driver.connect(host:String, port:Integer, username:String, password:String)	Success
TU63	Si verifica che la Connessione esegua correttamente il login al Server	driver.ConcreteConnection.login(username:String, password:String)	Success
TU64	Si verifica che la Connessione si chiuda correttamente	driver.ConcreteConnection. closeConnection()	Success
TU65	Si verifica che la Connessione invii correttamente le stringhe dei comandi al Server	driver.ConcreteConnection. executeQuery(query:String)	Success
TU66	Si verifica che il Client riconosca il comando di connessione e invii la richiesta corretta al Driver.	client.Client.checkLogin(ln:String)	Success
TU67	Si verifica che il Client (connesso) invii correttamente le stringhe dei comandi alla connessione.	client.Client.executeLine(ln:String)	Success
TU68	Si verifica che il Client riconosca il comando di disconnessione e chiuda la connessione.	client.Client.executeLine(ln:String)	Success
TU69	Si verifica che il Client riconosca il comando di uscita ed effettivamente termini l'applicazione.	client.Client.executeLine()	Success

TU70	Si verifica che il FileReader carichi correttamente da file la struttura dell'albero degli attori.	server.util.FileReader.{
		loadActorTree()
		}
			Success

TU71	Si verifica che il FileReader scriva correttamente su file la struttura dell'albero degli attori.	server.util.FileReader.{
		dumpActorTree()
		}
			Success

TU72	Si verifica che il FileReader carichi correttamente da file i dati.	server.util.FileReader.{
		loadData()
		}
			Success

TU73	Si verifica che il FileReader scriva correttamente su file i dati.	server.util.FileReader.{
		dumpData()
		}
			Success

Test	Descrizione	Metodi	Stato
TU74	Si verifica che l'Helper ritorni correttamente l'aiuto generico.	server.util.Helper.{ completeHelp() }	Success
TU75	Si verifica che l'Helper ritorni correttamente l'aiuto specifico del comando indicato.	server.util.Helper.{ specificHelp() }	Success

Tabella 12: Test di unità

3.6 Lista errori frequenti

- **Norme stilistiche:**
  - Nome del documento: non viene utilizzata la macro predisposta;
  - Versione del documento in prima pagina errata;
  - Immagini mancanti;
  - Spazi lasciati vuoti per aggiunte successive e non rimossi;
  - Mancanza di uniformità delle espressioni all'interno dello stesso documento;
  - Mancanze nella sezione dei riferimenti.
- **Italiano:**
  - Doppie;
  - Accenti.
- **LaTeX:**
  - mancanza dell'indice delle immagini e delle tabelle.
- **UML:**
  - incongruenze tra l'immagine contenente i diagrammi e la descrizione testuale della stessa;
  - errori nel testo delle immagini dovute a copia-incolla.

Elenco delle figure

Elenco delle tabelle