UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

DOCUMENTO DI SPECIFICA DEI REQUISITI

PER

Sistema di gestione issue in progetti software

Authors:
Ubaldo Musto N86005304
Giandomenico Iameo
N86002856

Supervisor:
Prof. Porfirio TRAMONTANA
Co-Supervisor:
Prof. Bernardo BREVE

Versione 1.0
Prima versione del prodotto software, contenente le funzionalità di base per la gestione collaborativa di issue in progetti software

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Corso di Laurea Triennale in Informatica

18 ottobre 2025

Indice

Acronimi e abbreviazioni

API	**A**pplication **P**rogramming **I**nterface
CASE	**C**omputer **A**ided **S**oftware **E**ngineering
REST	**RE**presentational **S**tate **T**ransfer
ANSI	**A**merican **N**ational **S**tandards **I**nstitute
IEEE	**I**nstitute **E**lectrical **E**lectronics **E**ngineers
SRS	**S**oftware **R**equirements **S**pecification

Lista delle definizioni

Utente	Si riferisce a qualsiasi utente del sistema e può essere sia *interno* che *esterno*.
Template di Cockburn	È una rappresentazione completa di un caso d'uso. Descrive in modo dettagliato tutti i passaggi necessari all'esecuzione di un caso d'uso e include sezioni di supporto come *precondizioni* e *postcondizioni*.
Mock-up	È un *artefatto*, una rappresentazione visiva statica utilizzata per descrivere un sottoinsieme delle funzionalità del sistema. Il suo fine è puramente illustrativo o espositivo.
Issue	È una segnalazione di un problema che è stato riscontrato da un membro del team di sviluppo. In altre parole, è un modo per avvisare il team che qualcosa dovrebbe essere sistemato.
Feature	In Informatica, si riferisce a una funzionalità o capacità specifica che il software offre ai propri utilizzatori. Nel contesto del prodotto software preso in esame, è un tipo di segnalazione (*issue*) usata per suggerire una nuova funzionalità.
Bug	In Italiano anche detto *Baco*, nell'ambito della programmazione Informatica, è un'anomalia che porta a un malfunzionamento di un software, producendo un risultato inatteso o errato.
Stakeholder	Gli *stakeholder* (soggetti interessati o persone coinvolte) sono tutte quelle persone che hanno un interesse nei confronti di un'organizzazione e che influenzano e/o sono influenzate dalle sue attività.
Caso d'uso	Si riferisce alla descrizione del comportamento del sistema quando riceve uno stimolo da parte di un attore. In UML, un caso d'uso è graficamente rappresentato da un ovale.
Attore	Specifica un ruolo (o un insieme di ruoli) assunto da un utente o altra entità che interagisce con il sistema.
Container	È una tecnologia che consente di creare un ambiente isolato, simulando un ambiente operativo, in cui le applicazioni possono essere eseguite senza preoccuparsi dei conflitti con le altre.

Linguaggi Object-Orientend	Sono linguaggi appartenenti al paradigma computazionale *Object-Oriented*. Questo pone al centro delle attività la modellazione di concetti del mondo reale e le loro relazioni che definiscono il problema che si vuole risolvere.
CASE	Significa *Computer-Aided Software Engineering* (ingegneria del software assistita dal computer) e comprende una vasta gamma di programmi diversi per aiutare le attività di processo del software.
Back-end	È la parte di un sistema informatico che gestisce il funzionamento di un'applicazione o sito web. Comprende tutto ciò che avviene dietro le quinte: elaborazione dei dati, logica del sistema, comunicazione della base di dati.
Front-end	È la parte di un'applicazione o di un sito web visibile e accessibile all'utente, ovvero l'interfaccia con cui l'utente interagisce.
Docker	È una delle piattaforme software container più utilizzate e diffuse nell'ambito della virtualizzazione a livello di sistema operativo.

Capitolo 1

Introduzione

1.1 Scopo del documento dei requisiti

Il presente *SRS* si propone di accogliere in forma organica una descrizione di tutte le specifiche tecniche e funzionali che caratterizzeranno il sistema per la gestione di issue in progetti software.
Esso è stato redatto seguendo le indicazioni dello standard IEEE/ANSI 830-1998 (IEEE 1998) per i documenti dei requisiti.

1.2 Scopo del prodotto

Il prodotto software che si vuole realizzare prende il nome di BugBoard26, una piattaforma che consentirà a un team di sviluppo di segnalare problemi relativi a un progetto, monitorare lo stato, assegnarli a membri del team e tenere traccia delle attività di risoluzione. Il sistema deve fornire le seguenti funzionalità:

1. Il sistema deve fornire un meccanismo di autenticazione semplice e sicuro, basato su email e password. Le informazioni gestite dall'applicazione sono critiche per l'azienda, ed è fondamentale preservarne l'integrità e la segretezza. Il sistema viene fornito con un account da amministratore già attivo, con credenziali di default. Un amministratore può creare ulteriori utenze, specificando una email, una password, e indicando se quell'utenza sarà "normale" oppure "di amministrazione".

2. Tutti gli utenti autenticati possono segnalare una issue indicando almeno un titolo e una descrizione. Alcuni utenti potrebbero voler specificare anche una priorità e sarebbe gradita la possibilità di allegare un'immagine. Le issue possono essere di diverso tipo: question (per richieste di chiarimenti), bug (per segnalare malfunzionamenti), documentation (per segnalare problemi relativi alla documentazione), e feature (per indicare la richiesta o il suggerimento di nuove funzionalità). Le issue create sono inizialmente nello stato "todo".

3. Il sistema deve offrire una vista riepilogativa delle issue, con la possibilità di filtrare o ordinare i risultati in base a criteri come tipologia, stato, priorità o altri parametri rilevanti.
4. Ogni bug dovrebbe poter essere assegnato (da un amministratore) a un membro del team. Quando viene assegnato un bug a un utente, quest'ultimo riceve una notifica. Gli utenti possono visualizzare i bug loro assegnati.
5. È necessaria una funzione di ricerca che consenta di trovare bug in base a parole chiave.
6. Il sistema dovrebbe suggerire automaticamente a chi assegnare un bug, basandosi sul carico di lavoro corrente degli utenti.

1.3 Riferimenti

Qui di seguito sono elencati documenti e indirizzi web a cui fa riferimento la presente specifica dei requisiti:

- Slide del corso di Ingegneria del software 2025/2026 fornite dai professori P.Tramontana e B.Breve.
- Ian, Sommerville. *Ingegneria del software*. 8ª edizione, Pearson Paravia Bruno Mondadori S.p.A., 2007.
- Modello IEEE/ANSI 830-1998 per strutturare il documento dei requisiti.
- Enciclopedia online Wikipedia.

1.4 Descrizione del resto del documento

La restante parte del presente documento è suddivisa in due ulteriori capitoli che forniscono una descrizione sempre più dettagliata delle funzionalità del prodotto software in questione. Il capitolo immediatamente successivo, *Descrizione generale*, è composto dai seguenti paragrafi:

1. *Prospettiva del prodotto:* descrive il contesto o ambiente operativo in cui sarà utilizzato il prodotto software in questione. Verrà specificata l'architettura generale, se il prodotto è un membro di una famiglia di prodotti, un sostituto di altri sistemi oppure un prodotto completamente autonomo.
2. *Funzioni del prodotto:* illustra i principali servizi (o funzionalità) del prodotto corredati da diagrammi. Il livello di dettaglio usato in questo paragrafo è a un alto livello; i dettagli delle funzionalità saranno forniti del capitolo 3.
3. *Caratteristiche utente:* esplicità le conoscenze richieste per poter utilizzare il prodotto in questione e fornisce una descrizione generale degli utilizzatori previsti dal sistema.

4. *Presupposti e dipendenze:* illustra le condizioni che il sistema presume siano verificate al momento del suo avvio, durante la sua esecuziione e dopo la terminazione. Verranno specificati anche *elementi esterni* (altri sistemi, persone o componenti) da cui il prodotto dipende e che sono necessari per il suo corretto funzionamento.

Nel terzo capitolo, *Requisiti specifici*, verranno illustrate le funzionalità che il prodotto software realizza. Il capitolo è composto dai seguenti paragrafi:

1. *Requisiti funzionali:* fornisce una descrizione dettagliata dei casi d'uso che verranno presentati nel capitolo 2. Tali requisiti indicano come il sistema dovrebbe reagire a particolari input o come dovrebbe comportarsi in particolari situazioni.
2. *Requisiti non funzionali:* esplicita i vincoli sui servizi o sulle funzioni offerti dal sistema. In questa sezione, i requisiti imposti sono stati divisi secondo la *classificazione di Sommerville*.

Capitolo 2

Descrizione generale

2.1 Prospettiva del prodotto

Il prodotto software sarà un'applicazione basata su un'architettura a due livelli, composta da due macro-componenti indipendenti: una parte di *back-end* e una di *front-end*.

Back-end
Avrà la responsabilità della gestione centralizzata dello stato del sistema, garantendo che tutte le informazioni e i dati persistenti siano conservati e aggiornati unicamente su di esso.

Front-end
Costituirà l'interfaccia utente che si appoggierà ai servizi offerti dal back-end esclusivamente attraverso la rete.

2.2 Funzioni del prodotto

In questo paragrafo sono stati elencati i requisiti funzionali utente, dedotti dalle funzionalità descritte nel primo capitolo, e descritti mediante i diagrammi di caso d'uso. Il paragrafo è suddiviso in due parti:

> Nella *prima parte* verrà fornito un elenco dei requisiti funzionali dedotti, espressi in modo comprensibile anche per gli utenti senza un'approfondita conoscenza tecnica.
>
> Nella *seconda parte* verranno presentati i diagrammi di caso d'uso, i quali saranno associati a uno o più requisiti.

2.2.1 Requisiti funzionali utente

RF1 **Il sistema deve offrire un meccanismo di autenticazione.**

RF2 **Il sistema deve poter offrire a un account superutente, già attivo per default, di creare ulteriori utenze.**

RF3 **Gli *utenti interni* possono segnalare issue, con la possibilità di specificare un titolo, una descrizione, una priorità, una immagine e il tipo.**

RF4 **Il sistema deve permettere la visualizzazione di tutte le issue segnalate, con la possibilità di filtrare e ordinare i risultati.**

RF5 **Il sistema deve permettere, solo a un *Amministratore*, di assegnare bug e notificare un membro del team.**

RF6 **Il sistema deve consentire a un *utente normale* di ricevere notifiche riguardanti i bug.**

RF7 **Ogni *utente normale* può visualizzare i propri bug assegnati.**

RF8 **Il sistema deve fornire una funzione di ricerca per trovare issue.**

RF9 **Il sistema dovrebbe suggerire automaticamente a chi assegnare un bug.**

2.2.2 Diagrammi di caso d'uso

Qui di seguito vengono illustrati i diagrammi di caso d'uso che modellano i requisiti sopra citati.

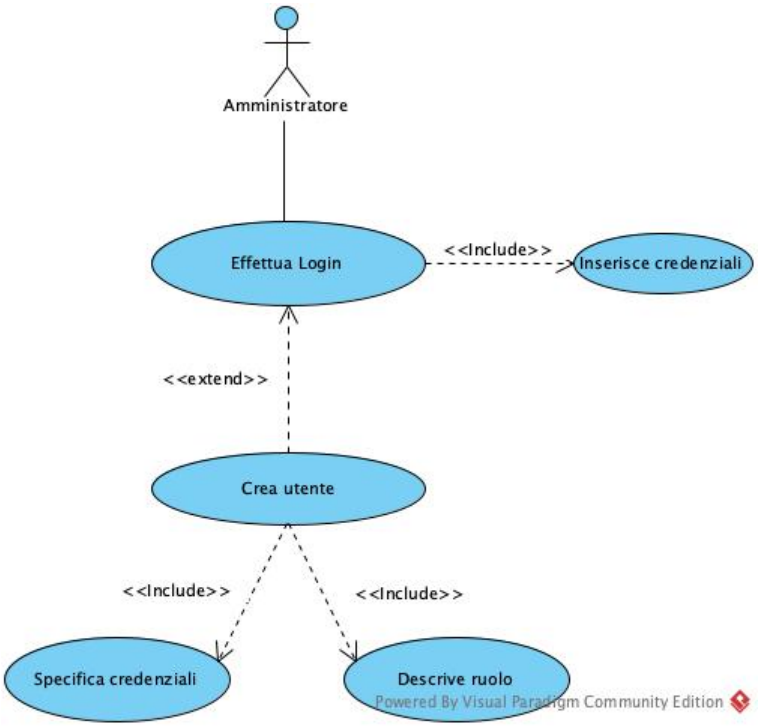

Figura 2.1: Modellazione requisiti *RF1*, *RF2*

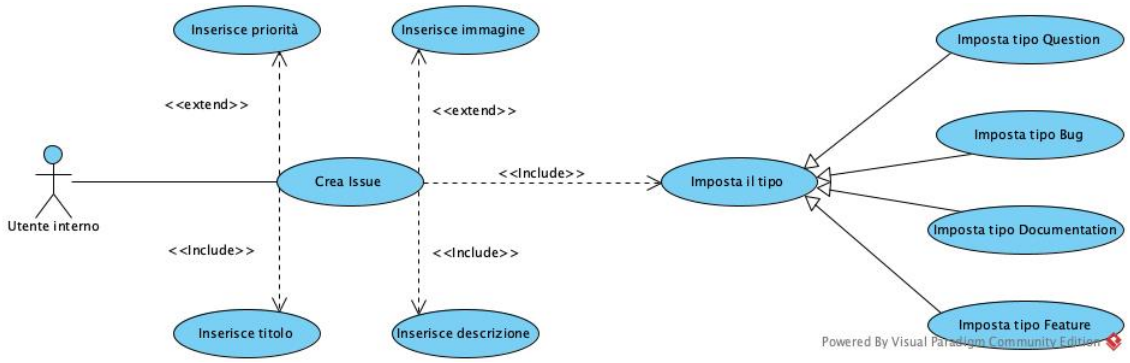

Figura 2.2: Modellazione requisito *RF3*

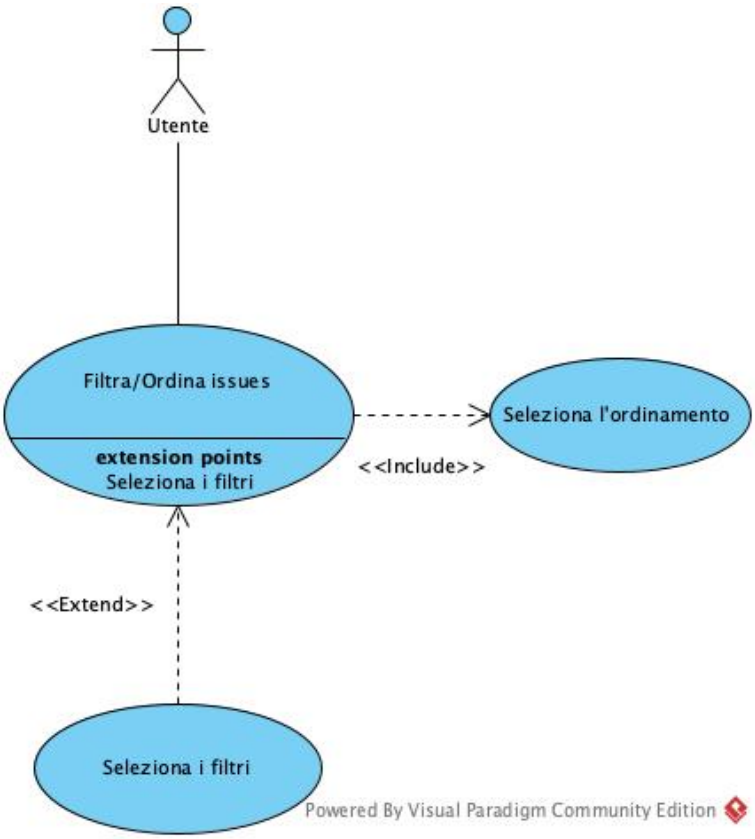

Figura 2.3: Modellazione requisito *RF4*

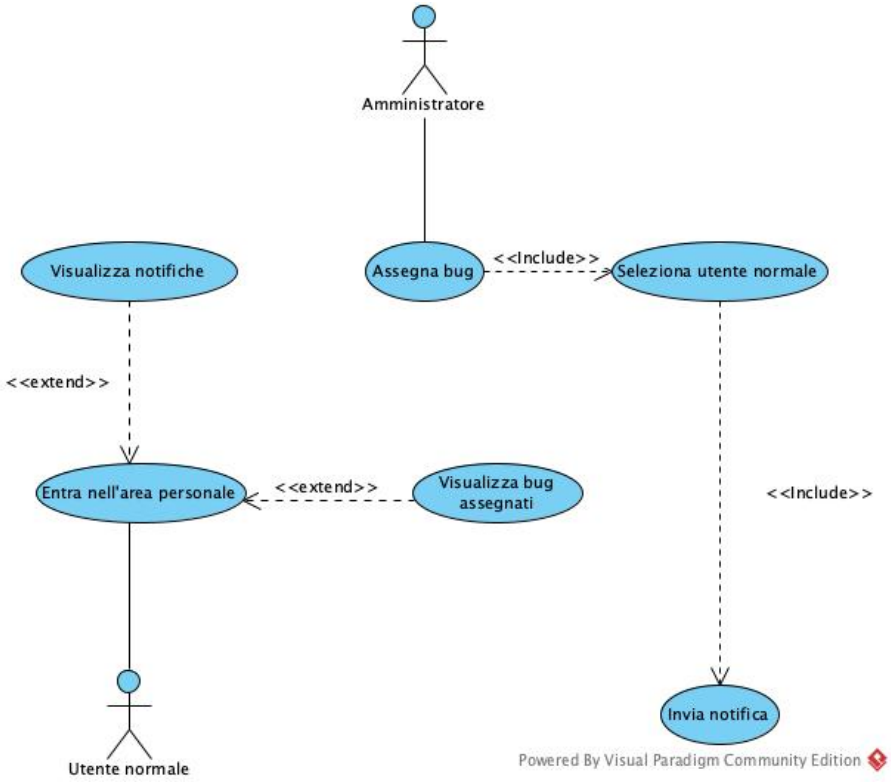

Figura 2.4: Modellazione requisiti *RF5*, *RF6*, *RF7*

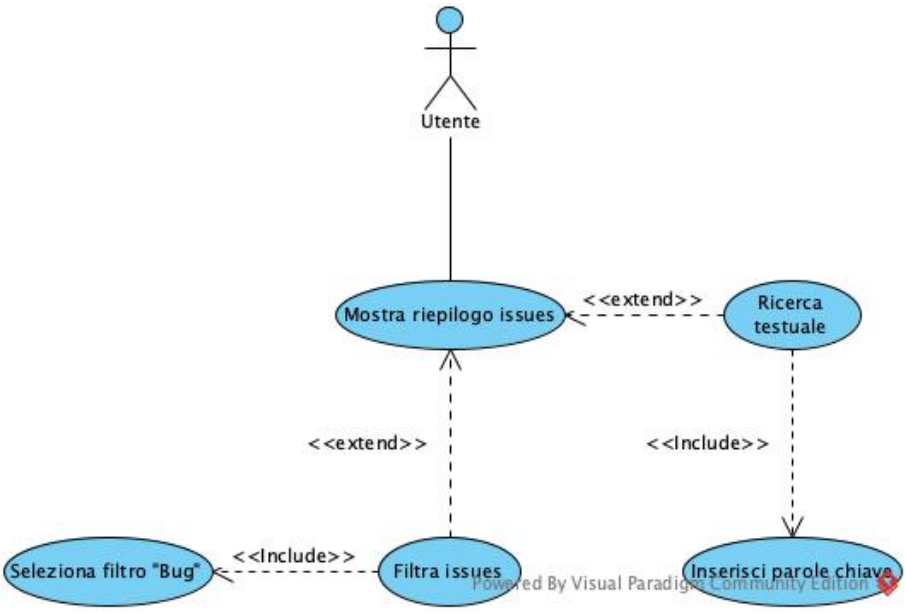

Figura 2.5: Modellazione requisito *RF8*

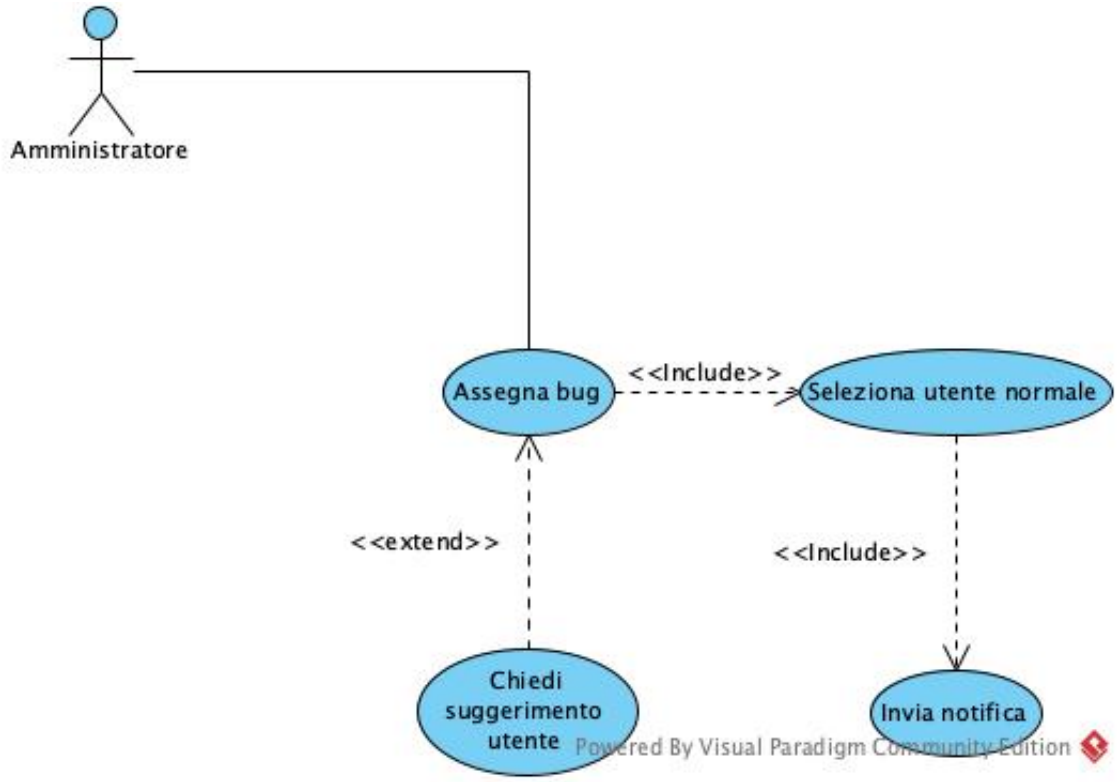

Figura 2.6: Modellazione requisito *RF9*

2.3 Caratteristiche utente

L'utilizzo di questo prodotto software da parte degli utenti richiede che questi ultimi abbiano una buona conoscenza del dominio applicativo in cui il software opera. Per tale motivo, gli utenti del sistema sono stati divisi in due categorie a seconda del livello di conoscenza del dominio:

1. *utenti interni*
2. *utenti esterni*

Gli *utenti interni* sono coloro che utilizzano la maggior parte delle funzionalità messe a disposizione dal sistema.
Tra questi vi sono, ad esempio, i membri di un team di sviluppo. Questi devono essere in grado di riconoscere e saper descrivere chiaramente i loro problemi (bug, suggerimento di nuove funzionalità, ecc.). Una scarsa conoscenza del dominio applicativo può portare all'incapacità di riconoscere e descrivere correttamente tali problemi, con conseguente perdita di tempo e potenziali fraintendimenti.
Gli *utenti esterni*, invece, sono coloro che fanno un uso limitato delle funzionalità del sistema, in quanto non possiedono una conoscenza approfondita del dominio applicativo.
Ad esempio, gli *stakeholder*, generalmente, hanno poca (o nessuna) esperienza tecnica nella risoluzione di un bug o nello sviluppo software. Tuttavia, dovrebbero almeno possedere una comprensione di base del contesto, per poter capire quanto un problema sollevato dagli sviluppatori possa impattare sul prodotto o sul mercato, anche se non sono loro a decidere la soluzione tecnica.

Nello specifico, le definizioni e i ruoli dei singoli attori del sistema sono:

Utente interno
Persona autenticata che può accedere alla maggior parte delle funzioni del sistema. Questi sono suddivisi in tre categorie: *Superutente*, *Utente amministratore* e *Utente normale*.

Utente esterno
Persona autenticata che ha accesso read-only al sistema. Ruolo creato per utenti *stakeholder* e affini.

Amministratore
Si riferisce a un utente che ha accesso completo al sistema, senza limitazione alcuna. Tra le operazioni che può effettuare figurano la gestione degli utenti e la gestione dei bug (come ad esempio, l'assegnazione di un bug a un membro del team).

Utente normale
Chiamato anche utente *regolare*, si riferisce a un utente senza privilegi speciali. Il suo ruolo è quello di segnalare problemi, proporre miglioramenti o fare richieste funzionali.

Tali account vengono assegnati normalmente ai membri di un team di sviluppo, i quali possono anche visualizzare i bug assegnati in un'apposita sezione.

2.4 Presupposti e dipendenze

2.4.1 Dipendenze tecnologiche

- *Containerizzazione del back-end*: Il committente desidera che una componente del software, ovvero la parte di *back-end*, sia distribuita utilizzando tecnologie di containerizzazione come *Docker*.
  - *Dipendenze*: Si assume che le macchine server supportino l'esecuzione di *container Docker*, includendo un demone *Docker* installato correttamente e configurato.

2.4.2 Dipendenze sull'ambiente dell'utente

- *Connettività di rete*: L'architettura di sistema prevede che il *front-end* comunichi con il *bakc-end* anche quando questo risiede in una rete esterna.
  - *Dipendenze*: Si assume che gli utenti finali dispongano di una connessione Internet stabile e affidabile per poter utilizzare l'applicazione senza interruzioni. La qualità dell'esperienza utente sarà direttamente dipendente dalla qualità della loro connessione

Capitolo 3

Requisiti specifici

Il presente capitolo cerca di fornire una descrizione più dettagliata dei requisiti funzionali utente presentati nel capitolo 2, nonché dei requisiti non funzionali imposti. Anche se, in teoria, i requisiti di sistema dovrebbero semplicemente descrivere il comportamento esterno e i vincoli operativi, non come il sistema dovrebbe essere progettato, in realtà, il livello di dettaglio adottato per specificare tali requisiti ha reso impossibile escludere tutte le informazioni sulla progettazione.

3.1 Requisiti funzionali

Il presente paragrafo si presta a descrivere in maniera più dettagliata i casi d'uso presentati nel precedente capitolo. Per la descrizione testuale di tali casi d'uso si è deciso di adottare un formato proposto da Alistar Cockburn e chiamato *template di Cockburn*. Il motivo di tale scelta è legato alla sua struttura tabellare, che facilita la comprensione delle interazioni tra attori e sistema, offrendo una rappresentazione esaustiva di tutti gli eventi.
Si fa presente al lettore che le tabelle seguenti contengono riferimenti a *mock-up*. Per una maggiore comprensione si raccomanda di consultare questi artefatti quando vengono citati. Questo paragrafo è suddiviso nei seguenti sottoparagrafi:

1. *Template di Cockburn*: Sebbene tale sottopagrafo preveda la presentazione di un *template di Cockburn* per ogni *caso d'uso* significativo, al momento è stato dettagliato unicamente il caso d'uso *Filtra/Ordina issues*.
2. *Prototipazione visuale via Mock-up*: I prototipi scelti sono stati presentati con una interfaccia simile a quella pianificata per il sistema da realizzare. In questo modo è stato facile identificare sia le inconsistenze del design grafico che quelle legate alla logica operativa del sistema.

3.1.1 Template di Cockburn

<table>
<tr><td>Use case #1</td><td colspan="3">Filtra/Ordina issues</td></tr>
<tr><td>Goal in Context</td><td colspan="3">Consente all'utente di ordinare e filtrare le issue in base a diversi criteri</td></tr>
<tr><td>Preconditions</td><td colspan="3">L'utente deve possedere un proprio account</td></tr>
<tr><td>Success End Condition</td><td colspan="3">Il sistema restituisce i risultati della ricerca avanzata</td></tr>
<tr><td>Failed End Condition</td><td colspan="3">il sistema non produce nessun risultato di ricerca</td></tr>
<tr><td>Primary Actor</td><td colspan="3">Utente</td></tr>
<tr><td>Trigger</td><td colspan="3">L'utente clicca sul pulsante "Filtra/Ordina"</td></tr>
<tr><td>Main Scenario</td><td colspan="3"><table>
<tr><th>Step n.</th><th>Utente</th><th>Sistema</th></tr>
<tr><td>1</td><td></td><td>Mostra M1</td></tr>
<tr><td>2</td><td>Clicca sul pulsante "Filtra/Ordina"</td><td></td></tr>
<tr><td>3</td><td></td><td>Mostra M2</td></tr>
<tr><td>4</td><td>Seleziona il tipo di filtro</td><td></td></tr>
<tr><td>5</td><td></td><td>Mostra M3</td></tr>
<tr><td>6</td><td>Clicca sul pulsante "Cerca"</td><td></td></tr>
<tr><td>7</td><td></td><td>Mostra M4</td></tr>
</table></td></tr>
</table>

3.1.2 Prototipazione visuale via Mock-up

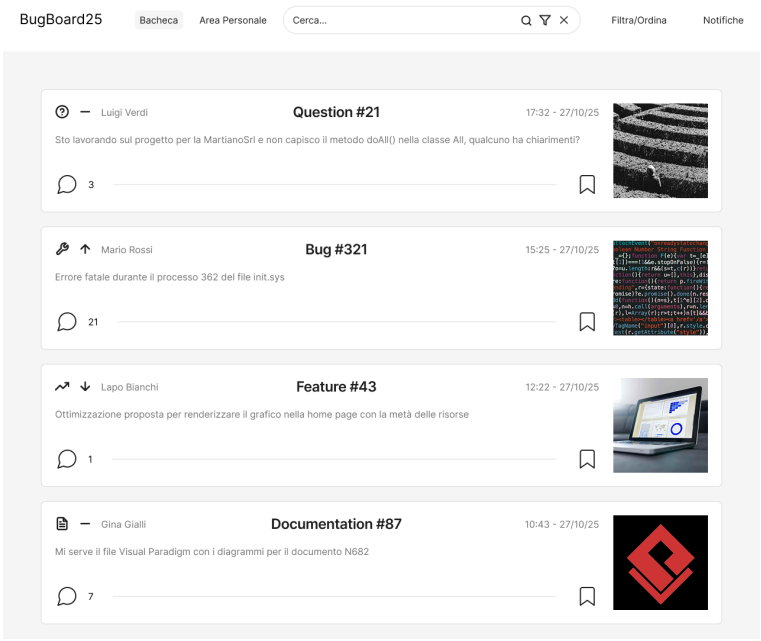

Figura 3.1: Mock-up *M1*

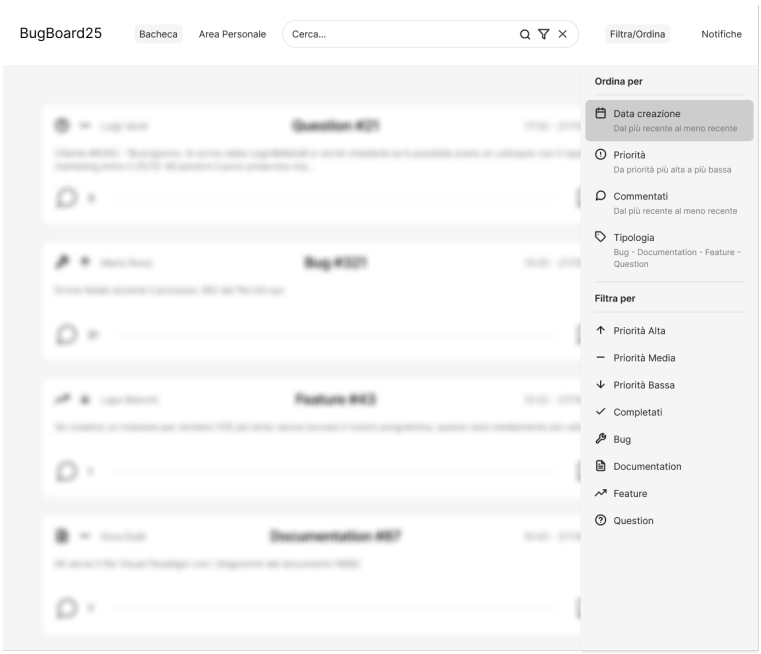

Figura 3.2: Mock-up *M2*

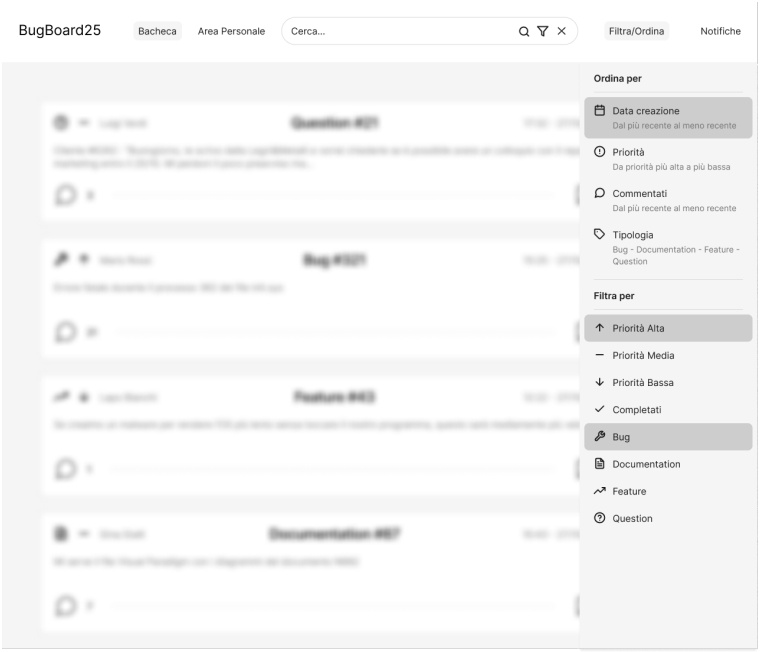

Figura 3.3: Mock-up *M3*

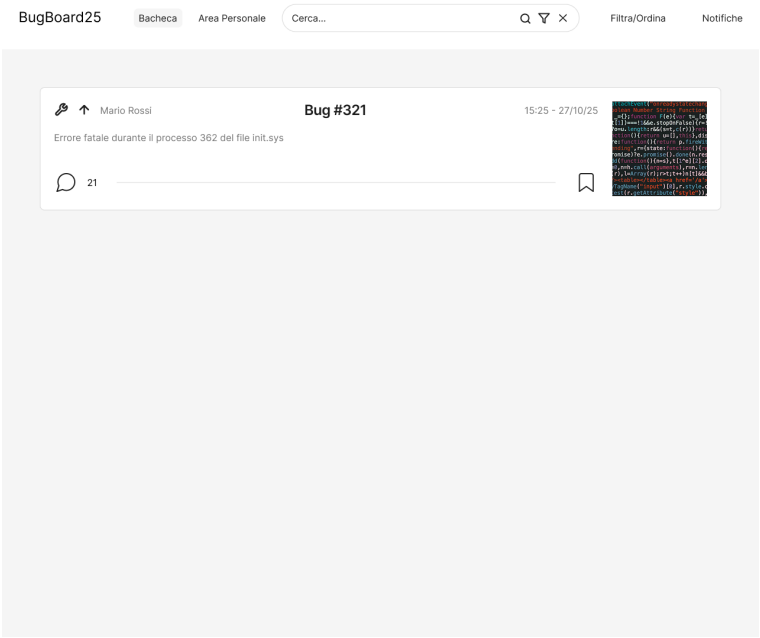

Figura 3.4: Mock-up *M4*

3.2 Requisiti non funzionali

3.2.1 Requisiti del prodotto

RP1 (*Requisito di affidabilità*): Il sistema deve offrire un alto grado di probabilità con cui fornisca correttamente i servizi come atteso dal cliente in

un determinato periodo di tempo.

RP2 (*Requisito di apprendimento*): Il sistema deve offrire un alto grado di *familiarizzazione* dell'utente con le funzioni del prodotto entro un determinato periodo di tempo.

RP3 (*Requisito di adeguatezza*): Il sistema deve facilitare il raggiungimento dei compiti e degli obiettivi da parte dell'utente.

RP4 (*Requisito di portabilità*): Una parte del sistema (*back-end*) può essere eseguita in ambienti diversi. Il termine ambiente può riferirsi alla piattaforma hardware o all'ambiente software.

RP5 (*Requisito di confidenzialità*): Il sistema deve impedire accessi non autorizzati a informazioni riservate.

3.2.2 Requisiti organizzativi

RO1 Il sistema deve basarsi su una architettura distribuita.
Requisiti di sistema:

- Il sistema deve prevederre due macro-componenti indipendenti: *back-end* e *front-end*.
- Il *back-end* deve essere composto da almeno una componente che gestisce la logica di business e i dati.
- Il *back-end* deve offire una interfaccia di programmazione, con cui comunicare, mediante *API REST*.
- Il *front-end* deve comunicare con il *back-end* solo tramite le *API* esposte.
- Il *front-end* deve essere realizzato come applicazione desktop, come applicazione web, oppure come applicazione mobile.

RO2 Per la fase di implementazione devono essere utilizzati linguaggi di programmazione Object-Oriented.
Requisiti di sistema:

- Sia il *back-end* che il *front-end* devono essere realizzati utilizzando linguaggi del paradigma ad oggetti.
- Il *front-end* dovrebbe essere un'applicazione desktop realizzata con Java e le librerie grafiche Swing o JavaFX.

RO3 In fase di produzione devono essere utilizzati tool di CASE.
Requisiti di sistema:

- Strumenti come Visual Paradigm devono essere utilizzati in fase di progettazione.

RO4 Il prodotto software e la relativa documentazione devono essere consegnati entro il 30 Settembre 2026.
Requisiti di sistema:

- Per il dettaglio completo delle scadenze di progetto si rimanda al capitolo **8** del documento **Progetto di Ingegneria del software.pdf**.

RO5 Il sistema deve essere progettato rispettando i requisiti antiplagio, in grado di soddisfare e superare gli standard richiesti in materia di originalità dei contenuti.
Requisiti di sistema:

- Tutti gli artefatti consegnati vengono processati con software antiplagio Turnitin, e raffrontati automaticamente con progetti degli anni scorsi e di quest'anno.