Progetto INGSW

Francesco Piscopo, Raffaele Ruggiero

19 ottobre 2025

1 Indice

Indice

2 Glossario

- Admin - È un utente con privilegi di amministrazione il quale può gestire utenti, configurare il sistema e accedere a tutte le eventuali funzionalità.
- Back-end - Parte nascosta al sistema presente sul server che esegue l'elaborazione dei dati ed altre funzionalità.
- Board - Piattaforma principale di BugBoard26 dove vengono visualizzate e gestite tutte le issue del progetto.
- Bug - Tipo specifico di issue che segnala un malfunzionamento o un comportamento non corretto del software.
- Browser - Applicazione web (Chrome, Firefox, Safari, etc.) attraverso cui gli utenti accedono all'interfaccia di BugBoard26.
- Client-server - Una tipologia di architettura che vede gli utenti come connessi da dei clienti che eseguono richieste e il sito al quale accedono + visto come fornitore di servizi.
- DBMS - Database Management System, un sistema che permetterà alla piattaforma di gestire i dati in maniera sicura ed efficiente.
- Download - Funzionalità di esportazione e scaricamento dei dati.
- Front-end - Parte visibile ed interattiva della pagina web.
- Issue - Elemento fondamentale del sistema che rappresenta un problema, richiesta o segnalazione relativa al progetto software.
- Login - Processo di autenticazione tramite email e password che consente l'accesso alle funzionalità del sistema.
- Logout - Processo di disconnessione dell'utente.
- Uptime - Metrica che misura la disponibilità del sistema di tempo in cui BugBoard26 è operativo.

3 Richieste del cliente

BugBoard26 è una piattaforma per la **gestione collaborativa di issue** in progetti software. Il sistema consente a team di sviluppo di **segnalare problemi** relativi a un progetto, **monitorarne lo stato**, **assegnarli** a membri del team e **tenere traccia delle attività** di risoluzione. Il sistema deve consistere in un'applicazione (mobile, desktop o web-based) **performante e affidabile**, attraverso cui gli utenti possono fruire delle funzionalità in modo **intuitivo e rapido**.

- Deve essere implementato un **sistema 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".

- 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".
- 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.
- Deve essere possibile **esportare l'elenco dei bug** in un formato condivisibile, come CSV, Excel o PDF.
- Il sistema deve permettere di **associare un numero variabile di etichette** personalizzabili ai bug (es. "frontend", "urgente", "sicurezza").
- Il sistema deve offrire **una modalità "readonly" per utenti esterni** (es. stakeholder), che consenta di visualizzare i bug e i commenti senza possibilità di modifica. Gli account per questi utenti vengono creati dagli amministratori.

3.1 Problemi con le richieste del cliente

Le richieste del cliente presentano una evidente contraddizione; per definizione un utente che esegue il login è **autenticato**, questo implica che l'utente **deve poter segnalare una issue a suo piacimento**. La contraddizione si presenta nel caso di un utente **esterno**, il quale non deve poter interagire con la board aggiungendo informazioni a quest'ultimo.

Dato questo problema abbiamo deciso di trattare gli utenti esterni e gli altri autenticati in maniera differente per poter mantenere una coerenza che riesca a soddisfare i requisiti richiesti dal clienti al meglio.

4 Introduzione

4.1 Panoramica

Questa documentazione descrive l'**SRS** (Specifica dei Requisiti Software) di una piattaforma relativa alla gestione di issue da parte di un team di sviluppo. Questa documentazione è utile a stakeholder, designer, sviluppatori e tester.

4.2 Scopo

BugBoard26 è una piattaforma che permette di tenere traccia di issue di varie tipologie relativi ad un progetto di un team di sviluppo. Questa piattaforma potrà risultare utile ai membri del team di sviluppo e ai vari stakeholder interessati al relativo progetto.

4.3 Ambito

- **BugBoard26 web-app** - Permetterà di interfacciarsi al sistema tramite browser.
- **Database postgres** - Gestirà i dati in maniera sicura e efficiente.

4.4 Panoramica del prodotto

4.4.1 Funzionalità del prodotto

Sono di seguito riportate le principali funzionalità di sistema tramite **use case diagrams** seguiti poi da una breve descrizione testuale. Nei diagrammi vengono utilizzati degli attori di sistema, senza però che vengano esplicitate le generalizzazioni che li definiscono, queste sono consultabili nell'apposito capitolo della documentazione.

Login Il sistema deve permettere ad un utente di autenticarsi in quest'ultimo tramite l'utilizzo di email e password.

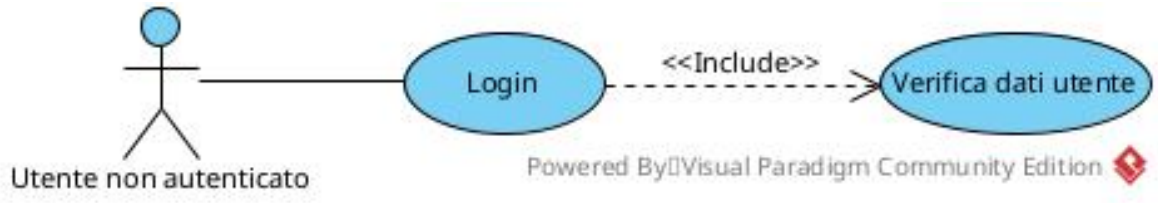

Figura 1: Diagramma UML raffigurante il caso d'uso di login.

Logout Il sistema deve permettere ad un utente autenticato di terminare la sua sessione.

Figura 2: Diagramma UML raffigurante il caso d'uso di logout.

Creazione utente Il sistema deve permettere agli utenti amministratori di poter creare nuovi utenti.

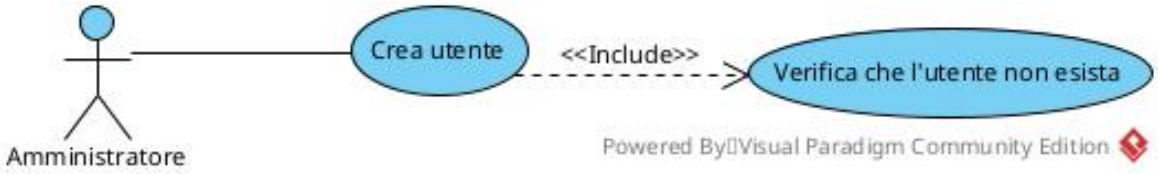

Figura 3: Diagramma UML raffigurante il caso d'uso di creazione di un utente.

Crea issue Il sistema deve permettere agli utenti autenticati (esclusi gli utenti esterni) di creare un issue, con varie funzionalità aggiuntive.

Figura 4: Diagramma UML raffigurante il caso d'uso di creazione di un issue da utente normale.

Figura 5: Diagramma UML raffigurante il caso d'uso di creazione di un issue da admin.

Modifica stato issue Il sistema deve permettere agli utenti autenticati (esclusi gli utenti esterni) di modificare lo stato di un issue.

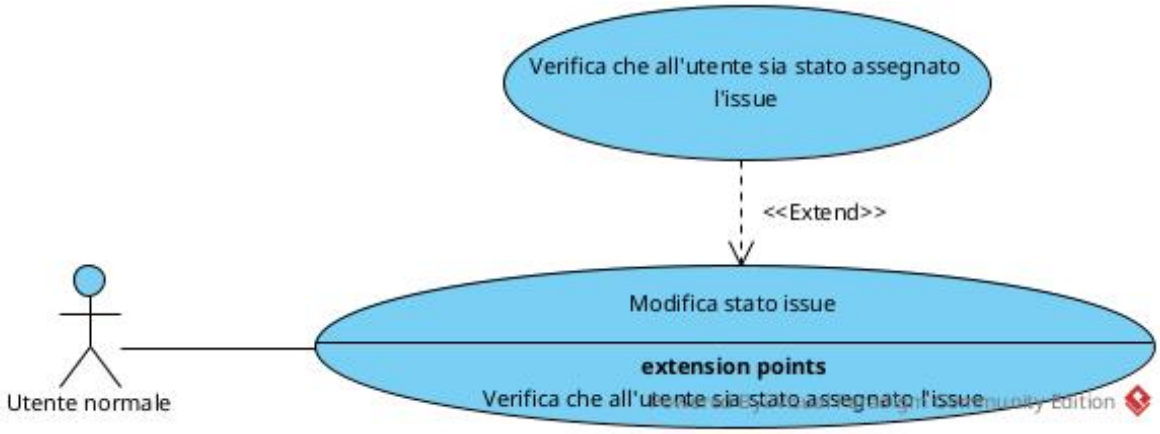

Figura 6: Diagramma UML raffigurante il caso d'uso di modifica di un issue da utente normale.

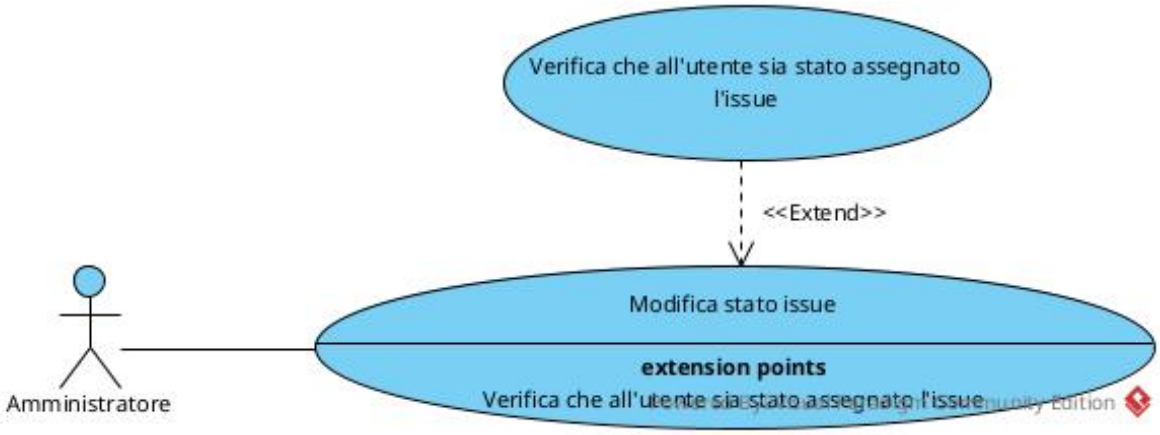

Figura 7: Diagramma UML raffigurante il caso d'uso di creazione di modifica di un issue da admin.

Assegna issue Il sistema deve permettere agli utenti autenticati (esclusi gli utenti esterni) di assegnarsi l'un l'altro degli issue.

Figura 8: Diagramma UML raffigurante il caso d'uso di assegnazione issue da utente normale a utente normale.

Figura 9: Diagramma UML raffigurante il caso d'uso di assegnazione issue tra utente normale e admin.

Figura 10: Diagramma UML raffigurante il caso d'uso di assegnazione issue da utente admin a utente admin.

Commenta issue Il sistema deve permettere agli utenti autenticati (esclusi quelli esterni) di commentare sotto uno specifico issue.

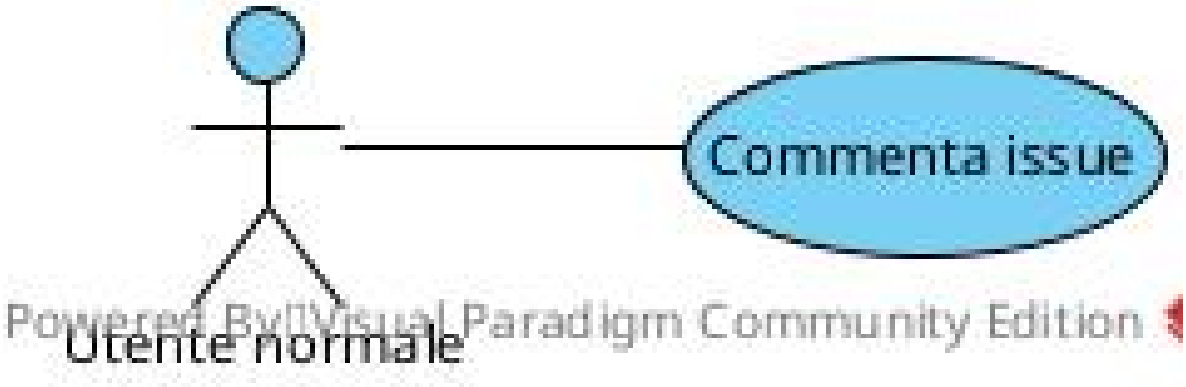

Figura 11: Diagramma UML raffigurante il caso d'uso di commento a un issue da parte di un utente normale.

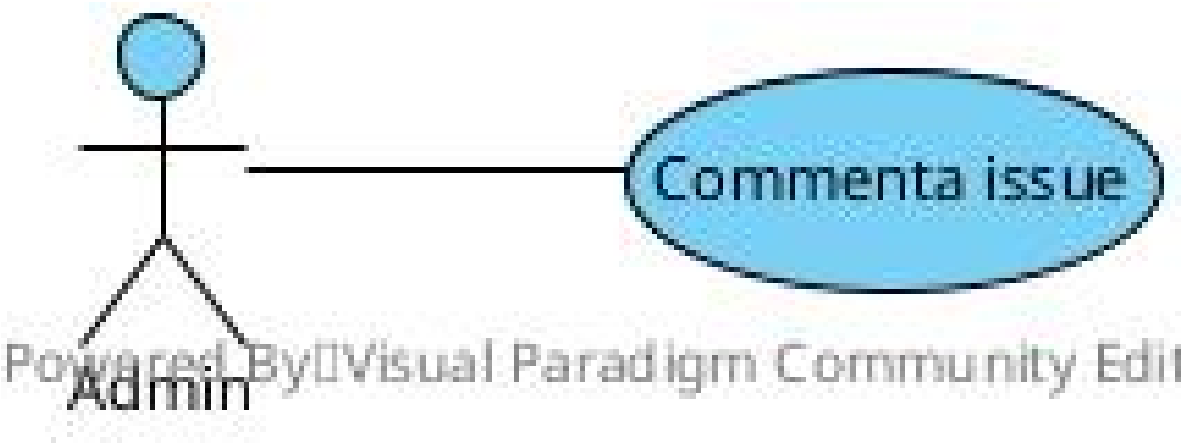

Figura 12: Diagramma UML raffigurante il caso d'uso di commento a un issue da parte di un admin.

Visualizza / cerca issue / esporta elenco bug Il sistema deve permettere agli utenti autenticati di poter visualizzare la lista completa di issue, di cercare un'issue specifico o di filtrare gli issue secondo certi criteri.

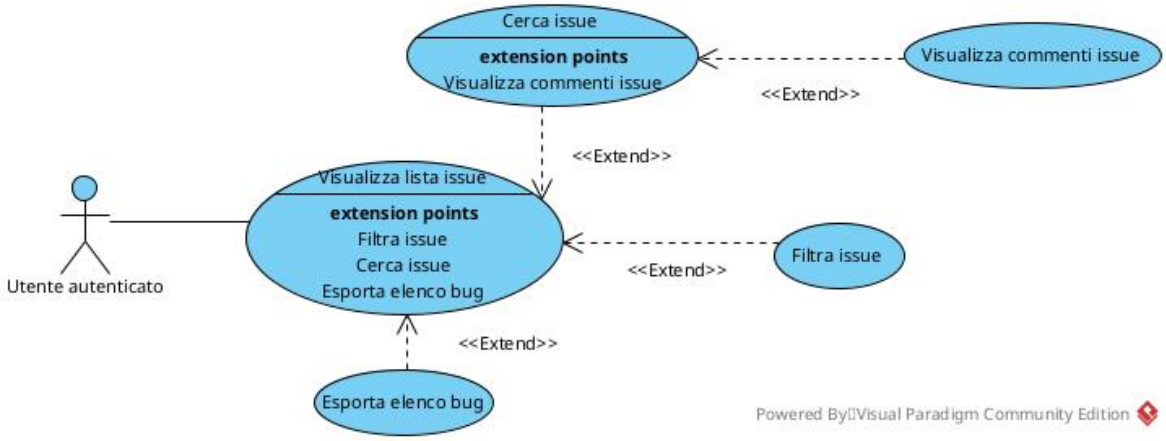

Figura 13: Diagramma UML raffigurante il caso d'uso di visualizzazione/ricerca degli issue.
NOTA: Gli utenti admin e normali presentano un'ulteriore estensione del caso d'uso alla ricerca del singolo issue che gli permette di commentarli.

Formalizzazione tramite Cockburn e MockUp Di seguito sono presentati:

- Un mock-up raffigurante alcune delle funzionalità della piattaforma, sia in modalità utente normale che utente admin.
- Un diagramma di Cockburn che, anche aiutandosi con il mock-up, spiega una funzionalità del sistema in maniera approfondita.

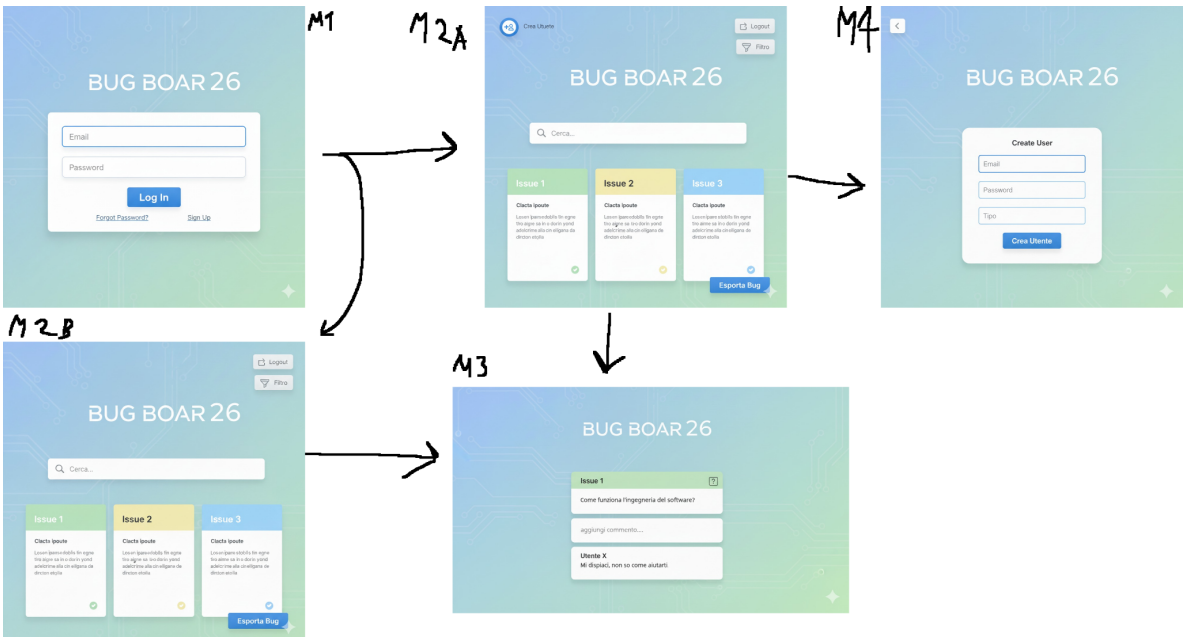

Figura 14: Mockup realizzato con AI.

	Esporta elenco bug		
Obiettivo	Esportare l'elenco dei bug (ovvero issue che non sono domande) in uno specifico formato		
Pre-condizioni	Lo stato si deve trovare in M1		
Condizione di successo	L'utente che vuole esportare scarica dalla piattaforma un file contenente tutti i bug		
Condizione di fallimento	L'utente riceve un messaggio di errore sullo schermo		
Attore principale	Utente		
Trigger	Click di un tasto per esportare tutti i bug		
Scenario principale	**Step n.**	**Utente**	**Sistema**
	1	Esegue il login con le sue credenziali	
	2		Autentica l'utente (mostra M2A o M2B)
	3	Preme sul tasto "Esporta bug"	
	4		Chiede all'utente tipo e nome file di output
	5	Inserisce tipo e nome file di output e clicca "Download"	
	6		Inizia il download
Estensione 1	5a	Non inserisce una delle due / tutte e due le informazioni e clicca "Download"	
	6a		Mostra messaggio di errore (M6)

Figura 15: Diagramma di cockburn per l'use case di esportazione dei bug.

Figura 16: M6.

4.4.2 Caratteristiche degli utenti

Gli utenti interessati presentano delle evidenti generalizzazioni elencate di seguito:

- Utente
  - Autenticato
    - Amministratore
    - Esterno
    - Normale
  - Non autenticato

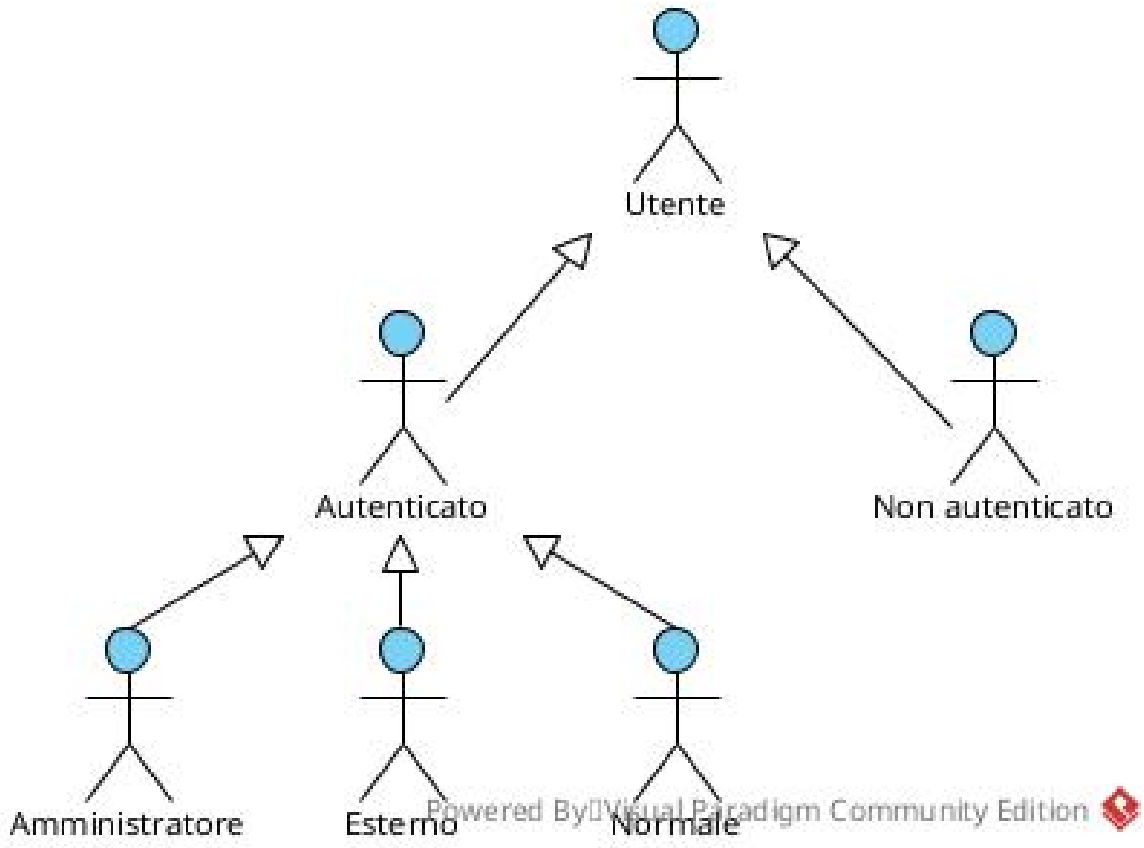

Figura 17: Diagramma UML raffigurante gli attori del sistema.

4.4.3 Vincoli

Design Il sistema deve essere **intuitivo**.

Sicurezza Il sistema deve preservare la **segretezza** dei dati degli utenti durante la fase di login.

Performance Il sistema deve essere **rapido**.

5 Requisiti

5.1 Funzionalità

5.1.1 Completezza

Il sistema deve garantire di poter eseguire tutte le funzionalità richieste dal cliente.

5.1.2 Correttezza

Il sistema deve eseguire correttamente le funzionalità richieste dal cliente.

5.1.3 Adeguatezza

Il sistema dovrebbe facilitare la gestione di issue relativi ad un progetto da parte di un team di sviluppo grazie alla sua **facilità d'uso e la sua efficienza**.

5.2 Requisiti di performance

5.2.1 Tempi di risposta

- Login - max 5 secondi
- Caricamento schermata principale - max 10 secondi
- Ricerca - max 10 secondi
- Esportazione bug - max 60 secondi
- Creazione utente - max 3 secondi

5.2.2 Capacità massima

Il sistema deve garantire di poter funzionare con 10 utenti collegati in maniera concorrente.

5.3 Requisiti di usabilità

5.3.1 Riconoscibilità

Il sistema deve presentare un design intuitivo.

5.3.2 Imparabilità

Il sistema dovrebbe poter essere imparato da un utente entro 60 minuti.

5.3.3 Protezione dagli errori umani

Il sistema dovrà ridurre al minimo le operazioni eseguibili da parte degli utenti e in caso di errore da parte di questi dovrà presentare un messaggio di errore senza causare guasti nel sistema in se.

5.3.4 Inclusività

Il sistema dovrebbe poter essere usato da ogni tipo di utente in quanto non sono presenti operazioni con vincoli di tempo/velocità/accuratezza.

5.4 Requisiti logici di database

TODO

5.5 Vincoli di design

5.5.1 Web-app

Il sistema dovrà dividere la logica **front-end** da quella **back-end** e utilizzare una architettura **client-server**.

5.5.2 DBMS

Il sistema dovrà interfacciarsi ad un **DBMS** per la gestione dei dati in maniera sicura e efficiente.

5.6 Attributi software di sistema

5.6.1 Affidabilità

Disponibilità Il sistema deve garantire un uptime del 99% durante l'utilizzo.

Infallibilità Il sistema deve garantire che ogni operazione possa essere eseguita senza causare problemi permanenti.

Tolleranza ai guasti Il sistema deve garantire che in caso di guasto questo possa tornare operativo entro 2 ore.

Recuperabilità Il sistema deve garantire che tutti i dati che vi sono stati caricati entro 10 minuti da un guasto siano ripristinabili al ristoro del funzionamento del sistema.

5.6.2 Sicurezza(Security)

Confidenzialità Il sistema deve garantire la privatezza durante l'accesso di un utente.

Integrità Il sistema deve garantire che non ci siano perdite di dati e che questi siano corretti e completi.

Non-ripudio Il sistema deve tenere traccia delle attività eseguite dagli utenti.

Autenticità Il sistema deve garantire che l'utente possa dimostrare la propria identità e possa di conseguenza adempire alle azioni egli consentite.

5.6.3 Compatibilità

Co-esistenza Il sistema deve poter permettere all'utente di utilizzare anche altri programmi sul proprio computer.

5.6.4 Manutenabilità

Modularità Il sistema deve essere fortemente modulare dato l'uso di linguaggi che si rifanno alla programmzione Object Oriented

Modificabilità Il sistema deve essere facilmente modificabile dato l'alto livello di astrazione tra i sottosistemi.

Testabilità Il sistema dovrebbe poter essere testabile facilmente su ogni sistema e in ogni sua funzionalità.

5.6.5 Flessibilità

Adattabilità Il sistema deve poter essere utilizzato su quasi ogni tipo di sistema che abbia un browser installato.

Scalabilità Il sistema dovrà favorire l'estensione di capienza di dati massimi e capacità di utenza massima.

Installabilità Il sistema deve poter essere installabile facilmente su ogni sistema munito di browser.

6 Validazione

TO DO.

6.1 Funzionalità

6.2 Requisiti di performance

6.3 Requisiti di usabilità

6.4 Requisiti di interfaccia

6.5 Requisiti logici di database

6.6 Vincoli di design

6.7 Attributi software di sistema