BugBoard26

Documento di Specifica dei Requisiti Software

Progetto di Ingegneria del Software
A.A. 2025/2026

Fabrizio Padulano Matricola: N86005058
Federico Padulano Matricola: N85005043

Ottobre 2025

Indice

1 Introduzione

1.1 Scopo

Questo documento specifica i requisiti funzionali e non-funzionali del sistema BugBoard26, una piattaforma web per la gestione collaborativa di issue in progetti software.

1.2 Scope

1.2.1 Incluso

- Autenticazione e gestione utenti (2 ruoli: admin, normale)
- Creazione, modifica e visualizzazione di issue
- Assegnazione di bug ai membri del team
- Notifiche automatiche
- Filtri e ricerca issue
- API REST back-end
- Interfaccia web responsive

2 Glossario

Il glossario definisce i termini chiave del dominio applicativo utilizzati nel sistema BugBoard26.

Bug Errore nel software da sistemare.

Stato Posizione della issue: da fare, in lavorazione o risolta.

Priorità Quanto è urgente la issue.

Amministratore Chi gestisce il sistema, assegna i bug e crea account.

Utente normale Chi segnala issue e lavora su quelle assegnate.

Assegnazione Dare un bug a una persona specifica del team.

Etichetta Tag per categorizzare le issue (es. "frontend", "urgente").

Archiviazione Spostare una issue fuori dalle liste principali.

Bug duplicato Problema già segnalato. Viene chiuso.

Carico di lavoro Quanti bug ha una persona da sistemare.

Cronologia Storico di tutti i cambiamenti fatti su una issue.

Dashboard Pagina per amministratori con numeri e statistiche.

Tempo medio di risoluzione Quanto tempo ci vuole in media per sistemare un bug.

Report mensile Riassunto mensile dell'attività del team.

Scadenza Data entro cui si dovrebbe risolvere la issue.

Notifica Messaggio automatico per avvisare di cose importanti.

3 Modellazione dei Casi d'Uso

3.1 Autenticazione Utente (Requisito 1)

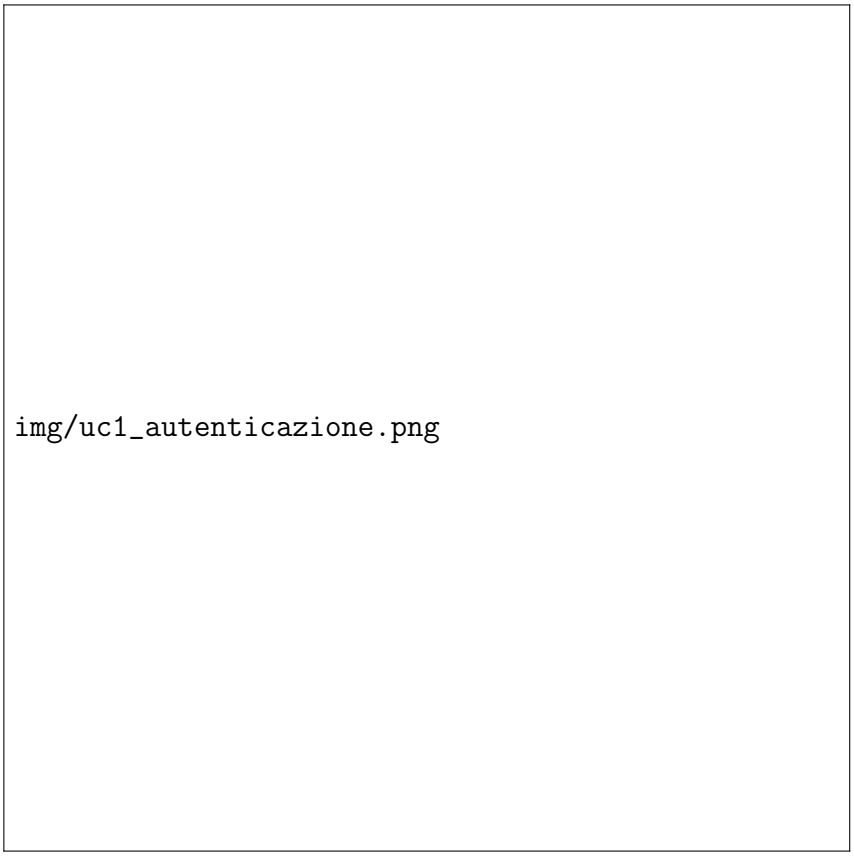

Figura 1: Use Case Diagram - Autenticazione Utente

3.2 Segnalare Issue (Requisito 2)

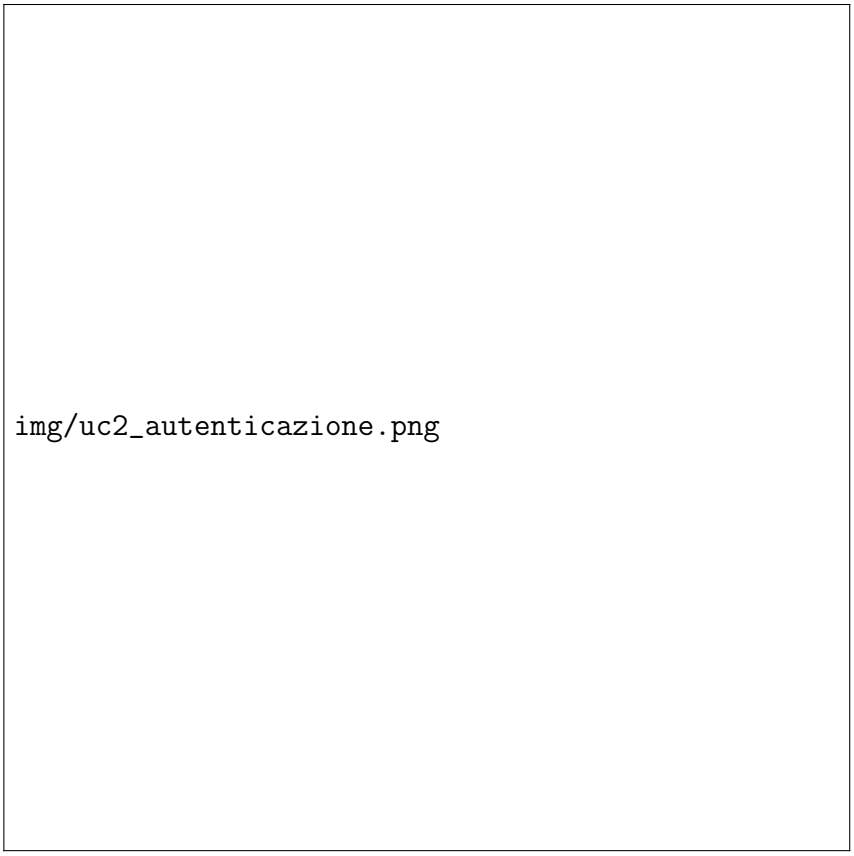

Figura 2: Use Case Diagram - UC2 Segnalare Issue

3.3 Visualizzare Vista Riepilogativa Issue (Requisito 3)

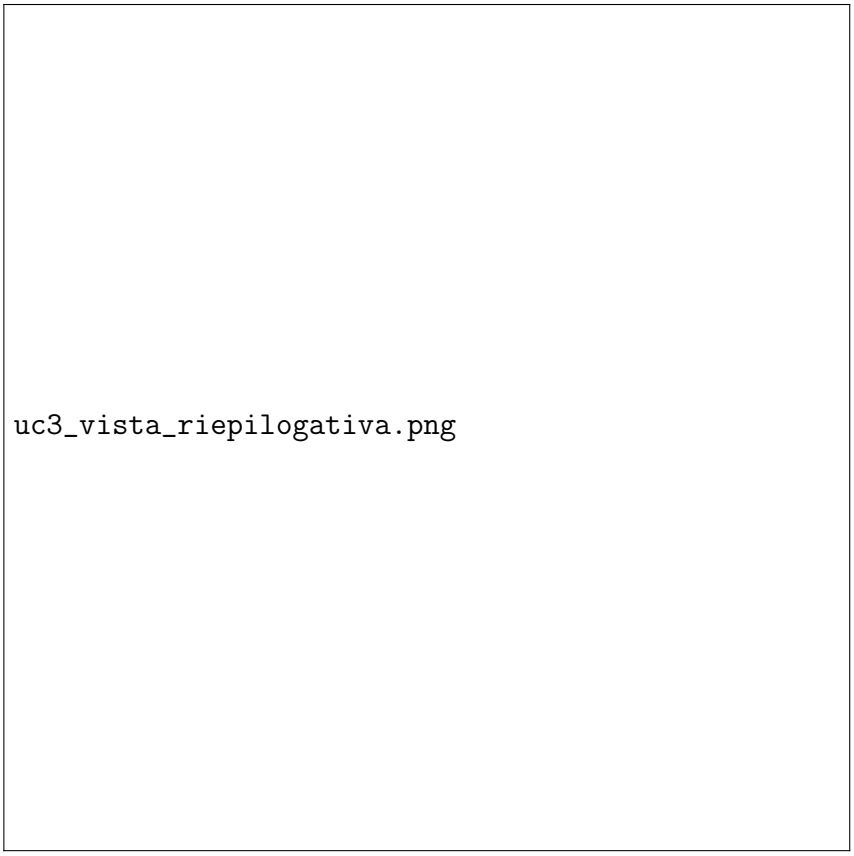

Figura 3: Use Case Diagram - UC3 Visualizzare Vista Riepilogativa Issue

3.4 Assegnare Bug (Requisito 4)

Figura 4: Use Case Diagram - UC4 Assegnare Bug

3.5 Modificare Bug Assegnato (Requisito 9)

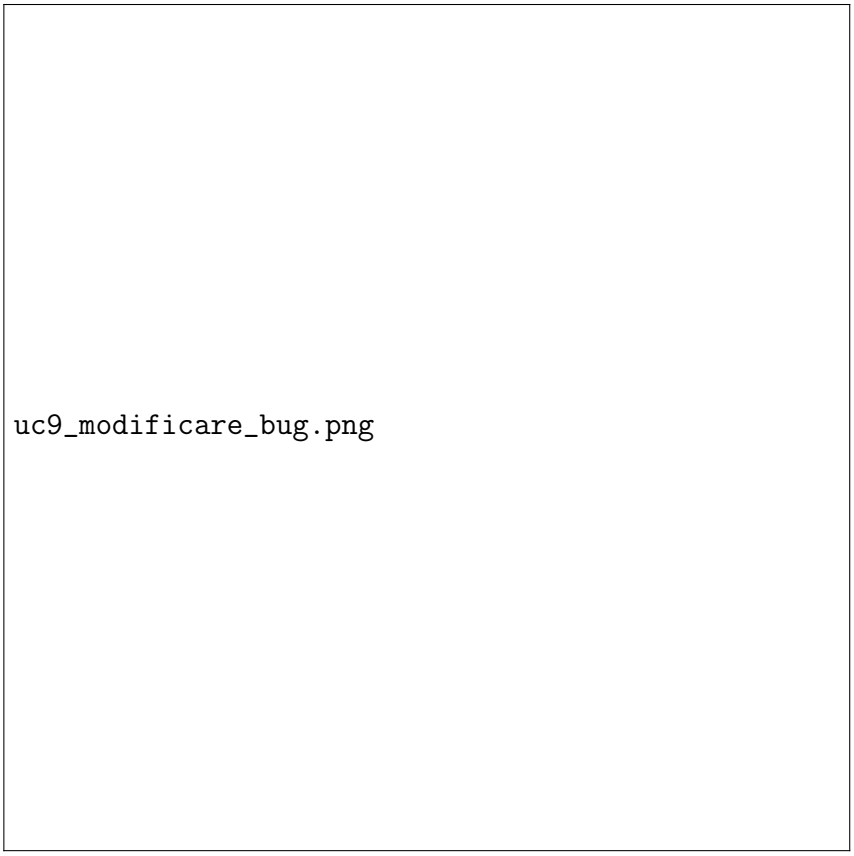

Figura 5: Use Case Diagram - UC9 Modificare Bug Assegnato

3.6 Impostare Scadenza Issue (Requisito 18)

Figura 6: Use Case Diagram - UC18 Impostare Scadenza Issue

4 Individuazione e Caratterizzazione degli Utenti (Attori) del Sistema

1. Utente Autenticato

Descrizione

Rappresenta qualsiasi membro del team di sviluppo che ha completato con successo il processo di autenticazione nel sistema BugBoard26. Costituisce la categoria base di utilizzatori del sistema con permessi standard.

Caratteristiche

- Possiede credenziali valide (email e password) registrate nel sistema
- Ha superato il processo di autenticazione
- Appartiene al team di lavoro del progetto software

Responsabilità e Capacità

- **Segnalazione bug:** Può creare nuove segnalazioni di problemi, specificando titolo, descrizione, tipologia (enhancement, bug, documentation, feature) e priorità

- **Visualizzazione:** Può accedere alla vista riepilogativa di tutti i bug del progetto
- **Filtro e ricerca:** Può applicare filtri per tipologia, stato, priorità e altri parametri rilevanti
- **Ordinamento:** Può ordinare i risultati secondo criteri specifici
- **Modifica limitata:** Può modificare esclusivamente i bug a lui assegnati (titolo, descrizione, tipologia, priorità, stato)
- **Visualizzazione assegnazioni:** Può visualizzare l'elenco dei bug assegnati a sé stesso
- **Comunicazione:** Può lasciare commenti sui bug, aggiungere chiarimenti o richiedere informazioni
- **Gestione scadenze:** Può impostare scadenze opzionali per la risoluzione di bug
- **Notifiche:** Riceve notifiche automatiche quando viene assegnato un bug

Vincoli

- Non può modificare bug assegnati ad altri membri del team
- Non può gestire gli utenti del sistema
- Non può assegnare bug ad altri utenti
- Richiede autenticazione per accedere a qualsiasi funzionalità

2. Amministratore

Descrizione

Rappresenta un utente con privilegi elevati che ha responsabilità di gestione e supervisione dell'intero sistema BugBoard26. È una specializzazione dell'Utente Autenticato, ereditandone tutte le capacità e aggiungendo funzionalità amministrative.

Caratteristiche

- Possiede tutte le caratteristiche di un Utente Autenticato
- Ha un ruolo specifico di "Amministrazione" assegnato nel sistema
- Al momento dell'installazione del sistema esiste già un account amministratore con credenziali predefinite
- Rappresenta figure di supervisione come project manager, team leader o responsabili tecnici

Responsabilità e Capacità

Capacità Ereditate (da Utente Autenticato)

- Tutte le funzionalità disponibili agli utenti standard

Capacità Amministrative Esclusive

- **Gestione utenti:**
  - Creare nuovi account utente
  - Specificare email, password e ruolo per ogni utente
  - Modificare utenti esistenti
  - Assegnare ruoli (Utente standard o Amministratore)
- **Modifica completa:**
  - Modificare qualsiasi bug del sistema indipendentemente dall'assegnatario
  - Accesso completo a tutte le segnalazioni
- **Assegnazione bug:**
  - Assegnare bug a qualsiasi membro del team
  - Riassegnare bug già assegnati
  - Il sistema notifica automaticamente l'utente quando riceve un'assegnazione
- **Supervisione:**
  - Monitorare l'attività del team
  - Gestire il flusso di lavoro del progetto

Vincoli

- Richiede autenticazione come per gli utenti standard
- Le modifiche alle utenze sono tracciate dal sistema

Relazione con altri Attori

- **Generalizzazione:** L'Amministratore è una specializzazione dell'Utente Autenticato
- **Eredità:** Eredita tutte le capacità dell'Utente Autenticato più le funzionalità amministrative

3. Sistema (Attore Secondario)

Descrizione

Rappresenta il sistema BugBoard26 stesso quando agisce autonomamente per eseguire operazioni automatiche, in particolare per la gestione delle notifiche.

Caratteristiche

- Attore non umano
- Opera in modo automatico senza intervento diretto degli utenti
- Reagisce a eventi specifici del sistema

Responsabilità e Capacità

- **Invio notifiche automatiche:** Quando un amministratore assegna un bug a un utente, il sistema invia automaticamente una notifica all'utente assegnatario
- **Gestione eventi:** Rileva gli eventi di assegnazione e attiva le corrispondenti notifiche

Interazioni

- Collabora con l'Amministratore durante il processo di assegnazione bug
- Interagisce con gli Utenti Autenticati per recapitare le notifiche

5 Requisiti Non-Funzionali e di Sistema

I requisiti non-funzionali definiscono le proprietà qualitative del sistema BugBoard26. A differenza dei requisiti funzionali che descrivono cosa il sistema deve fare, questi definiscono come il sistema deve comportarsi in termini di performance, sicurezza, affidabilità e usabilità.

5.1 Sicurezza

5.1.1 1. Autenticazione e Crittografia

Le credenziali degli utenti devono essere trasmesse e memorizzate in modo sicuro. Le password devono essere crittografate nel database.

5.1.2 2. Controllo Accessi

Il sistema deve verificare i permessi prima di ogni operazione critica. Gli utenti normali non devono poter modificare bug altrui o accedere a funzionalità amministrative. Questo controllo deve avvenire sia nel client che nel server.

5.2 Performance

5.2.1 3. Tempi di Risposta

L'applicazione deve essere responsiva. Le operazioni comuni (caricare la lista dei bug, filtrare, cercare) devono completarsi in massimo 2-3 secondi. Le operazioni di salvataggio (creare una nuova issue, modificare un bug) devono essere completate entro 3 secondi. Questo garantisce un'esperienza utente fluida senza frustrazioni dovute a caricamenti lenti.

5.2.2 4. Caricamento e Filtraggio Veloce

La visualizzazione della lista dei bug e l'applicazione di filtri devono essere rapidi e fluidi. Anche con centinaia/migliaia di issue nel sistema, il caricamento della pagina non deve superare 2 secondi e i filtri devono rispondere immediatamente. L'applicazione deve implementare paginazione per gestire efficientemente grandi quantità di dati.

5.3 Affidabilità e Disponibilità

5.3.1 5. Gestione Errori

Quando accadono errori (server non raggiungibile, ecc.), l'applicazione deve gestirli elegantemente mostrando messaggi chiari all'utente. Deve essere sempre possibile tornare a uno stato stabile.

5.4 Usabilità

5.4.1 6. Interfaccia Intuitiva

L'interfaccia deve essere facile da usare. Le operazioni più comuni devono essere raggiungibili rapidamente. I messaggi di errore devono spiegare chiaramente cosa è andato storto

e come risolverlo. La terminologia usata deve essere coerente e corrispondere al glossario del progetto.

5.4.2 7. Design Responsive

Il design deve adattarsi automaticamente a diverse dimensioni di schermo (desktop, tablet, mobile).

5.5 Architettura

5.5.1 8. Separazione Front-end e Back-end

Il front-end e il back-end devono essere completamente separati. La comunicazione deve avvenire esclusivamente tramite API (HTTP REST).

5.5.2 9. Containerizzazione e Deployment

Il back-end deve essere distribuito in un container Docker. Questo garantisce che l'applicazione funzioni allo stesso modo in sviluppo, test e produzione. Il Dockerfile deve essere incluso nel repository insieme al codice sorgente. L'applicazione deve essere deployabile su AWS (oppure Google Cloud o Azure)

6 Formalizzazione Caso d'Uso Significativo

Questa sezione conterrà la formalizzazione dettagliata di un caso d'uso significativo con template di A. Cockburn e relativi mock-up.

USE CASE #9 - Modificare Bug

Informazioni Generali

USE CASE #9	Modificare Bug
Goal in Context	Un utente autenticato (Utente o Amministratore) vuole modificare le informazioni di un bug esistente nel sistema per aggiornarne lo stato, correggere informazioni o aggiungere dettagli.
Preconditions	• L'utente ha effettuato l'autenticazione nel sistema • Esiste almeno un bug nel sistema • Per l'Utente: il bug deve essere assegnato a lui • Per l'Amministratore: può modificare qualsiasi bug
Success End Condition	Il bug è stato modificato con successo e le modifiche sono salvate nel sistema. La cronologia delle modifiche viene aggiornata con timestamp e autore della modifica.
Failed End Condition	Il bug non viene modificato e rimane nello stato precedente. Il sistema può mostrare un messaggio di errore.
Primary Actor	Utente Autenticato (può essere Utente o Amministratore)

Trigger	L'utente seleziona un bug dalla lista e sceglie l'opzione di modifica

Main Scenario

Step	**Utente**	**System**
1	Accede alla lista dei bug e seleziona un bug da modificare	
2		Verifica che l'utente abbia i permessi per modificare il bug selezionato
3		Mostra la schermata di modifica del bug con i campi editabili (Titolo, Descrizione, Tipologia, Priorità, Stato) precompilati con i valori attuali
4	Modifica uno o più campi del bug (es. cambia lo stato da "In Progress" a "Done", aggiorna la descrizione, modifica la priorità)	
5	Conferma le modifiche cliccando sul pulsante "Salva"	
6		Valida i dati inseriti (titolo non vuoto, campi nel formato corretto)
7		Salva le modifiche nel database
8		Aggiorna la cronologia delle modifiche registrando: timestamp, utente che ha effettuato la modifica, campi modificati
9		Mostra un messaggio di conferma "Bug modificato con successo"
10		Reindirizza l'utente alla vista dettagliata del bug aggiornato

Extensions

Extension #1 - Dati non validi

Step	**Utente**	**System**
6a	<L'utente ha lasciato il campo titolo vuoto o ha inserito dati non validi>	
		Rileva che i dati non sono validi
		Mostra messaggi di errore specifici accanto ai campi problematici (es. "Il titolo è obbligatorio", "Seleziona una tipologia valida")

		Mantiene la schermata di modifica aperta con i dati inseriti
		Ritorna allo step 4 del Main Scenario

Extension #2 - Annullamento modifica

Step	Utente	System
4a	Decide di non salvare le modifiche e clicca su "Annulla" o "Indietro"	
		Scarta tutte le modifiche non salvate
		Reindirizza l'utente alla vista precedente (lista bug o dettaglio bug) senza salvare alcuna modifica
		Il caso d'uso termina

Extension #3 - Errore di sistema durante il salvataggio

Step	Utente	System
7a	<Si verifica un errore di connessione al database o altro errore tecnico>	
		Rileva l'errore durante il tentativo di salvataggio
		Esegue il rollback della transazione per mantenere la consistenza dei dati
		Mostra un messaggio di errore "Impossibile salvare le modifiche. Riprova più tardi."
		Mantiene la schermata di modifica aperta con i dati inseriti dall'utente
		Ritorna allo step 4 del Main Scenario, permettendo all'utente di riprovare

Extension #4 - Amministratore modifica bug assegnato ad altri

Step	Amministratore	Utente Assegnatario	System
8a	<L'amministratore modifica un bug assegnato ad un altro utente>		
			Registra la modifica nella cronologia con l'ID dell'amministratore
			Invia una notifica all'utente assegnatario informandolo della modifica effettuata dall'amministratore
		Riceve la notifica	
			Continua con lo step 9 del Main Scenario

Mockup requisito 9

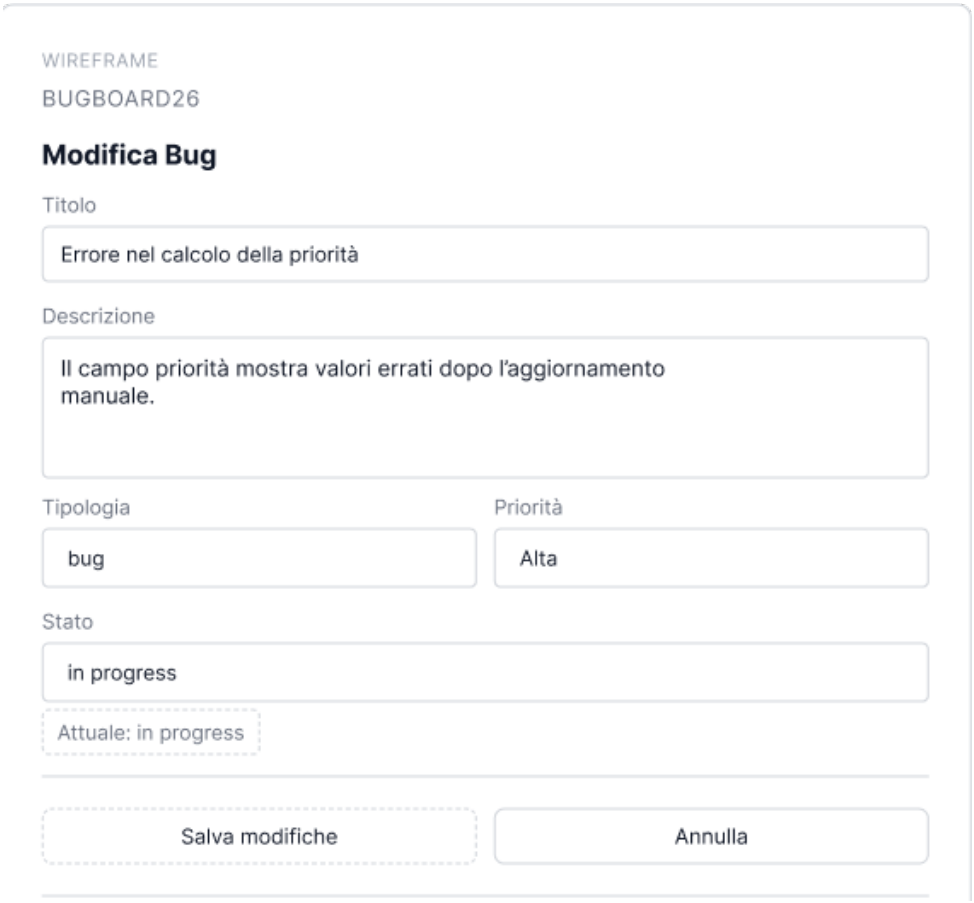

Figura 7: Wireframe del requisito

Appendice: Informazioni Progetto

Stack Tecnologico

- **Front-end**: Angular
- **Back-end**: Spring Boot
- **Database**: PostgreSQL
- **Containerizzazione**: Docker
- **Comunicazione**: API REST JSON su HTTP

Team

- Fabrizio Padulano (N86005058)
- Federico Padulano (N85005043)

Vincoli di Implementazione

- Back-end e front-end completamente separati
- Nessun utilizzo di servizi MBaaS (Firebase, Supabase, etc.)
- HTTPS obbligatorio
- Docker obbligatorio per back-end
- Linguaggio Object-Oriented