Documento di Specifica dei Requisiti Software

Progetto: BugBoard26

Corso di Ingegneria del Software – A.A. 2025/2026

Università degli Studi di Napoli Federico II – DIETI

Indice

1 Introduzione **3**

1.1 Scopo del sistema 3

2 Glossario **4**

3 Target degli utenti **5**

4 Requisiti funzionali principali **6**

5 Requisiti non funzionali **7**

6 Requisiti di sistema **8**

7 Modellazione dei casi d'uso **9**

7.1 Requisito 8 9

1. Introduzione

1.1 Scopo del sistema

BugBoard26 è una piattaforma collaborativa per la **gestione di issue e bug** nei progetti software. Consente ai membri di un team di:

- segnalare problemi e richieste di nuove funzionalità;
- assegnare, commentare e risolvere bug;
- monitorare l'attività del gruppo di sviluppo tramite dashboard e report.

Il sistema è progettato per garantire **sicurezza, tracciabilità e collaborazione**, separando la logica di back-end (gestione dati e API) dal front-end (interfaccia utente).

2. Glossario

Termine	Descrizione
BugBoard26	Piattaforma per la gestione collaborativa di issue in progetti software.
Issue	Segnalazione creata da un utente per indicare un problema, domanda o suggerimento.
Bug	Tipo di issue che segnala un malfunzionamento.
Segnalatore	Utente che crea un'issue e riceve notifiche sugli aggiornamenti.
Amministratore	Utente con privilegi completi: può creare utenti, assegnare bug, archiviare, generare report.
Utente autenticato	Utente registrato che ha effettuato il login. Può segnalare, commentare e modificare le proprie issue.
Utente non autenticato	Visitatore non ancora loggato; può solo autenticarsi.
Utente readonly (stakeholder)	Utente con permessi di sola lettura.
Dashboard	Vista riepilogativa di dati e statistiche.
Etichetta (label)	Tag personalizzato associabile a un bug (es. *frontend*, *urgente*).
Priorità	Livello di urgenza assegnato a un bug (*bassa*, *media*, *alta*).
Stato	Condizione del bug (*todo*, *in progress*, *done*, *archiviato*).
Commento	Messaggio testuale associato a una issue.
Notifica	Avviso automatico di eventi (assegnazione, risoluzione, scadenze, ecc.).
Cronologia modifiche	Registro di tutte le modifiche di una issue con autore e data.
Report mensile	Documento con statistiche aggregate sulle attività del team.
Archiviazione	Operazione che sposta un bug risolto in archivio.
Duplicato	Bug segnalato più volte e chiuso come copia di un altro.
Scadenza	Termine entro cui una issue deve essere risolta.
Esportazione	Funzione per salvare l'elenco bug in CSV, Excel o PDF.
Sistema distribuito	Architettura con separazione tra back-end (logica e dati) e front-end (interfaccia).
Back-end	Componente server che gestisce la logica di business e la persistenza dei dati.
Front-end	Interfaccia utente (desktop, web o mobile) che comunica con il back-end via API REST.

3. Target degli utenti

Categoria	Descrizione	Obiettivi principali
Amministratori	Membri del team con responsabilità di gestione.	Gestire utenti, assegnare bug, creare report e monitorare l'attività complessiva.
Utenti autenticati (sviluppatori)	Membri del team che segnalano, correggono o commentano bug.	Segnalare issue, aggiornare lo stato, collaborare nella risoluzione dei problemi.
Utenti readonly (stakeholder)	Figure esterne con permessi di sola lettura.	Monitorare l'avanzamento dei lavori e consultare i bug aperti.
Utenti non autenticati	Visitatori o nuovi membri.	Accedere al sistema tramite login o registrazione gestita dall'amministratore.

4. Requisiti funzionali principali

ID	Descrizione funzionale	Attori
RF1	**Autenticazione sicura basata su email e password.**	Utente non autenticato
RF2	**Creazione e gestione di utenti da parte dell'amministratore.**	Amministratore
RF3	**Segnalazione di issue con titolo, descrizione, tipo, priorità e immagine.**	Utente autenticato
RF4	**Filtraggio e ordinamento delle issue per tipo, stato o priorità.**	Utente autenticato, Amministratore
RF5	**Assegnazione dei bug e invio notifica all'assegnatario.**	Amministratore
RF6	**Commenti associati a ogni bug.**	Tutti gli utenti autenticati
RF7	**Modifica dello stato dei bug assegnati e notifica al segnalatore.**	Utente autenticato
RF8	**Dashboard amministrativa con statistiche e grafici.**	Amministratore
RF9	**Esportazione elenco bug in CSV, Excel o PDF.**	Amministratore
RF10	**Cronologia delle modifiche di ogni issue.**	Utente autenticato, Amministratore
RF11	**Archiviazione dei bug risolti.**	Amministratore
RF12	**Ricerca per parola chiave tra le issue.**	Utente autenticato
RF13	**Gestione di etichette personalizzabili.**	Amministratore
RF14	**Segnalazione e chiusura automatica di bug duplicati.**	Amministratore
RF15	**Suggerimento automatico di assegnazione in base al carico di lavoro.**	Sistema
RF16	**Impostazione di scadenze per le issue.**	Amministratore
RF17	**Modalità readonly per stakeholder.**	Stakeholder
RF18	**Visualizzazione cronologia attività e report mensili.**	Amministratore

5. Requisiti non funzionali

Categoria	Descrizione
Sicurezza	Le credenziali devono essere cifrate. Accessi autenticati e sessioni sicure.
Usabilità	Interfaccia intuitiva, tempi di risposta inferiori a 3 secondi nelle operazioni standard.
Affidabilità	Il sistema deve garantire integrità dei dati e resistenza agli errori di rete.
Scalabilità	Possibilità di gestire più progetti e utenti contemporaneamente.
Disponibilità	Il sistema deve poter essere distribuito su infrastruttura cloud (es. AWS, Azure).
Portabilità	Front-end indipendente dal back-end; il sistema deve funzionare come web app, desktop o mobile.
Manutenibilità	Codice modulare e documentato; logica separata in componenti indipendenti.
Prestazioni	Operazioni critiche (login, ricerca, assegnazione) completate in meno di 2 secondi.
Compatibilità	Supporto ai principali browser e sistemi operativi.
Tracciabilità	Ogni azione (creazione, modifica, assegnazione) deve essere registrata nella cronologia.

6. Requisiti di sistema

Componente	Requisiti
Back-end	Applicazione server REST che gestisce autenticazione, business logic, persistenza e notifiche. Deve essere indipendente dal front-end e distribuibile tramite container (es. Docker).
Front-end	Interfaccia utente che comunica con le API REST. Può essere implementata come SPA (React/Angular) o applicazione desktop/mobile.
Database	Sistema relazionale o documentale persistente. Deve mantenere informazioni su utenti, bug, commenti, etichette e cronologia.
API REST	Endpoint sicuri per autenticazione, gestione issue, commenti, dashboard e report.
Sistema di notifiche	Servizio interno per inviare email o notifiche push su eventi rilevanti (assegnazioni, risoluzioni, scadenze).

7. Modellazione dei casi d'uso

7.1 Requisito 8

Titolo	Esportazione bug in vari formati
Attori primari	Utente
Scopo	Permettere all'utente di esportare i bug selezionati nei formati CSV, Excel, PDF o JSON.
Precondizioni	L'utente è stato autenticato e dispone delle autorizzazioni richieste. Almeno un bug è selezionato dalla lista.
Postcondizioni	Viene fornito un file nel formato richiesto, contenente i bug selezionati.
Scenari principali	1. L'utente seleziona i bug dalla lista. 2. L'utente avvia l'esportazione (cliccando un pulsante di esportazione o scenario simile). 3. Il sistema chiede all'utente di scegliere il formato (CSV, Excel, PDF o JSON). 4. L'utente seleziona il formato desiderato. 5. Il sistema fornisce il file contenente i bug selezionati nel formato scelto. 6. Il sistema rende il file disponibile per il download.
Estensioni	1. Se l'utente non ha selezionato alcun bug, il sistema mostra un messaggio di errore e termina il caso d'uso. 2. Se l'utente chiude la finestra di dialogo, il caso d'uso termina senza effettuare l'esportazione. 3. Se si verifica un errore durante la creazione del file, il sistema visualizza un errore e termina il caso d'uso.