BugBoard

A.A. 2025-26

Splat your bugs away.

16/10/2025

Giovanni Adamo	**N86004180**
Giuseppe Costabile	**N86003576**

Sommario

0. Premessa

Il presente artefatto, come richiesto dal committente, funge da guida completa all'utilizzo del Software BugBoard, presentando una dettagliata documentazione e studio dell'intero percorso di sviluppo di tale Software. Il documento è stato interamente sviluppato da Giovanni Adamo e Giuseppe Costabile per il CDL in Informatica presso l'*Università degli Studi di Napoli Federico II.* Tutto l'operato è soggetto a licenza di tipo MIT ed è possibile visualizzarlo e scaricarlo via GitHub.

Info e contatti in fondo alla pagina corrente.

0.1 Struttura del documento

Il contenuto della documentazione è così suddiviso:

- *Parte A: Documento di Specifica dei Requisiti Software.*
- *Parte B: Documento di Design del Sistema.*
- *Parte C: Testing e valutazione dell'usabilità.*

La documentazione segue interamente struttura e ordine esposto nella sezione *8. Output attesi dal committente* del documento *Progetto di Ingegneria del Software*, con le seguenti variazioni:

- *Sezione 1*: il *Glossario* è stato spostato in fondo al documento, e il punto *1.e Formalizzazione dei casi d'uso* è stato posizionato subito dopo il *1.b Modellazione dei casi d'uso* per una questione di maggiore chiarezza e coerenza delle informazioni.

È stata inoltre aggiunta una sezione dedicata al riepilogo delle versioni del documento.

0.2 Info e Contatti

Di seguito, alcuni canali di comunicazione tramite i quali è possibile raggiungere gli autori dell'operato, per contatti o visione di ulteriori produzioni (in ambito accademico e/o personale).

Giovanni Adamo (N86004180)

- **Email (istituzionale) :** giovanni.adamo@studenti.unina.it
- **GitHub:** https://github.com/Gazen27

Giuseppe Costabile (N86003576)

- **Email (istituzionale) :** giuse.costabile@studenti.unina.it
- **GitHub:** https://github.com/gospi4

0.3 Presentazione del Software

BugBoard è un applicativo in grado agevolare la gestione di errori durante lo sviluppo Software, migliorandone la risoluzione collaborativa grazie ad un'ambiente intuitivo e facile da navigare. Il sistema prevede la possibilità di condividere con l'utenza le proprie richieste, specificandone il tipo, la priorità, i tag associati e altri parametri, grazie ai quali sarà possibile effettuare una ricerca più approfondita. È necessario autenticarsi per utilizzare i servizi dell'applicativo, il quale avrà una vista differenziata per l'utenza semplice e per l'amministrazione.

Le funzionalità assegnate al gruppo sono: 1, 2, 3, 10.
Eventuali funzionalità extra sono indicate all'interno del documento.

L'applicativo, nello specifico, consente di:

- Segnalare una nuova issue inserendola a sistema, specificandone obbligatoriamente titolo, descrizione e tipologia. In fase di caricamento, inoltre, è possibile associarne anche ulteriori specifiche (opzionali), quali priorità, tag e caricamento di un'immagine.
- Per gli account di tipo amministratore, sarà possibile iscrivere al sistema nuovi utenti (semplici o, a loro volta, amministratori); solo gli utenti semplici possono iscriversi autonomamente al sistema senza ausilio di un admin.
  Funzionalità extra: gli amministratori hanno accesso all'eliminazione globale di richieste.
- Ogni utente può ottenere un elenco riepilogativo delle issues presenti sul sistema, tramite la sezione *sfoglia richieste*; per filtrare e ordinare al meglio i risultati, è disponibile la *ricerca avanzata*. È possibile visualizzare nel dettaglio ogni richiesta interessata.
- Funzionalità extra: è possibile aggiornare a posteriori una richiesta già caricata sul sistema, purché sia di propria appartenenza; dalla voce dedicata, si può scegliere tra aggiornarne lo stato oppure eliminarla definitivamente dall'elenco di issues.

0.4 Versioni

Di seguito sono elencate le varie versioni del documento, con relativo riepilogo e data di rilascio. Tali *releases* corrispondono alle diverse consegne effettuate entro le deadline fornite.

v.0.1 - beta	16/10/2025	Strutturata la base del documento e il layout generale. Stesura completa della parte A, corrispondente alla sezione ***1. Documento di Specifica dei Requisiti Software.***

Parte A:
Documento di Specifica dei Requisiti

1. Casi d'uso

In questa sezione viene effettuato uno studio sui casi d'uso individuati, con un'attenta analisi dei singoli, modellazione tramite Use Case Diagrams e formalizzazione di uno dei casi scelti.

Si noti che il sistema seguirà la struttura indicata dalle funzionalità assegnate al gruppo, ma ciò non preclude la presenza di eventuali funzionalità extra, le quali saranno contrassegnate come tali.

Per ulteriori informazioni, consultare la sezione *0.3 Presentazione del Software*.

1.1 Casi d'uso individuati

Il sistema presenta la necessità di individuare tre attori principali:

- **Guest:** corrisponde ad un utente non ancora autenticato sulla piattaforma.
- **Utente autenticato:** è la figura dell'utente semplice, che ha superato la procedura di autenticazione ed è autorizzato all'uso delle funzionalità proposte.
- **Admin:** l'utenza di amministrazione; è una specializzazione dell'*utente autenticato*, avendo a disposizione tutte le sue funzionalità in aggiunta a quelle amministrative.

Al momento, non sono previsti attori secondari.
Per i casi d'uso individuati, sono elencati come segue e raggruppati per attore:

Guest:

- **Registrazione:** sistema di autenticazione per i nuovi utenti mai registrati alla piattaforma. Il servizio richiede l'inserimento di email e password.
- **Login:** consente l'accesso alla piattaforma ad utenti già registrati; anche in questo caso è previsto l'inserimento di una email e una password entrambi validi.

Utente Autenticato:

- **Registrare nuova richiesta:** il sistema consente di iscrivere una nuova richiesta di supporto da parte degli utenti, specificandone il tipo, il titolo e una descrizione, e aggiungendo dettagli facoltativi come immagini, priorità e tag associati.
- **Gestire richieste** (proprie)**:** da questa funzionalità è possibile aprire un menu di gestione, contenente due opzioni (corrispondenti alle estensioni indicate); si noti che tale operazione è possibile effettuarla soltanto sulle proprie issues.
    - **Aggiornare richiesta:** estensione di *gestire richieste* – consente di aggiornare lo stato della richiesta selezionata (stato di default: *todo*).
    - **Eliminare richiesta:** estensione di *gestire richieste* – consente di eliminare definitivamente dalla piattaforma la richiesta selezionata.
- **Sfoglia richieste:** consente di visualizzare la lista completa di richieste in elenco sulla piattaforma, con una esposizione riepilogativa per ciascuna di esse.
- **Ricerca avanzata:** è possibile, tramite questa funzionalità, visualizzare un elenco personalizzato di richieste in base a dei filtri; i risultati possono essere ordinati secondo criteri scelti dall'utente.

- **Visualizza dettagli richiesta:** una volta ottenuto l'elenco delle issues, è possibile cliccare su una delle richieste proposte per visualizzarne approfonditamente i dettagli.

Admin:

- **Creare nuovo utente:** attraverso tale funzionalità, un amministratore può creare un nuovo utente iscrivendolo al sistema; l'utente creato potrà essere di tipo semplice (*utente autenticato*) o amministratore (*admin*).
- **Eliminare richieste** (globale)**:** è consentito agli amministratori di eliminare singolarmente le richieste iscritte a sistema, anche se non sono le proprie.

1.2 Modellazione dei casi d'uso: Use Case Diagrams

Di seguito, la rappresentazione UML dello Use Case Diagram.
Si noti che i casi d'uso e gli attori individuati sono indicati e ben esposti in *1.1 Casi d'uso individuati*.

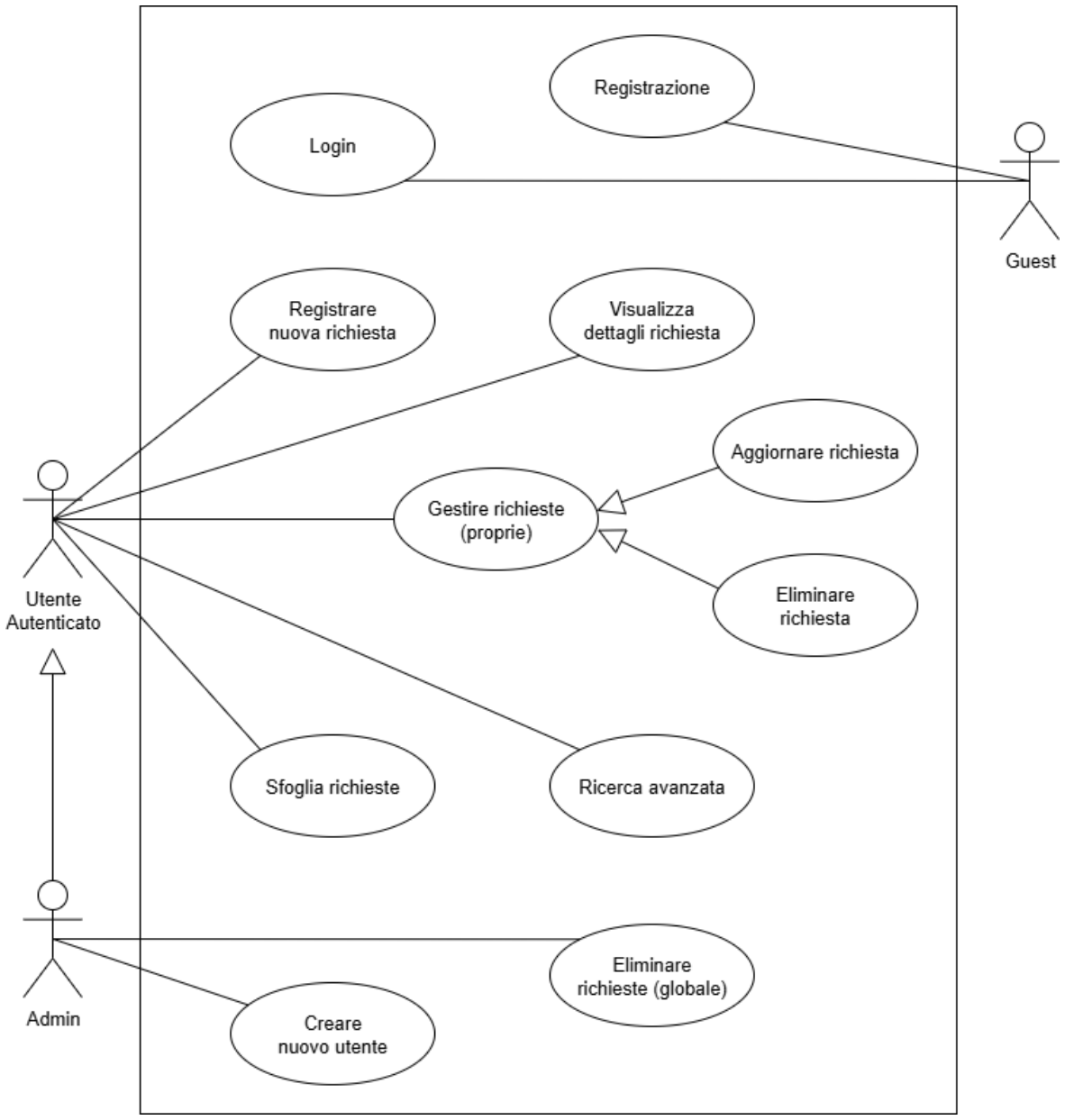


Figura 1 - Use Case Diagram

Attenzione: si noti che i caso d'uso *Gestire richieste (proprie)*, ed *Eliminare richieste (globale)*, corrispondono a funzionalità non presenti nella traccia assegnata a questo gruppo; pertanto, potrebbero essere rimossi durante le prossime releases dell'artefatto.
Verificare la sezione *0.4 Versioni* per maggiori informazioni.

1.3 Formalizzazione dei casi d'uso

Il caso d'uso scelto per essere sviluppato come *Fully-dressed* è: *Registrare nuova richiesta*.

Il motivo di tale scelta è rappresentato dal fatto che questa funzionalità è tra le più complete sul sistema (escluse le procedure di autenticazione), e consente un ampio coinvolgimento da parte dell'utente, il quale interagirà costantemente con le schermate presentate di seguito.

1.4 Prototipazione visuale: Mock-up

La prototipazione visuale del caso d'uso *Registrare nuova richiesta* è sviluppata su multiple schermate, che vanno dalla pagina home (da cui è possibile accedere alle varie funzionalità principali), agli scenari di risoluzione e di errore. Ogni mock-up è spiegato nel dettaglio, prima di passare alla visualizzazione del flusso principale. Si noti che nonostante si tratta di un *Low-Fidelity Prototype*, il prodotto finale sarà quanto più simile possibile in termini di design a quanto mostrato.

M1. Schermata Home

In questa schermata è possibile visualizzare le principali funzionalità offerte dal sistema; la voce *Nuova Richiesta* è quella interessata, e consente di registrare una nuova richiesta di supporto al sistema. *Sfoglia richieste* e *Ricerca Avanzata* servono ad effettuare una ricerca delle issues già presenti: nel secondo caso, è possibile usufruire di una ricerca più accurata previa inserimento dei parametri desiderati.

M2. Informazioni essenziali

È il primo dei tre step necessari per registrare una nuova issue; i dati richiesti in questa sezione sono obbligatori, in quanto l'assenza di ognuno di essi priverebbe la richiesta di riconoscibilità, e risulterebbe impossibile trovarla, distinguerla e comprenderla. Tale schermata, dunque, consente l'inserimento di un titolo, una descrizione, e una tipologia tra quelle proposte nel menu a tendina.

Provando a proseguire senza indicare uno o più dei parametri richiesti, il sistema non consentirà di proseguire oltre, evidenziando in rosso i campi assenti, e notificandolo con un breve messaggio in fondo alla pagina corrente. Una volta inseriti i dati mancanti, sarà possibile proseguire.

M3. Ulteriori informazioni (opzionali)

Nella sezione mostrata, è invece possibile inserire ulteriori informazioni, qualora si volesse offrire un livello di dettaglio superiore alla propria richiesta. I tag (etichette personalizzabili) sono scelti dall'utente e convalidati dopo l'aggiunta di uno spazio. La priorità è a scelta tra quelle proposte (bassa, normale, alta), ed è possibile allegare un'immagine per maggiore chiarezza.

M4. Schermata di riepilogo

In questa schermata vengono mostrati i dati inseriti dall'utente prima della pubblicazione, in modo da revisionarli e poterli modificare tornando indietro mediante l'apposito pulsante situato in basso. Premendo il tasto *Pubblica*, la richiesta viene validata e, nel caso non ci siano problemi di rete, iscritta al sistema, presentandola con stato iniziale *todo* nell'elenco di issue disponibili.

M5. Successo: Schermata di reindirizzamento

Una volta completato l'inserimento di tutti i dati e risolti eventuali errori, sarà visualizzata una pagina che mostra un messaggio di successo, accompagnata dalle possibili opzioni che il sistema offre: tornare alla home, oppure eseguire una ricerca (semplice o avanzata) delle issues; da questo momento, la richiesta appena registrata, comparirà nell'elenco di issues iscritte al sistema.

M6. Caricamento immagine fallito

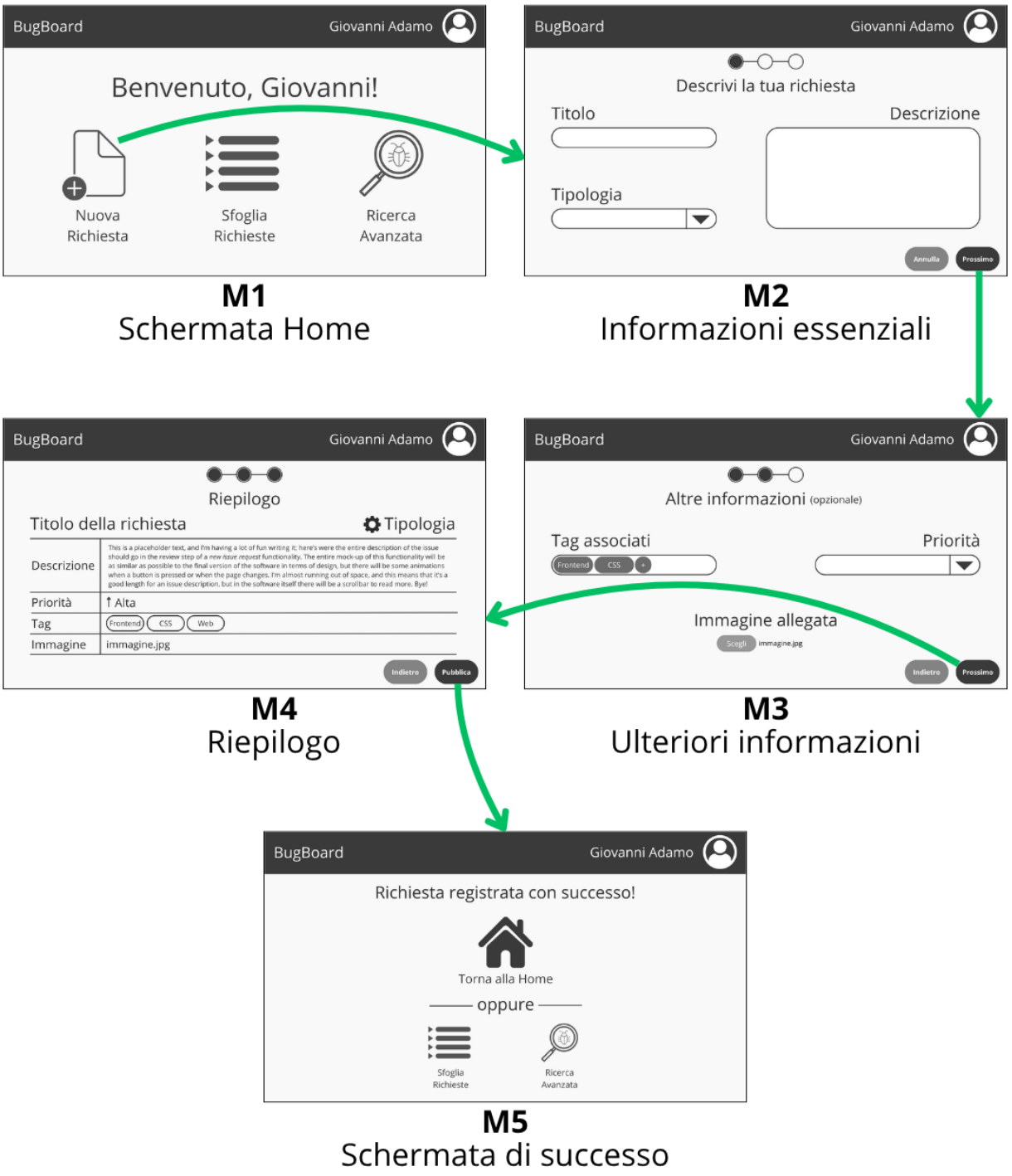
M7. Errore di connessione al Database

1.5 Prototipazione visuale: Flusso principale ed estensioni

Per mostrare la sequenza di schermate che il sistema visita a seguito delle interazioni con l'utente, è possibile verificare diversi scenari; per una maggiore chiarezza nella lettura, sono stati separati in tre diversi flussi, in modo da evitare spiacevoli intrecci di frecce ad intralciare la visibilità.

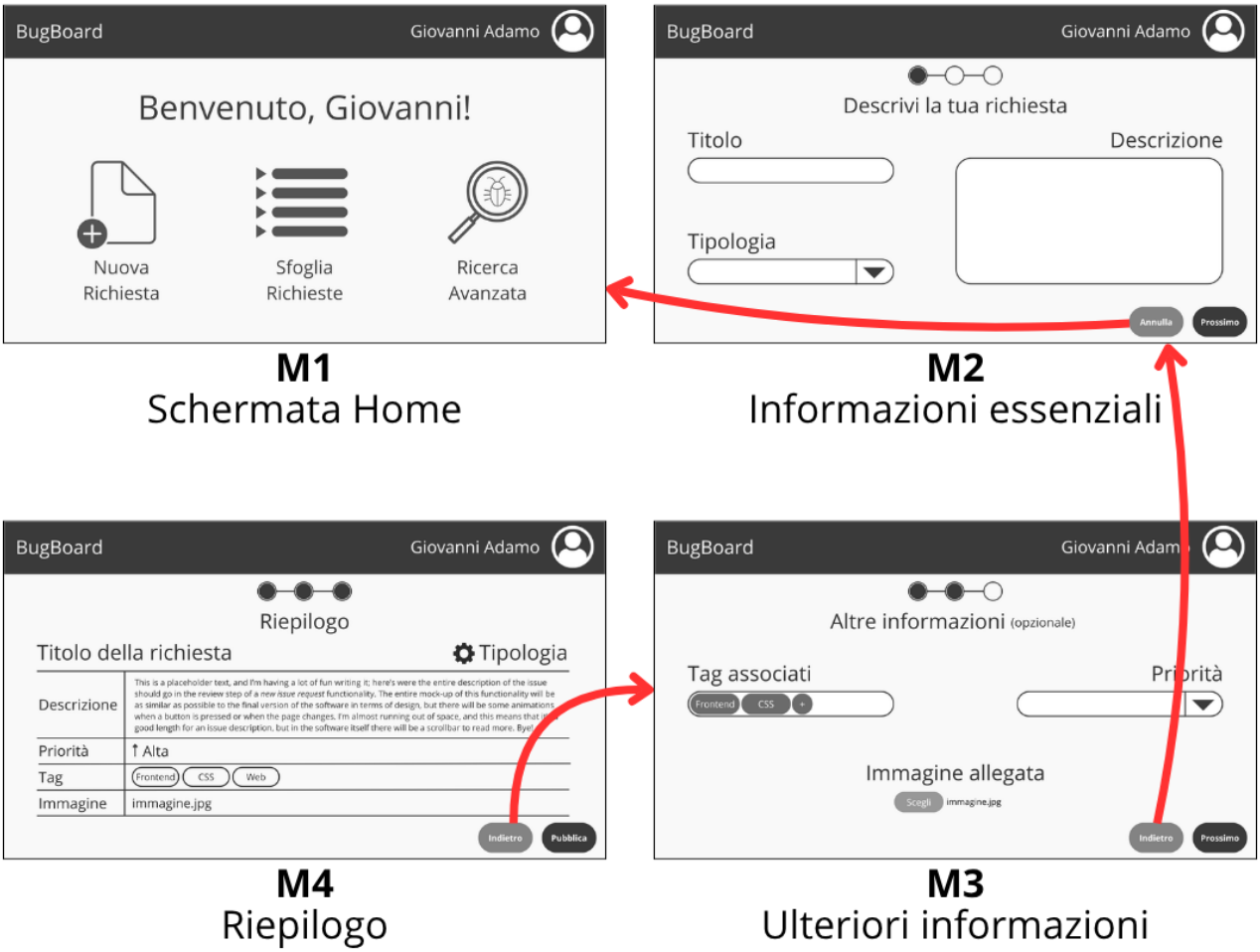
Flusso Principale

Questo flusso è la reazione del sistema prevista nel caso in cui non ci siano problemi o modifiche da parte dell'utente. I trigger di passaggio tra una schermata all'altra, come si può notare dal punto di partenza delle frecce, corrispondono ai tasti di progressione in basso a destra, mentre nel primo caso, al click sul tasto *Nuova Richiesta*.



Flusso “inverso”

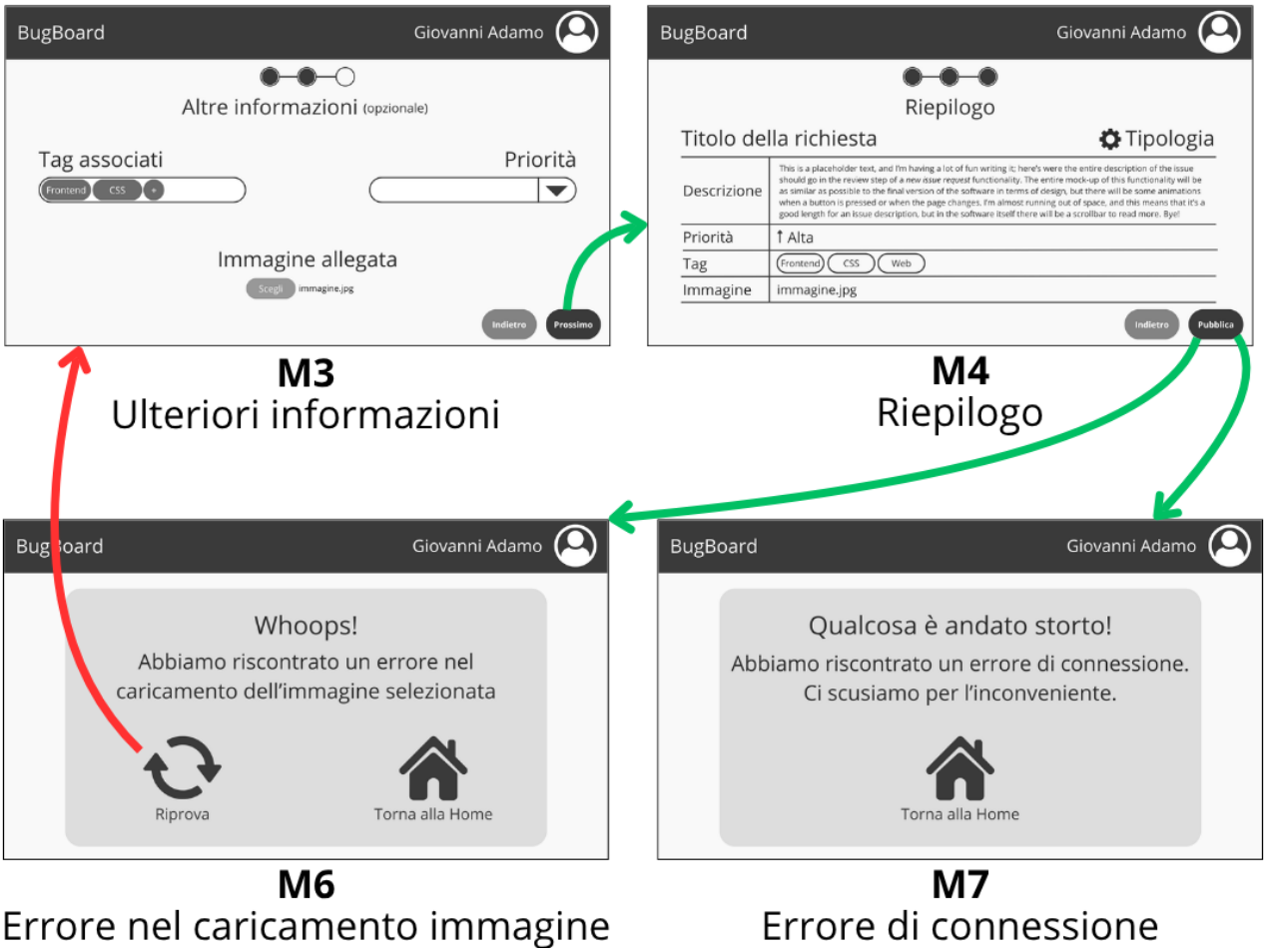
Non è propriamente un flusso sequenziale, ma mostra cosa succede in ogni schermata se l’utente decide di revisionare qualcosa o di annullare il processo, tornando dunque alla schermata precedente; ciò avviene mediante i tasti *indietro* o *annulla*, situati di fianco ai tasti di progressione.



Si noti che tale “interruzione” del flusso regolare, tornando alla pagina precedente, può avvenire in qualsiasi step del *main scenario* e in qualsiasi schermata, tranne che in quella di successo, dalla quale sarà invece possibile di tornare alla home (*M1*) oppure effettuare una ricerca.

Estensioni

Possono presentarsi casi in cui il sistema riscontra un problema; ciò può avvenire nel momento di validazione della richiesta, come segue. Sono possibili due principali casi di errore: problemi con il caricamento di un'immagine (*M6*) oppure un errore di connessione al Database (*M7*); il flusso sottostante mostra cosa succede in presenza di uno dei due errori.



Si noti che nel caso di assenza di uno dei parametri obbligatori (*M2*), il sistema impedirà semplicemente di proseguire, evidenziando in rosso i campi mancanti, come indicato (*M2a*).

M2a. Campi obbligatori assenti

1.6 Descrizione strutturata: Tabella di Cockburn

Il caso d'uso *Registrare nuova richiesta* è descritto mediante formalismo tabellare di A. Cockburn. Gli step del *main scenario* e delle *extensions* nella tabella sottostante fanno riferimento ai prototipi visuali esposti nella sezione *1.4 Prototipazione visuale*.

USE CASE #1	*Registrare nuova richiesta*
Obiettivo	Un utente vuole registrare una nuova richiesta di supporto issue
Precondizioni	L'utente deve essersi autenticato al sistema
Postcondizione di Successo	Il sistema registra una nuova richiesta in elenco con stato *todo*
Postcondizione di Fallimento	La richiesta viene rifiutata e l'utente è reindirizzato alla pagina home
Attore principale	Utente (autenticato)
Trigger	L'utente clicca su *Nuova richiesta* dalla pagina home

Tabella 1.1 - Caso d'uso: Registrare nuova richiesta

Main Scenario	**Step**	**Utente (autenticato)**	**Sistema**
	1	Clicca su *Nuova richiesta* in *M1*	
	2		Mostra *M2*
	3	Compila i campi obbligatori e clicca *Prossimo*	
	4		Mostra *M3*
	5	Aggiunge eventuali informazioni aggiuntive e clicca *Prossimo*	
	6		Mostra *M4*
	7	Controlla i dati inseriti e clicca *Pubblica*	
	8		Valida la richiesta e Mostra *M5*

Tabella 1.2 - Caso d'uso: Registrare nuova richiesta - Main Scenario

Durante il flusso principale (*main scenario*) è possibile che qualcosa vada storto, o che l'utente decida di revisionare qualcosa. Di seguito, le diverse estensioni possibili, mostrate in tabella.

Extension #1 (l'utente non inserisce campi essenziali in *M1*)	**Step**	**Utente (autenticato)**	**Sistema**
	3a	Almeno uno dei campi obbligatori non è stato compilato e clicca *Prossimo*	
	4a		Non consente il proseguimento (Mostra *M2a*)
Extension #2 (problemi con il caricamento dell'immagine)	**Step**	**Utente (autenticato)**	**Sistema**
	5b	Aggiunge eventuali informazioni aggiuntive ma l'immagine è in un formato non consentito e clicca *Prossimo*	
	6		Mostra *M4*
	7	Controlla i dati inseriti e clicca *Pubblica*	
	8b		Mostra *M6*
Extension #3 (problemi di connessione al Database)	**Step**	**Utente (autenticato)**	**Sistema**
	7	Controlla i dati inseriti e clicca *Pubblica*	
	8c		Non riesce a connettersi al DB (Mostra *M7*)

Tabella 1.3 - Caso d'uso: Registrare nuova richiesta - Extensions

Come indicato nella sezione *1.5 Flusso principale ed estensioni*, in qualsiasi momento durante la procedura di registrazione di una nuova richiesta, l'utente potrà tornare alla schermata precedente (eccetto durante l'istanza di inizio e quella di fine). Per evitare ridondanze, è sembrato superfluo rappresentarle mediante formalismo tabellare, in quanto si tratterebbe di numerose *extension* identiche, quasi una per ogni step. Per una più chiara rappresentazione, è consigliato consultare la sottosezione *Flusso "inverso"* in *1.5 Flusso principale ed estensioni.*

2. Target utenti: Personas

In questa sezione verrà analizzato il target dell'utenza potenzialmente interessata all'utilizzo del sistema tramite delle apposite schede di presentazione, le quali indicheranno i tratti principali di ogni categoria, a sostegno del perché possano essere propensi all'utilizzo dell'applicativo *BugBoard*; tali cards saranno inoltre accompagnate da una breve spiegazione, a giustificare quali aspetti del sistema siano pertinenti ad una determinata fetta di utenza, come risolve le loro criticità e riesca a soddisfare i propri obiettivi, rimanendo in linea con i loro interessi personali.

Questa esposizione è il risultato di interviste con gli stakeholders e i possibili utilizzatori del sistema (nel nostro caso, ovviamente, in maniera astratta e ideologica).

2.1 Categoria Amministratori

L'utenza di amministrazione copre un ruolo particolare per il sistema; si tratta di utenti capaci di iscrivere a loro volta ulteriori utenti (semplici o di amministrazione) alla piattaforma. È una figura indicata soprattutto a direttori di progetti o leader dei team di sviluppo, in modo da agevolare tutti i membri nella risoluzione di issues e tenere sotto controllo l'intera situazione dal punto di vista delle criticità inerenti allo sviluppo di un software.

Shigeru Miyamoto

Amministratore

Info

72 anni — Direttore

Kyoto — Sposato

Tratti personali

Creativo | Attento | Collaborativo

Biografia

Shigeru Miyamoto è un veterano dell'industria videoludica e una figura di riferimento nella progettazione di esperienze di gioco accessibili. Da sempre interessato a combinare divertimento e innovazione tecnologica, crede che anche gli strumenti di sviluppo debbano essere intuitivi da usare. Abituato a coordinare team di programmatori, designer e artisti, apprezza piattaforme che favoriscano la collaborazione trasparente e la condivisione dei feedback. Ama esplorare nuove tecnologie e trovare modi per migliorare la qualità del lavoro quotidiano.

Obiettivi

- Garantire alta qualità e stabilità dei progetti software, individuando e risolvendo tempestivamente bug e problemi ricorrenti.
- Ottenere una collaborazione efficace tra i membri dei suoi team, facilitando la comunicazione tra gli sviluppatori.
- Disporre di una piattaforma chiara, ordinata e visivamente gradevole, che renda semplice monitorare lo stato dei ticket e le priorità.
- Gestire l'iscrizione dei membri dei suoi team alle piattaforme di risoluzione issues.

Criticità

- Non sopporta la mancanza di interazione tra team: odia quando gli sviluppatori non hanno a disposizione mezzi di confronto, perdendo le comunicazioni sui bug.
- Può trovare difficile adattarsi a strumenti troppo rigidi e poco flessibili, che non consentono approcci creativi al lavoro.
- Diventa impaziente quando i processi di segnalazione o revisione dei bug risultano troppo lenti o complessi da interpretare.

Shigeru Miyamoto: è la figura del direttore in ambiente di sviluppo software; il suo compito è gestire il team e assicurarsi che tutto proceda come dovuto. Come da lui ricercato, gli è possibile gestire l'iscrizione alla piattaforma per i membri dei suoi team e usufruire lui stesso dei servizi proposti, in modo da facilitare la risoluzione delle issues; il tutto sarà inoltre disponibile su una piattaforma chiara, ordinata e visivamente gradevole, esplicitando in particolare lo stato dei ticket e la loro priorità, senza ledere alla propria creatività a causa di ricerche lente e troppo poco chiare.

Per lui la collaborazione è essenziale, e in *BugBoard* può trovare ciò che cerca per il suo team.

2.2 Categoria Utenti

La categoria *utenti* corrisponde alla fetta più corposa di utilizzatori del sistema *BugBoard*. Ogni utente potrà sfruttare i servizi dell'applicativo per pubblicare nuove richieste di aiuto inerenti a diversi ambiti dello sviluppo di un software, avendo a disposizione la possibilità di implementare ogni dettaglio (obbligatorio o facoltativo) per rendere la propria richiesta il più precisa possibile; allo stesso modo, sarà a disposizione degli utenti la possibilità di ricercare nel sistema altre issues, così da collaborare al meglio con altri sviluppatori e/o membri del proprio team di lavoro.

Masahiro Sakurai: è la figura dell'utente operativo, lo sviluppatore che gestisce pienamente l'avanzamento del software dal lato codice; considerato il suo bisogno di spiegare minuziosamente ogni aspetto del suo lavoro, potrà indicare ogni dettaglio nelle sezioni dedicate al momento di pubblicazione di una nuova richiesta, ed è disponibile la possibilità di visualizzare accuratamente le singole issues disponibili dall'elenco riepilogativo in una vista dedicata, in modo da tenere l'aspetto di approfondimento dettagliato separato dalla vista chiara ed immediata ottenuta dopo la ricerca, lasciando l'ambiente il meno confusionario possibile. Inoltre, vi è la possibilità di ricercare ogni tipologia di issues inserendo molteplici parametri, a soccorrere il suo bisogno di individuare all'istante ciò di cui ha bisogno, e la visualizzazione di immagini lo aiuta ad immergersi al 100% nel problema da risolvere, collaborando alle richieste altrui e ricevendo il giusto aiuto per le sue issues.

Per lui l'ordine è molto importante, e *BugBoard* risolve ogni dubbio con chiarezza e pertinenza.

Satoru Iwata

Utente tecnico

Info

55 anni | Redattore tecnico

Sapporo | Sposato

Tratti personali

Empatico | Comunicativo | Paziente

Biografia

Satoru Iwata è uno sviluppatore esperto con forte passione per la chiarezza e la condivisione della conoscenza; crede che una buona documentazione sia parte integrante della qualità del software, e non un compito secondario.
Spinto da un approccio empatico e collaborativo, dedica molto tempo ad analizzare i processi di lavoro e a migliorare la comunicazione tra i diversi membri del team.
Ama leggere saggi tecnici, studiare nuove metodologie di sviluppo e riflettere su come rendere la tecnologia più accessibile a tutti.

Obiettivi

- Garantire che la documentazione dei progetti sia chiara, completa e coerente con lo stato effettivo del software in sviluppo.
- Ridurre le ambiguità nella comunicazione, favorendo una maggiore collaborazione tra sviluppatori e redattori tecnici.
- Utilizzare la piattaforma per monitorare e correggere issue legate a guide, manuali e specifiche in modo efficiente.
- Prendere parte anche alla ricerca di correzione delle issues lato software.

Criticità

- Infastidito quando le piattaforme non distinguono chiaramente tra bug tecnici e problemi di documentazione, rendendo difficile la priorità di risoluzione.
- Nota con disappunto che molti servizi sono poco predisposti al coinvolgimento dell'utenza anche sul lato guide e di redazione tecnica.
- Trova che diversi sistemi abbiano interfacce troppo poco flessibili, ostacolando la gestione delle informazioni condivise.

Satoru Iwata: nonostante questo servizio sembra essere pensato alla stretta condivisione di issues a livello tecnico, riesce in realtà a soddisfare anche la parte di utenza rappresentata da questa identità, ovvero coloro che sono specializzati soprattutto nella redazione tecnica e nella stesura della documentazione inerente al software in sviluppo, senza escludere però la collaborazione sul lato effettivamente operativo. Questa categoria, come indicato nella scheda soprastante, ricerca infatti un ambiente in cui è possibile condividere issues riguardo a guide tecniche e manuali in maniera efficiente, in modo da mettere in contatto sviluppatori e redattori: così facendo, sarà possibile monitorare eventuali problemi legati allo sviluppo in modo da poterli gestire sul lato documentalistico e viceversa. Inoltre, una delle criticità spesso riscontrate dai redattori tecnici, è proprio la mancata differenziazione tra issues tecniche e di documentazione, facilmente riconoscibili su questo sistema grazie ad una visualizzazione riepilogativa delle issues, la quale mostra chiaramente la tipologia di richiesta, e se quest'ultima è legata ad un problema tecnico o di redazione (oltre al fatto che è facilmente possibile risalire alle sole richieste legate alla documentazione grazie alla sezione di filtri e ricerca avanzata).

Per lui la documentazione è fondamentale, e *BugBoard* non ne eclissa la priorità.

Attenzione: le personas individuate rappresentano un'idea dei possibili utilizzatori del sistema, ma non sono state realmente intervistate; sono stati presi come esempio alcuni dei volti più noti di Nintendo, adattando il loro profilo a quello di utenti possibilmente interessati ad utilizzare il servizio, ma rimanendo il più possibile in linea con i loro reali tratti personali e ambienti di interesse, provando a conciliare la realtà con lo scenario desiderato, e rendendo la rappresentazione il più veritiera ed affidabile possibile, nei limiti dell'adattamento.

In memoria di Satoru Iwata

3. Requisiti non-funzionali e di dominio

In questa sezione sono elencati i requisiti non-funzionali e successivamente quelli di dominio; si noti che alcuni requisiti potrebbero essere rimossi successivamente, in quanto potrebbero risultare contrastanti in fase di implementazione con il resto della struttura del sistema sviluppato; al contempo, potrebbero esserne aggiunte di ulteriori. Controllare la sezione *0.4 Versioni*.

3.1 Requisiti non-funzionali

Sono esposti requisiti non-funzionali atti a descrivere le caratteristiche qualitative che il sistema deve possedere per garantire affidabilità, sicurezza e una buona esperienza. Non riguardano le singole funzionalità operative, ma le modalità con cui devono essere realizzate e fornite all'utente.

NF1 - Sicurezza e riservatezza dei dati
Il sistema deve garantire la protezione delle informazioni sensibili gestite, in particolare le credenziali degli utenti (email e password). L'autenticazione deve essere sicura e preservare la riservatezza dei dati, evitando accessi non autorizzati. Le password devono essere conservate in forma *hashed* (*hashing*) mediante algoritmi di sicurezza, assicurando che non siano mai memorizzate in chiaro. È necessario implementare controlli di validazione e meccanismi di protezione contro attacchi comuni (come *SQL injection*, *brute-force attack*, *cross-site scripting*).

NF2 – Usabilità e accessibilità
L'interfaccia utente deve essere intuitiva, chiara e facilmente navigabile. Gli utenti devono poter segnalare, consultare e filtrare le issue in modo rapido, con interazioni semplici e coerenti. Il design deve seguire principi di ergonomia e usabilità, riducendo il numero di azioni necessarie per compiere le operazioni più frequenti. Deve inoltre essere garantita una fruizione corretta sulla piattaforma desiderata (desktop, mobile o web, a seconda della scelta di sviluppo).

NF3 – Prestazioni e affidabilità del sistema
Il sistema deve essere performante, assicurando tempi di risposta in meno di 3 secondi e garantendo una disponibilità minima del 99%, mantenendo l'integrità dei dati anche in presenza di accessi concorrenti. Deve poter gestire un numero significativo di richieste simultanee; la persistenza dei dati e le operazioni di rete devono essere ottimizzate per garantire continuità del funzionamento anche in condizioni di carico elevato. L'infrastruttura sarà ospitata su servizi cloud *AWS*, sfruttando componenti gestiti per la computazione e l'archiviazione dei dati.

NF4 – Architettura modulare e manutenibilità
L'architettura del sistema deve essere modulare, consentendo la separazione netta tra front-end e back-end. Il front-end deve comunicare con il back-end esclusivamente tramite *API REST*, così da garantire indipendenza tra i due componenti e permettere la sostituzione o l'aggiornamento di una parte senza impattare sull'altra. Il design deve essere orientato al riuso e alla facilità di estensione, favorendo la futura implementazione di nuove funzionalità o modifica di quelle esistenti.
La separazione tra front-end e back-end consente una facile distribuzione in ambienti cloud *AWS*.

NF5 – Gestione centralizzata e consistenza dei dati
La gestione dei dati è incentrata sul back-end, che ha la responsabilità di mantenere coerenza e persistenza delle informazioni. Tutte le modifiche ai dati devono avvenire tramite il back-end, il quale opera attraverso meccanismi transazionali (es. proprietà *ACID*), per assicurare uniformità tra le diverse istanze dell'applicazione e prevenire conflitti. La persistenza dei dati è garantita da un servizio di database gestito su cloud, che assicura affidabilità, backup automatici e consistenza.

NF6 – Qualità del codice e principi object-oriented
Il sistema deve essere sviluppato utilizzando un linguaggio di programmazione object-oriented, seguendo i principi di incapsulamento, astrazione e riuso del codice. Devono essere adottate convenzioni di codifica chiare, accompagnate da strumenti di analisi della qualità del codice (es. SonarQube). L'obiettivo è garantire manutenibilità, leggibilità e scalabilità del prodotto software.

3.2 Requisiti di dominio generali

I requisiti elencati descrivono il contesto applicativo in cui il sistema opera, indicando i vincoli imposti dalle norme derivanti dall'ambiente in cui viene utilizzato; sono legati al contesto del problema reale (sicurezza, privacy, integrità, ecc…). È inoltre indicata la fonte da cui essi derivano.

DG1 – Conformità alle normative sulla protezione dei dati
Il sistema deve garantire che il trattamento dei dati personali (email, password, contenuti pubblicati dagli utenti) avvenga nel pieno rispetto del Regolamento Europeo sulla Protezione dei Dati (*GDPR*). Ciò implica che i dati sensibili siano conservati in modo sicuro, accessibili solo agli utenti autorizzati e non condivisi senza consenso esplicito.

Origine: Normativa sulla privacy - Regolamento *GDPR*

DG2 – Sicurezza e integrità delle informazioni
Solo gli utenti autenticati possono accedere ai dati e alle funzionalità del sistema. Tutte le informazioni gestite (issue, utenti, allegati, impostazioni) devono essere protette contro modifiche non autorizzate, perdite accidentali o accessi impropri. Devono essere adottate misure di autenticazione sicura, cifratura delle password e controlli di integrità sui dati.

Origine: Buone pratiche di sicurezza informatica – *ISO/IEC 27001*

DG3 – Tracciabilità delle operazioni
Ogni azione significativa (come creazione, modifica o eliminazione di issue, gestione utenti) deve poter essere tracciata, associando l'operazione all'utente che l'ha eseguita e al momento in cui è avvenuta. Questa caratteristica è necessaria per garantire sicurezza in contesti collaborativi.

Origine: Buone pratiche di *accountability* – derivate dal *GDPR*

DG4 – Affidabilità e disponibilità del servizio
Il sistema deve garantire un'elevata disponibilità dei servizi e la possibilità di recupero dei dati in caso di malfunzionamenti, con l'obiettivo di preservare la continuità operativa dei team di sviluppo che fanno affidamento sulla piattaforma, in contesto privato o aziendale.

Origine: Requisiti di qualità del software – standard *ISO/IEC 25010*

3.3 Requisiti di dominio specifici

In questo caso, i requisiti di dominio indicati sono specifici del sistema BugBoard, seguendo dunque i vincoli indicati dal committente e dall'ecosistema interno all'applicativo.
Non sono indicate le fonti da cui derivano i requisiti, in quanto coincidono con le richieste del committente (in questa situazione, la traccia del progetto fornita). Dunque, il dominio del sistema è un ambiente di gestione collaborativa di bug e issues in progetti di sviluppo software.

DS1 – Struttura delle issues
Ogni issue rappresenta un elemento di lavoro da gestire all'interno del progetto. Essa è caratterizzata da attributi fondamentali quali titolo, descrizione, tipologia, priorità, stato, eventuale immagine allegata e insieme di etichette personalizzabili (tag).

DS2 – Ruoli e permessi
Nel dominio del sistema esistono ruoli con privilegi differenti: l'utente standard, che può segnalare e visualizzare issue, e l'amministratore, che in aggiunta gestisce gli utenti sul sistema. Ogni ruolo definisce livelli di accesso differenti e operazioni consentite all'interno dell'applicazione.

DS3 – Etichette e classificazione
Le etichette sono elementi di categorizzazione che permettono di associare ad ogni issue parole chiave o tag personalizzati (es. *frontend*, *urgente*, *sicurezza*). Esse consentono una migliore organizzazione e una più rapida individuazione dei bug durante le fasi di ricerca e filtraggio.

DS4 – Ciclo di vita delle issues
Ogni issue segue un ciclo di vita definito da una sequenza di stati che ne rappresentano l'avanzamento nel processo di gestione (es. *todo* → *done*). Il sistema deve mantenere traccia delle transizioni per garantire tracciabilità e trasparenza.

4. Glossario

In questa sezione è racchiuso un insieme della terminologia utilizzata nella stesura del documento. Si noti che la sezione *Glossario* sarà aggiornata dopo ogni versione della documentazione, a seguito delle scelte prese in merito alle tecnologie da impiegare per lo sviluppo del software.

A

ACID
Acronimo che descrive le proprietà fondamentali di una transazione in un sistema di Database, le quali garantiscono affidabilità e coerenza dei dati. Esse sono:
Atomicità - ogni transazione è indivisibile: o viene eseguita completamente, o non viene eseguita.
Consistenza - una transazione può portare il DB solo da uno stato valido a un altro stato valido.
Isolamento - le transazioni concorrenti non interferiscono tra loro.
Durabilità - una transazione rimane permanente anche in caso di guasti del sistema.

API (Application Programming Interface)
Insieme di regole e specifiche che consentono ad un componente software di interagire con un altro, stabilendo metodi e protocolli da utilizzare. In questo contesto, le *API* permettono, ad esempio, la comunicazione tra *front-end* e *back-end*, consentendo lo scambio di informazioni (come l'invio di una nuova issue al server).

API REST
Un particolare tipo di *API*, ovvero un'interfaccia che segue i principi architetturali del modello *REST*. Utilizzano protocolli *HTTP* standard (come *GET, POST, PUT, DELETE*) per consentire operazioni sulle risorse. In questo caso, costituiscono il canale di comunicazione tra *client* e *server*, consentendo operazioni sul database, come recuperare la lista delle issue o aggiornare il loro stato, e garantendo l'indipendenza tra interfaccia utente e logica applicativa.

AWS (Amazon Web Services)
Piattaforma di servizi cloud fornita da Amazon che offre un insieme di servizi (anche *on-demand*) per la computazione, l'archiviazione e la gestione dei dati attraverso Internet. In questo contesto, *AWS* viene utilizzato per l'infrastruttura cloud del sistema, includendo componenti per la computazione (*Cloud Computing*), l'archiviazione (*S3*) e il Database, garantendo scalabilità, affidabilità e continuità operativa.

B

Brute-Force Attack
Si tratta di una tecnica di attacco informatico che consiste nel tentativo di indovinare credenziali di accesso (spesso password) provando tutte le combinazioni possibili. È un metodo efficace contro sistemi che non adottano misure di sicurezza adeguate, come il blocco dopo un numero limitato di tentativi o la verifica *captcha*. Per contrastarlo, sarà implementato un meccanismo di protezione come l'*hashing* sicuro delle password.

C

Cross-Site Scripting (XSS)
È una vulnerabilità di sicurezza delle applicazioni web che consente di iniettare codice malevolo in pagine visualizzate da altri utenti. Questa tecnica può consentire il furto di credenziali o dati sensibili, oltre a manipolare l'interfaccia dell'applicazione. Per prevenire gli attacchi *XSS*, è necessario validare e sanificare gli input utente.

H

Hashing
Processo crittografico che trasforma un dato in ingresso (come una password) in una stringa di lunghezza fissa chiamata *hash*; si tratta di una trasformazione non reversibile, quindi non è possibile risalire al dato originale partendo dall'*hash*. Viene utilizzato per memorizzare le password in modo sicuro: al momento dell'autenticazione, il sistema confronta l'*hash* della password fornita con quello salvato nel database. Questo processo protegge il sistema da numerosi attacchi.

HTTP (Hypertext Transfer Protocol)
Protocollo di comunicazione utilizzato per lo scambio di informazioni tra client e server nel web. È la base del funzionamento delle applicazioni web e definisce le regole per la richiesta e la risposta di risorse. Le operazioni più comuni si basano sui metodi *GET*, *POST*, *PUT* e *DELETE*, spesso utilizzati anche nelle *API REST*.

L

Low-Fidelity Prototype
Rappresentazione semplificata dell'interfaccia utente di un sistema software (prototipo a bassa fedeltà). Si tratta di una bozza visiva che mira a comunicare la struttura, la disposizione dei contenuti e le principali funzionalità, senza concentrarsi troppo sugli aspetti grafici o sui dettagli estetici. Realizzato in fase iniziale del progetto per validare le scelte di design prima dello sviluppo effettivo. Differentemente, l'*High-Fidelity Prototype* riproduce fedelmente l'aspetto visivo e il comportamento interattivo finale del sistema. In questo documento, è stato presentato un *Low-Fidelity Prototype* per la prototipazione visuale della funzionalità di inserimento nuova richiesta.

R

REST (Representational State Transfer)
Modello architetturale per la progettazione di sistemi distribuiti, in cui la comunicazione tra componenti avviene tramite risorse identificate da *URL* e manipolate attraverso un'interfaccia. Definisce una serie di principi, tra cui separazione tra *client* e *server*, assenza di stato (*statelessness*), e utilizzo di risorse rappresentabili in formati standard.
L'architettura *REST* è l'applicazione pratica dei principi *REST* nella progettazione di servizi web.

S

SonarQube
Piattaforma *Open Source* per l'analisi del codice sorgente, utilizzata per valutare la qualità del software in termini di manutenibilità, sicurezza e affidabilità. Il sistema analizza il codice per individuarne le vulnerabilità, fornendo report dettagliati. Può essere integrato nei processi di sviluppo per garantire che il codice rispetti standard di qualità predefiniti. SonarQube è uno strumento utile per assicurare la qualità e la manutenibilità del codice, in linea con i requisiti non funzionali relativi alla qualità del software presentati.

SQL (Structured Query Language)
È un linguaggio standard utilizzato per l'interrogazione e la gestione di basi di dati relazionali. Consente di creare, modificare, cancellare e consultare dati all'interno di un database mediante comandi come *SELECT*, *INSERT*, *UPDATE* e *DELETE*. In questo contesto, *SQL* può essere impiegato dal back-end per gestire le informazioni relative a utenti, issue e altre entità del sistema.

SQL Injection
Si tratta di una tecnica di attacco informatico che sfrutta la mancanza di validazione degli input utente per inserire all'interno di una query *SQL* comandi malevoli. Può consentire di accedere, modificare o eliminare dati in modo non autorizzato. Per prevenirla, è necessario utilizzare query parametrizzate, sistemi di validazione e meccanismi di escaping dei dati immessi dall'utente.
In *BugBoard*, la protezione da *SQL Injection* è un requisito di sicurezza fondamentale.