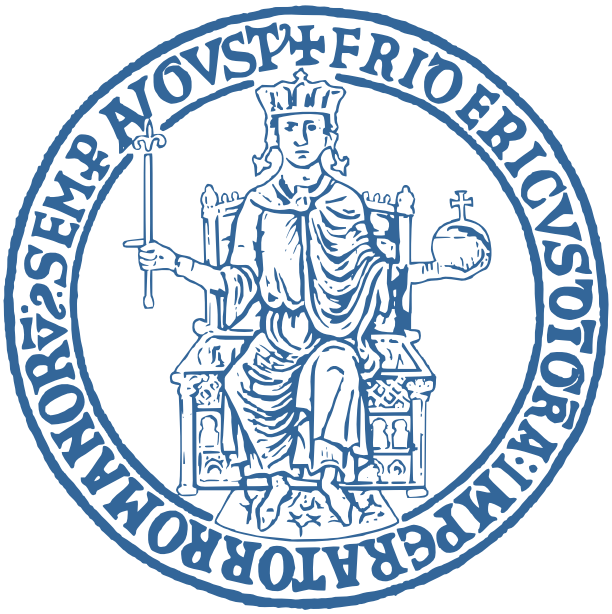
UNIVERSITÀ DEGLI STUDI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI INGEGNERIA DEL SOFTWARE

ANNO ACCADEMICO 2025/2026

Progettazione e sviluppo della piattaforma BugBoard26

Indice

1 INTRODUZIONE

1.1 Chi siamo

Benvenuto su **BugBoard26**!
BugBoard26 è una piattaforma di *issues handling* che fornisce una soluzione unica per:

- Dividere in modo facile developer in progetti.
- Segnalare e gestire intuitivamente issue di vario tipo.
- Gestire in modo efficiente tutte le persone coinvolte in un progetto (anche non sviluppatori) tramite una gerarchia di utenze.

Glossario

issue il "problema" identificato all'interno di un progetto che concerne l'utente.

piattaforma Vedi sistema.

sistema BugBoard26.

2 INGEGNERIA DEI REQUISITI

2.1 Casi d'uso

In questa sezione ci interesseremo all'individuazione dei casi d'uso. Come si può evincere dallo use case diagram riportato qui di seguito:

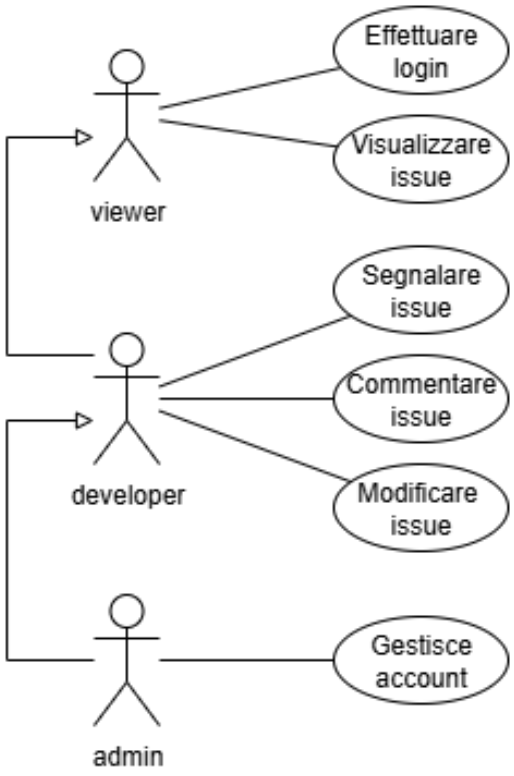


Figura 1: Use Case Diagram

Tutti gli utilizzatori della piattaforma possono essere divisi in tre grandi macrocategorie:

- **Viewer**: individuati anche nella figura di uno stakeholder. Sono utilizzatori, non necessariamente del settore, che hanno comunque interesse a visualizzare le issue legate al progetto senza poterle però aggiungere o modificare.
- **Developer**: rappresentano la stragrande maggioranza di utilizzatori della piattaforma. I developer sono coloro che contribuiscono attivamente all'individuazione e risoluzione delle issue.
- **Admin**: rappresentano l'estensione di un developer con permessi di creazione e gestione di altre utenze.

2.2 Individuazione delle personas

Ora esamineremo alcune personas che rispecchiano alcuni dei tipi di utilizzatori della nostra piattaforma.

Nome: Mark Party Età: 52 anni Posizione: Product Owner
Obiettivi: • Sarebbe molto utile poter vedere l'andazzo del team così da sapere in che direzione indirizzarlo e come gestirlo.
Bio: Sono un economista italo-americano di Boston. Nell'arco della mia carriera mi sono ritrovato a gestire diverse start-up e gruppi di lavoro, nonostante non capisca molto di queste diavolerie informatiche, mi ritengo molto più capace a gestire e portare avanti prodotti

Nome: Aleksander Lilia Età: 24 anni Posizione: Developer
Obiettivi: • Per lavorare in modo efficiente devo sapere quali problemi devo sistemare e magari avere del feedback dai miei colleghi. • Nel caso dovessi trovare dei problemi, vorrei avere un modo comodo per segnalarli in modo dettagliato. • Una volta risolti tali problemi vorrei poter segnalarlo al mio team.
Bio: Sono un developer di Izdebki, dopo essermi laureato all'università di Cracovia mi sono trasferito a Napoli per lavoro e per amore. Sono grande amatore della filosofia "work smarter not harder" che cerco di applicare in ogni modo possibile.

Nome: Pierrelouis Frascout Età: 37 anni Posizione: Team leader
Obiettivi: • Voglio poter gestire i membri del mio team in modo chiaro ed efficiente. • Voglio poter tenere traccia dei progressi fatti dal mio team e come si sta comportando. • Voglio condividere con tutte le persone interessate, l'andamento del nostro team.
Bio: Sono un developer di Izdebki, dopo essermi laureato all'università di Cracovia mi sono trasferito a Napoli per lavoro e per amore. Sono grande amatore della filosofia "work smarter not harder" che cerco di applicare in ogni modo possibile.

2.3 Requisiti non funzionali e di dominio

2.3.1 Requisiti non funzionali

I requisiti non funzionali da noi individuati sono:

- **Permanenza dei dati:** attraverso un database non MBaaS.
- **Utilizzo di linguaggi orientati agli oggetti.**
- **Implementazione di un modello Client-Server.**
- **Elevata manutenibilità.**
- **Efficienza e affidabilità:** non essendo la piattaforma safety-critical, limitazioni di tempo e memoria occupata sono da considerarsi standard e ragionevoli.

2.3.2 Requisiti di dominio

Non sono stati individuati requisiti di dominio particolarmente differenti da quelli forniti nella traccia.

2.4 Formalizzazione di un requisito

Qui di seguito riportiamo la formalizzazione di un requisito quale la visualizzazione di una issue, prima mediante il suo mockup:

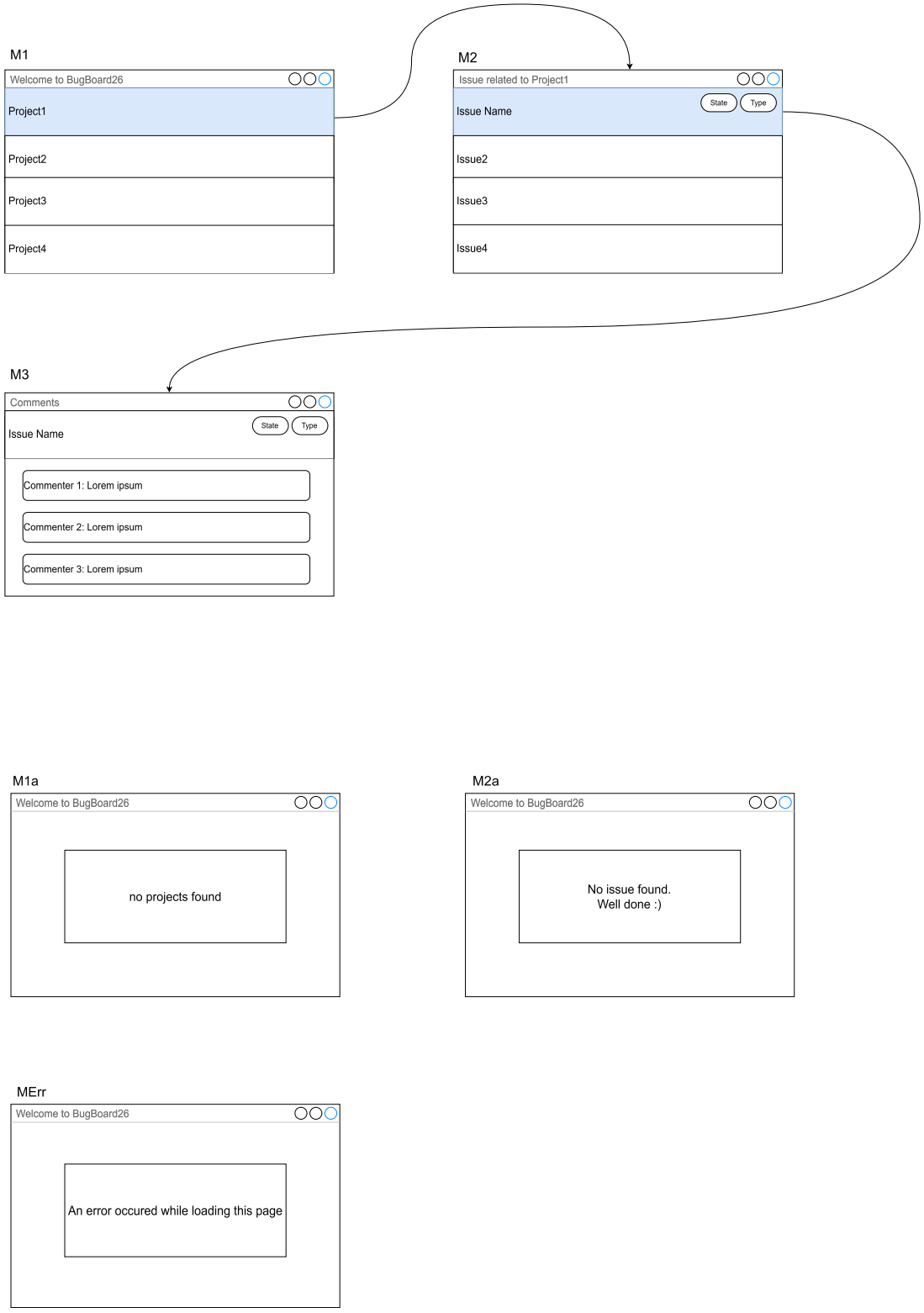


Figura 2: Visualizza issue

E qui di seguito riportiamol'inerente tabella di Cockburn:

USE CASE	*Visualizza issue*
Goal	Un utente vuole visualizzare una issue, le sue proprietà e i commenti.
Preconditions	L'utente ha un account e si è autenticato.
Success end conditions	Il sistema mostra la issue e i suoi commenti.
Failed end conditions	Il sistema mostra una pagina di errore.
Primary actor	Qualsiasi tipo di user.
Trigger	L'utente fa accesso.
Main scenario	**Step n.** / **Utente** / **Sistema** 1 / / Mostra M1 2 / Clicca su un progetto / 3 / / Mostra M2 4 / Clicca su una issue / 5 / / Mostra M3
Extension n° 1 (User has no projects)	1a / / Mostra M1a
Extension n° 2 (Project has no issues)	2b / / Mostra M2b
Extension n° 3 (Generic error)	1,3,5 err / / Mostra MErr

3 SYSTEM DESIGN

3.1 Architettura del sistema

L'architettura del sistema adottata per la realizzazione della piattaforma è di tipo **Client-Server**, più specificatamente a microservizi. I microservizi da noi individuati sono:

- **Servizio di gestione utenze:** si occupa della gestione delle utenze e dell'autenticazione.
- **Servizio di gestione issue:** si occupa della gestione delle issue e dei relativi commenti.

3.2 Decomposizione del sistema

Il sistema è composto dai seguenti componenti principali:

1. **Frontend:** interfaccia utente accessibile tramite browser web.
    - Login/Registrazione
    - Visualizzazione progetti e issue
    - Creazione/modifica progetti e issue
    - Gestione profili utente (solo per admin)
2. **API Gateway:** punto di accesso unico per il client web che smista le richieste ai vari microservizi.
3. **Backend:** microservizi che gestiscono le funzionalità principali della piattaforma.
    - **Utenze:** gestione utenti e profili
    - **Gestione issue:** gestione progetti e issue
4. **Database:** sistema di gestione dei dati persistenti.
    - **Utenze:** memorizzazione dati utenti e profili
    - **Gestione issue:** memorizzazione dati progetti e issue

3.3 Scelta delle tecnologie

Per la realizzazione della piattaforma BugBoard26, abbiamo scelto le seguenti tecnologie:

- **Frontend:** *Angular.ts* per la costruzione dell'interfaccia utente, grazie alla sua modularità e facilità di integrazione con backend RESTful.
- **API Gateway:** *Java con Spring* per il bilanciamento del carico e la gestione delle richieste tra i microservizi.
- **Backend:** *Java con Spring* per la creazione dei microservizi, grazie alla sua robustezza.
- **Database:** *PostgreSQL* per la gestione dei dati relazionali, grazie alla sua affidabilità e scalabilità.