# INTRODUZIONE

Benvenuti 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

|  |  |
| --- | --- |
| Termine | Definizione |
| Issue | Il “problema” identificato all’interno di un progetto che concerne l’utente |
| sistema, piattaforma | BugBoard26 |
|  |  |

# INGEGNERIA DEI REQUISITI

## 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:

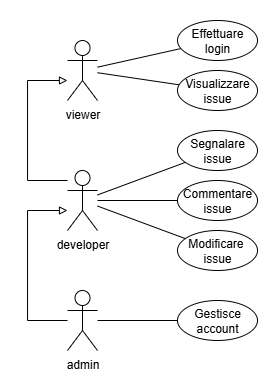


Diagramma 1: Use Case Diagram di BugBoard26

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.

## 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 software engineer di Nantes ma ho vissuto buona parte della mia vita a Roma. Sono una persona risolutiva ed estremamente orientata al pratico e questo si riflette nel mio modo di lavorare |

## Requisiti non funzionali e di dominio

### Requisiti non funzionali

I requisiti non funzionali da noi individuati sono:

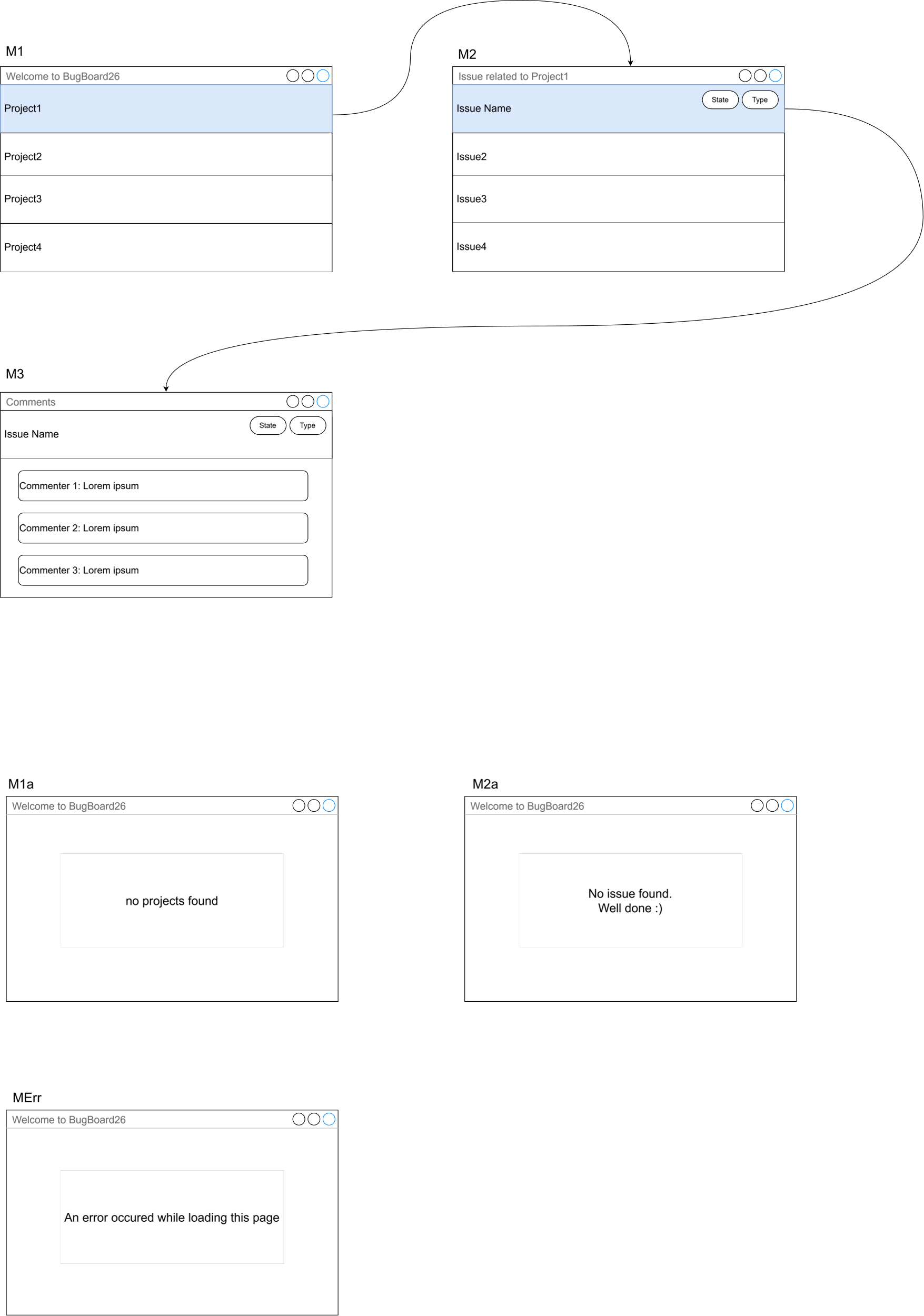
* **Permanenza dei dati:** attraverso un database non MBaaS.
* **Utilizzo di linguaggi orientati agli oggetti.**
* **Implementazione di un modello Client-Server.**
* **Elevata manutenibilità.**
* **Efficienza e affidabilità:** non essendo la piattaforma safety-critical, limitazioni di tempo e memoria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

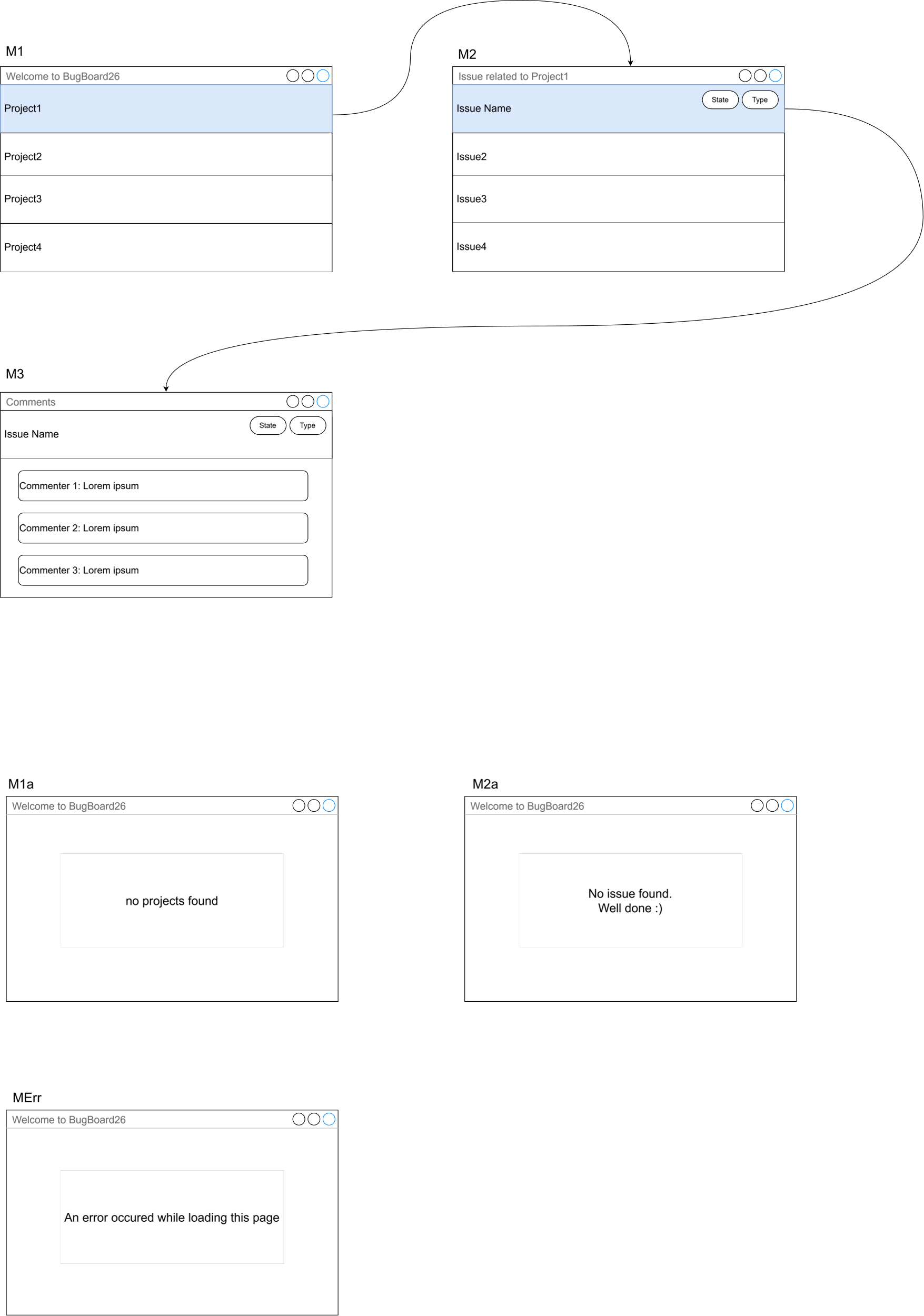
### Requisiti di dominio

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

## Formalizzazione di un requisito

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





E qui di seguito riportiamo l’inerente tabella di Cockburn:

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE | View issue | | |
| Goal | An user wants to view an issue, its properties and its comments | | |
| Preconditions | The user has an account and is authenticated | | |
| Success end conditions | The system shows the issue and its comments | | |
| Failed end conditions | The system shows an error page | | |
| Primary actor | Any type of user | | |
| Trigger | User logs in | | |
| Main scenario | Step n. | User | System |
|  | 1 |  | Shows M1 |
|  | 2 | Clicks on a project |  |
|  | 3 |  | Shows M2 |
|  | 4 | Clicks on a issue |  |
|  | 5 |  | Shows M3 |
| Extension n.1 (User has no projects) |  |  |  |
| 1a |  | Shows M1a |
| Extension n.2 (Project has no issues) |  |  |  |
| 2b |  | Shows M2b |
| Extension n.3 (generic error) |  |  |  |
| 1, 3, 5 err |  | Shows MErr |