Universidad de Oviedo

Escuela de Ingeniería Informática
Universidad de Oviedo

UNIVERSIDAD DE OVIEDO
ESCUELA DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA DEL SOFTWARE

TRABAJO DE FIN DE GRADO

IR-Board

Requirements Management Platform

Author:
Javier Carrasco Arango
Tutors:
Jorge Álvarez Fidalgo
Benjamín López Pérez

2026-02-25

Declaration of originality

I, , with DNI and UO294532, hereby declare that this work is completely original and all sources used during the development of it have been correctly cited. In Avilés, Asturias, on the xx of xx of 2026,

Signed:

Abstract

Keywords

About the document

This document follows a template provided by Jorge Álvarez Fidalgo.

It has been adapted and modified to fit the specific needs of the project during its development.

Special thanks to

Index

Figure Index

Table Index

Chapter 1. Introduction

1.1 Object

The primary object of this project is the design and implementation of **IR-Board**, a comprehensive Requirements Management Platform (RMP) designed to support the full lifecycle of software requirements engineering. The system aims to bridge the gap between traditional documentation standards and modern agile methodologies, providing a centralized environment for eliciting, refining, and managing requirements.

Key objectives of the system include:

- **Methodological Compliance:** Implementing a framework that follows international standards for requirements specification, specifically the **IEEE 830** guidelines and the **ISO/IEC/IEEE 29148** standard.
- **Hybrid Documentation Support:** Providing tools for both traditional modeling (use cases, flowcharts, and decision tables) and Agile practices (user story mapping and management).
- **Relation-Based Access Control (ReBAC):** Developing a sophisticated security model where permissions are not merely role-based but determined by the dynamic relationship between users and specific entities (Projects and Functionalities), implemented through a **Zero-Trust** architecture.
- **Lifecycle and State Management:** Automating the management of requirement states (Pending Approval, Approved, Deactivated, etc.) and project lifecycles to ensure data integrity and traceability.
- **Collaborative Engineering:** Facilitating stakeholder management and real-time concurrency control to prevent data conflicts during collaborative editing sessions.
- **System Observability:** Integrating a high-standard monitoring stack to provide administrators with full visibility into the infrastructure's health and security audit logs.

1.2 Background

1.3 Current state

1.4 Definitions and Abbreviations

1.5 Scope

1.6 Assumptions and Constraints

Chapter 2. Theoretical Background

Chapter 3. Feasibility Study and Alternatives Analysis

Chapter 4. Initial Project Planning and Management

4.1 Stakeholder Identification

4.2 OBS and PBS

4.3 Initial Planning

4.4 Risk Analysis

4.5 Initial Budget

Chapter 5. System Analysis

5.1 Input Documentation

5.2 Users and Characteristics

In this project, instead of conventional system-wide roles, the approach I found fit best the nature of requirement management systems was a relation-based role system. Therefore, two sets of user permissions can be defined: system-level permissions and project-level permissions.

User	Description
Admin	Has access to project creation, deactivation, purge of removed projects... Still needs a project role to add to or modify a project, even if created by himself. Can link users as project manager to a project.
Basic	A basic user, the default. Has access to his profile management.

Table 1: List of permission levels on the system

A user is either an admin or not. Due to zero-trust, unless he is added as project manager to it he wont be able to modify any project, even if it was created by himself.

User	Description
ProjectManager	Access to add, edit and disable functionalities, requirements within those functionalities, documents, and can link users to that project.
RequirementEngineer	Linked to a functionality, has access to add, modify and disable requirements in that functionality, as well as with documents linked to the project.
Stakeholder	Linked to a functionality, has read-only access to the requirements of that functionality, and documents linked to the project.

Table 2: List of permissions within a project

These permissions overlap, in case a user is linked to a project in several ways. In that situation, due to this overlap, the most high level permission prevails.

For example, if a user is listed as requirement engineer and stakeholder in the same functionality, the user will be able to view (both permissions), and add, modify, disable and all other requirement engineer actions as well.

5.3 Requirements Analysis

Here are presented some diagrams that helped during the requirement edduction process.

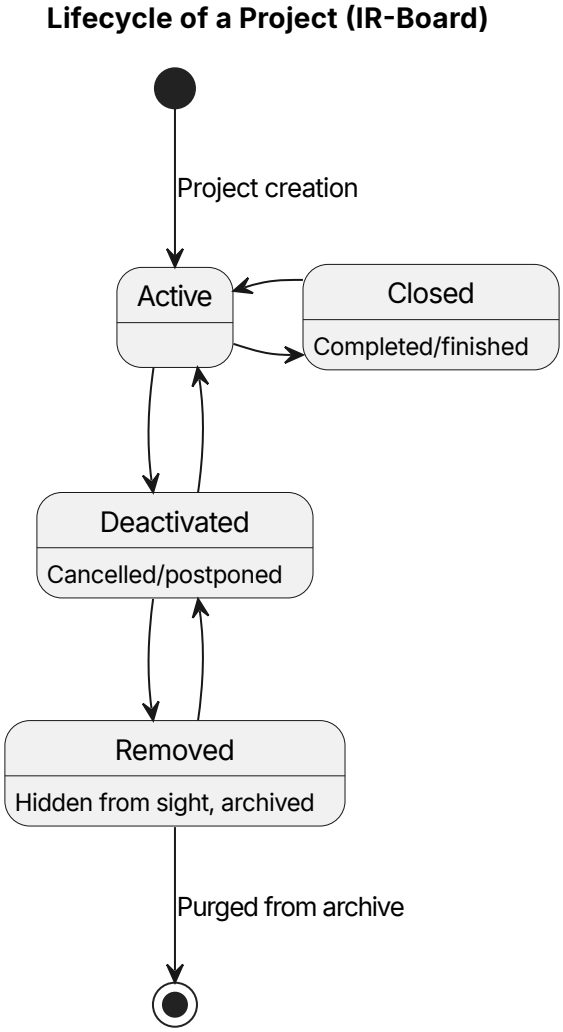

Figure 1: Lifecicle of a project entity

Active - A project entity that is currently in progress.

Closed - A project entity that has been finished. As the nature of a project is that one never ends, it does not represent an end state.

Deactivated - A project entity that has been cancelled, postponed etc; A project that is not finished but is not currently in use.

Removed - A project entity that has been archived for removal, placed in the trash bin in case it is needed as last resort.

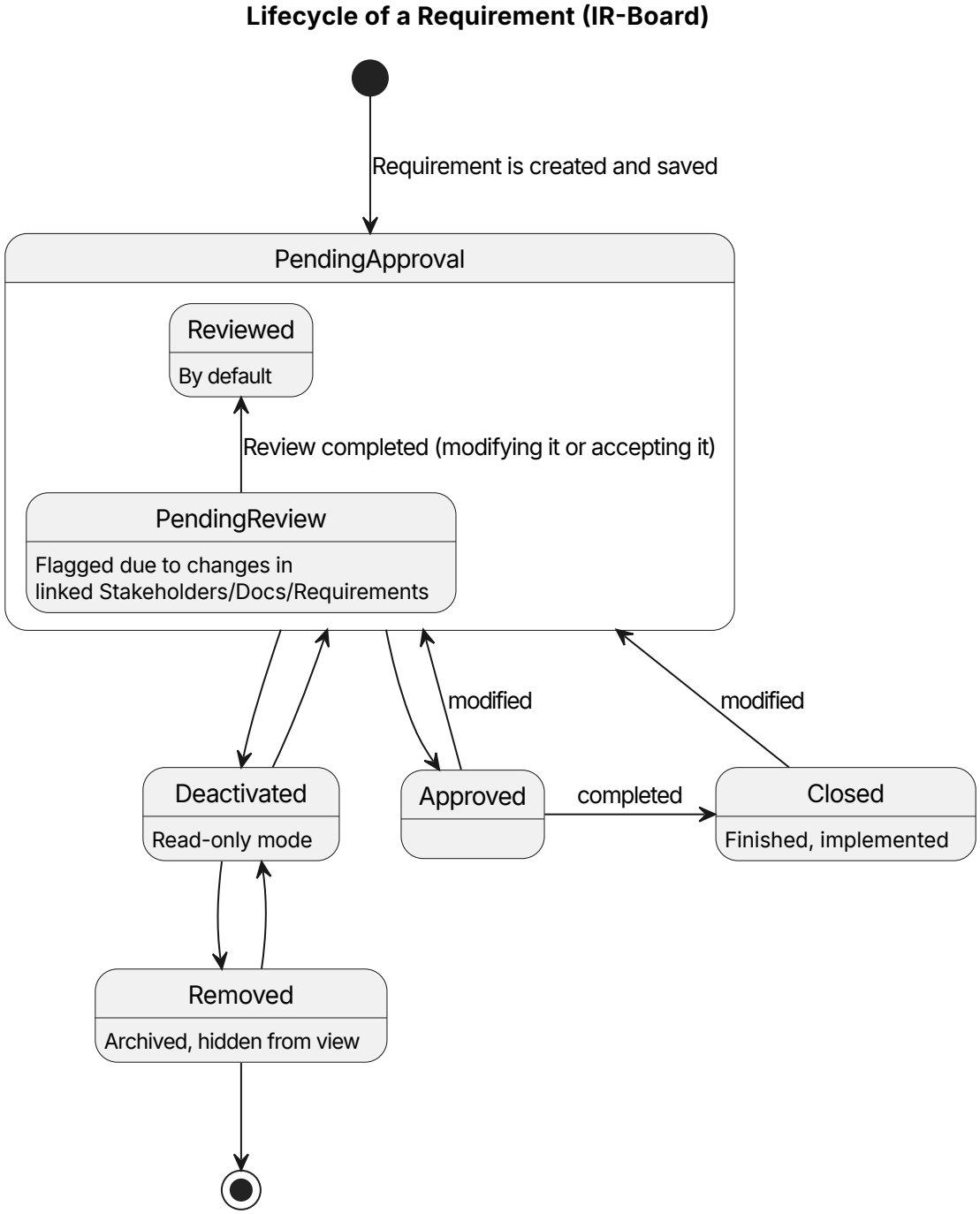

Figure 2: Requirement entity's state diagram

PendingApproval - A requirement entity that has not been validated by a stakeholder as it currently is. It's a complex state to ease development, as the pending review can be seen as an extension of itself, and therefore can be a simple boolean flag.

PendingReview - A requirement entity that needs attention and possible modification, due to a change on a linked entity. Expected to be purely a flag.

Approved - A requirement entity that has been validated by the appropiate stakeholders with the project manager outside the system.

Deactivated - A requirement entity has been deactivated for some reason, be cancelled or an error, and does not count towards the metrics of the project.

Removed - A requirement entity that has been deemed innecessary to the project. It is hidden from view, archived.

5.4 System Analysis

5.4.1 Class Analysis

5.4.2 Data Modeling

5.4.3 Process Modeling

5.4.4 User Interface Definition

5.5 Requirements Specification

5.5.1 External Interfaces

5.5.2 Functional Requirements

5.5.2.1 project management

- PM.1. The system must allow an admin to create a project
  - PM.1.1. The system must require a Project name to generate a project correctly
  - PM.1.2. The system must require a description to generate a project correctly
  - PM.1.3. The system must require a user to be project manager to generate a project correctly
  - PM.1.4. The system must allow to optionally change the priority style set for the project, either:
    - PM.1.4.1. Ternary (High, Medium, Low) Predefined
    - PM.1.4.2. MOSCOW (Must, Should, Could or Won't have)
- PM.2. The system must allow an admin to deactivate a project
  - PM.2.1. The system must ask for confirmation before deactivating
  - PM.2.2. The system must put the project on read only mode
- PM.3. The system must allow an admin to reactivate a deactivated project
- PM.4. The system must allow an admin to modify an active project
- PM.5. The system must allow to link users with a project
  - PM.5.1. The system must allow an admin to link users to a project as project manager
  - PM.5.2. The system must allow an admin or project manager linked to the project to link users to a functionality on said project as a stakeholder user
  - PM.5.3. The system must allow a project manager to link users to one or more functionalities of a project as requirement engineers.
- PM.6. The system must allow access to the project description/dashboard to users linked to it or a functionality of it.

PM.6.1. The system must show the total split of requirements by their states (pie chart)
PM.6.2. The system must not take into account deactivated requirements toward any metric
PM.6.3. The system must show the different functionalities of the project
PM.7. The system must allow a project manager to mark as approved all elements in a project
PM.8. The system must allow a project manager to add a functionality to a project
PM.8.1. A functionality needs a name and unique set of letters for its dynamic identifier.
PM.8.2. The system must automatically attempt to get the letters for the dynamic identifier from the name
PM.8.2.1. The system must take the first letter from every word in the name.
PM.8.2.2. If the identifier is already in use by another functionality on the same project, the system will suggest one letter more of each word on the name.
PM.8.2.3. If the system cannot generate a new set of letters to identify its requirements, a message must be shown to the project manager.
PM.8.3. The system must deny adding any functionality with an identifier matching another on the same project.
PM.8.4. The system must automatically link the project manager to the new functionality
PM.9. The system must allow a project manager to modify a functionality.
PM.10. The system must allow a project manager to deactivate a functionality.
PM.10.1. The system must ask for confirmation before deactivating
PM.10.2. The system must put the functionality (elements contained by it) on read only
PM.11. The system must allow a project manager to reactivate a functionality.
PM.12. The system must allow a project manager to mark as approved all elements in a functionality
PM.13. The system must allow a project manager to generate a baseline for a project.
PM.13.1. The system must perform a snapshot of the project once a baseline is set.
PM.14. The system must allow a project manager to export the project's requirements onto a pdf file

5.5.2.2 Stakeholders management

SM.1. The system must allow any user linked with the project access to view its stakeholders
SM.1.1. The system must show if a stakeholder is flagged as pending review
SM.1.2. The system must show the identifier, the name and part of the description
SM.1.3. The system must allow to collapse and expand stakeholders with children

SM.1.4. The system must allow to view the details of a stakeholder
SM.1.4.1. The system must show all atributes of a stakeholder
SM.1.4.2. The system must show all requirements linked to it
SM.2. The system must allow a project manager to add a new stakeholder to a project
SM.2.1. The system must only allow a stakeholder to be added to a project the user is linked to.
SM.2.2. The system must require a name to generate a stakeholder
SM.2.3. The system must require a description to generate a stakeholder
SM.2.4. The system must generate the identifier for the stakeholder
SM.3. The system must allow a project manager or requirement engineer to link a stakeholder to one or more requirements on the same project
SM.3.1. The system must only allow the user to link the stakeholder to a requirement on a functionality they are linked to
SM.4. The system must allow a project manager or requirement engineer to unlink a stakeholder from one or more requirements
SM.4.1. The system must only allow the user to unlink a stakeholder from a requirement of a functionality they are linked to.
SM.5. The system must allow a project manager to deactivate a stakeholder from a project the user is linked to
SM.5.1. The system must show the user the amount of entities affected by the deactivations
SM.5.2. The system must flag all entities linked as pending review
SM.5.3. The system must ask for confirmation before deactivating
SM.5.4. The system must put the stakeholder on read only mode
SM.6. The system must allow a project manager or requirement engineer to modify a stakeholder
SM.6.1. The system must flag as pending review linked entities upon saving with changes.

5.5.2.3 Requirement management

RM.1. The system must allow users linked to a project or functionality of a project access to its requirements
RM.1.1. The system must only allow users linked to the functionality of a functional requirement access to it
RM.1.2. The system must allow to collapse and expand requirements with children
RM.1.3. The system must show the dynamic identifier, the name, state and part of the description
RM.1.4. The system must allow to view the details of a requirement
RM.1.4.1. The system must show the internal unique identifier
RM.1.4.2. The system must show all atributes of a requirement
RM.1.4.3. The system must show all stakeholders linked to it
RM.1.4.4. The system must show all requirements cross-linked with it
RM.1.4.5. The system must show all documents linked to it
RM.1.4.6. The system must show the previous dynamic identifiers

- RM.2. The system must allow a requirement engineer or a project manager to add a requirement to a project the user is linked to
  - RM.2.1. The system must only allow a functional requirement to be added to a functionality the user is linked to.
  - RM.2.2. The system must allow the user to generate a requirement as a child of another requirement (nesting).
  - RM.2.3. The system must assign automatically the dynamic identifier
    - RM.2.3.1. The identifier must be based on its relation to other requirements.
    - RM.2.3.2. The identifier must represent if it is a functional or non functional requirement (FR or NFR)
    - RM.2.3.3. The identifier must represent the folder/component that holds the requirement (user management → UM)
  - RM.2.4. The system must assign automatically the internal unique identifier
    - RM.2.4.1. The identifier must represent the project that will hold the requirement
    - RM.2.4.2. The identifier must represent whether the requirement is functional or non functional
    - RM.2.4.3. The identifier must have a random element to ensure a low colision rate
  - RM.2.5. The system must ask for the following data for a functional requirement:
    - RM.2.5.1. The system must require a name
    - RM.2.5.2. The system must require a description
    - RM.2.5.3. The system must require a priority following the one set on the project creation
    - RM.2.5.4. The system must allow to set a stability, which is optional
    - RM.2.5.5. The system must allow to set a stakeholder as origin, which is optional
  - RM.2.6. The system must ask for the following data for a non-functional requirement:
    - RM.2.6.1. The system must require a name
    - RM.2.6.2. The system must require a description
    - RM.2.6.3. The system must allow to set a measurement unit
    - RM.2.6.4. The system must allow to set a comparison operator
      - RM.2.6.4.1. equal to, less than or greater than
    - RM.2.6.5. The system must allow to set a threshold value
      - RM.2.6.5.1. This value represents the minimum value to mark the requirement as passed
    - RM.2.6.6. The system must allow to set a target value
      - RM.2.6.6.1. This value represents the optimal value desired by the team
    - RM.2.6.7. The system must allow to set an actual value
      - RM.2.6.7.1. This value represents the current status of the measurement

RM.2.7. The system must automatically set the new requirement as pending approval

RM.3. The system must allow a project manager or requirement engineer to link a requirement on a functionality they are linked to, to another entity

RM.3.1. The system must allow to link a requirement with a stakeholder of the same project

RM.3.2. The system must allow to un-link a requirement with a stakeholder

RM.3.3. The system must allow to link a requirement with one or more requirements of functionalities of the same project the user is linked to

RM.3.4. The system must allow to un-link a requirement with other requirements of functionalities of the same project the user is linked to

RM.3.5. The system must allow to link a requirement with one or more documents of the same project.

RM.3.6. The system must allow to un-link a requirement with one or more documents of the same project.

RM.4. The system must allow a requirement engineer or a project manager to deactivate a requirement pending approval on a functionality they are linked to

RM.4.1. The system must show the user the amount of entities that will be affected by the deactivation

RM.4.2. The system must ask for confirmation

RM.4.3. The system must flag any requirements linked to the deactivated requirement as pending review

RM.4.4. The system must put the requirement on read only

RM.5. The system must allow a project manager or requirement engineer to set a deactivated requirement as removed

RM.5.1. The system must hide from view a removed requirement, effectively archiving it

RM.6. The system must allow a requirement engineer or a project manager to reactivate a requirement on a functionality they are linked to

RM.6.1. The system must automatically flag as pending review the reactivated requirement

RM.7. The system must allow a requirement engineer or a project manager to modify a requirement on a project

RM.7.1. The system must only allow a project manager or requirement engineer to modify functional requiremets of a functionality the user is linked to.

RM.7.2. The system must flag linked requirements as pending review upon saving with changes.

RM.8. The system must allow a project manager to mark as approved one or more requirements

RM.8.1. The system must only allow to mark as approved a requirement that is pending approval, not pending review nor deactivated.

RM.9. The system must allow a project manager to mark as closed a requirement

RM.9.1. The system must set the requirement as pending approval if it is modified.

RM.10. The system must allow a project manager or requirement engineer linked to a functionality of the project to change the position of a requirement

RM.10.1. The system must allow reordering of functional requirements to users linked to the same functionality.

RM.10.2. The system must update the dynamic identifier automatically

RM.10.3. The system must set the order of requirements using a floating point order value

RM.11. The system must allow a project manager or requirement engineer to review an requirement flagged as pending a review

RM.11.1. The system must allow removing the flag if no changes are required.

RM.11.2. The system must allow modifying the requirement upon review.

RM.11.2.1. The system must remove the flag upon saving with changes.

RM.11.2.2. The system must flag the linked entities as pending a review.

5.5.2.4 User management

UM.1. The system must allow an admin to invite new users to the system

UM.1.1. The system must provide different levels of authorisation.

UM.1.1.1. The system must have the levels: Admin, project manager, requirement engineer and stakeholder user

UM.1.2. The system must ask the admin to set the name, surname, and email of the invited user

UM.1.2.1. The system must generate an signup code as a temporal password

UM.1.2.2. The system must automatically send an invitation with the signup code to the email of the invited user

UM.2. The system must allow an admin to view the name and surname of a user from the system

UM.3. The system must allow an admin to modify the name and surname of a user from the system

UM.4. The system must allow an admin to view the current permissions of a user from the system

UM.5. The system must allow an admin to generate a new invite with a signup code for a user

UM.6. The system must allow any user with valid credentials to sign in to the system

UM.6.1. The system must prompt any user signing in with a signup code to set a permanent password.

UM.6.1.1. The system must ensure the password is between 15 and 64 characters long.

UM.6.1.2. The system must make use of a random salt specific of each user.

UM.6.1.3. The system must remove any password or signup code of the user upon seting a permanent password.

UM.6.2. The system must temporally block the user after 3 consecutive failed attempts
UM.7. The system must allow an admin to deactivate a user from the system
UM.7.1. A deactivated user remains on the system but cannot access it
UM.8. The system must allow an admin to reactivate a user from the system

5.5.2.5 Document management and modelling

DMM.1. The system must allow users linked to a project access to documents of that project
DMM.1.1. The system must show entities linked to the document.
DMM.2. The system must allow a project manager or a requirement engineer to add document to a project
DMM.2.1. The user must be linked to the project
DMM.3. The system must allow a document to be linked to one or more requirements of the same project
DMM.3.1. The system must flag those requirements linked to it as pending a review if the document is altered
DMM.4. The system must allow a project manager or requirement engineer to update a document
DMM.4.1. The user must be linked to the project the document is on.
DMM.4.2. The system must flag as pending a review any requirements linked to the document
DMM.5. The system must allow a project manager to disable a document
DMM.5.1. The system must flag as pending a review any requirements linked to the document
DMM.6. The system must allow a requirement engineer to model diagrams using a Draw.io integration

5.5.2.6 Concurrency

C.1. The system must block other users from modifying an entity that another user is already modifying
C.1.1. The system must release automatically the entity if the user modifying it saves and exits (stops modifying).
C.1.2. The system must release automatically the entity after a predetermined timeout period
C.1.3. The system must release automatically the entity if the user editing it modifies another entity
C.1.4. The system must only accept changes to the entity from the user who holds the entity
C.2. The system must display for other users who is modifying the entity

5.5.2.7 Search and filtering

SF.1. The system must allow searching an entity by internal unique identifier.
SF.1.1. The system must search lexically
SF.1.2. The system must allow the user to see the details of the found entity
SF.1.2.1. only if an exact match occurs,
SF.1.2.2. only if the user has access to it.

SF.2. The system must allow users to filter entities they have access to
  SF.2.1. The system must allow filtering out deactivated requirements
  SF.2.2. The system must allow filtering requirements based on priority
  SF.2.3. The system must allow filtering requirements based on state
  SF.2.4. Any filter must be reversible; ascending or descending order

5.5.2.8 Observability

O.1. The system must allow an admin to acess an observability dashboard with information logs
O.2. The system must allow an admin to monitor the health and performance metrics of the microservices infrastructure
O.3. The system must allow an admin to query and visualize logs from all system components

5.5.3 Usability Requirements

5.5.4 Performance Requirements

5.5.5 Logical Database Requirements

5.5.6 Design Constraints

5.5.7 System Attributes

5.5.8 Supporting Information

5.6 Test Plan Analysis

To ensure the reliability, maintainability, and performance of the IR-Board system, a multi-dimensional testing strategy has been defined. This plan covers the entire development lifecycle, from code quality to system behavior under stress.

5.6.1 Code maintainability and unit testing with SonarQube

The project utilizes SonarQube as a Static Application Security Testing (SAST) tool. This analysis is integrated into the development workflow to ensure the following:

- Maintainability: Identification of “code smells” and technical debt that could hinder future scalability.
- Code Coverage Validation: Monitoring the percentage of the source code executed during automated tests. This ensures that critical business logic is thoroughly verified, maintaining a high safety net against regressions and establishing a minimum threshold of tested code before deployment.
- Reliability: Detection of potential bugs and logic errors through automated pattern matching.
- Security: Scanning for common vulnerabilities and ensuring compliance with industry standards (e.g., OWASP Top 10).

5.6.2 Load testing

For the load testing phase, the primary focus will be on stressing the critical entry points of the system, specifically the traffic flow passing through Traefik and Oathkeeper toward the Spring Boot backend. The goal is to simulate bursts of concurrent users to identify the exact point where identity validation latency begins to degrade

the user experience or if Kratos' session management can handle the expected volume. This process goes beyond checking for server crashes; it involves using the Grafana stack to monitor how container resources scale and ensuring the internal network routing maintains stability under heavy pressure.

5.6.3 Usability testing

Regarding usability testing, the plan involves observing real users interacting with the React interface to validate that the integration of Grafana dashboards feels intuitive and seamless. Special attention will be paid to how easily users can navigate between Loki logs and the core business logic, ensuring that the underlying complexity of the microservices architecture remains completely transparent to the end user. The ultimate objective is to confirm that authentication flows do not create unnecessary friction and that the frontend information hierarchy allows for efficient data management without requiring prior technical knowledge from the operator.

Chapter 6. System Design

6.1 System Architecture

The system architecture follows a Microservices approach based on the Zero Trust security model. This ensures flexibility and scalability while maintaining a high level of isolation between business logic and infrastructure concerns. To guarantee a professional security standard while maintaining a manageable project scope, core identity and access management responsibilities have been delegated to the Ory Open Source ecosystem.

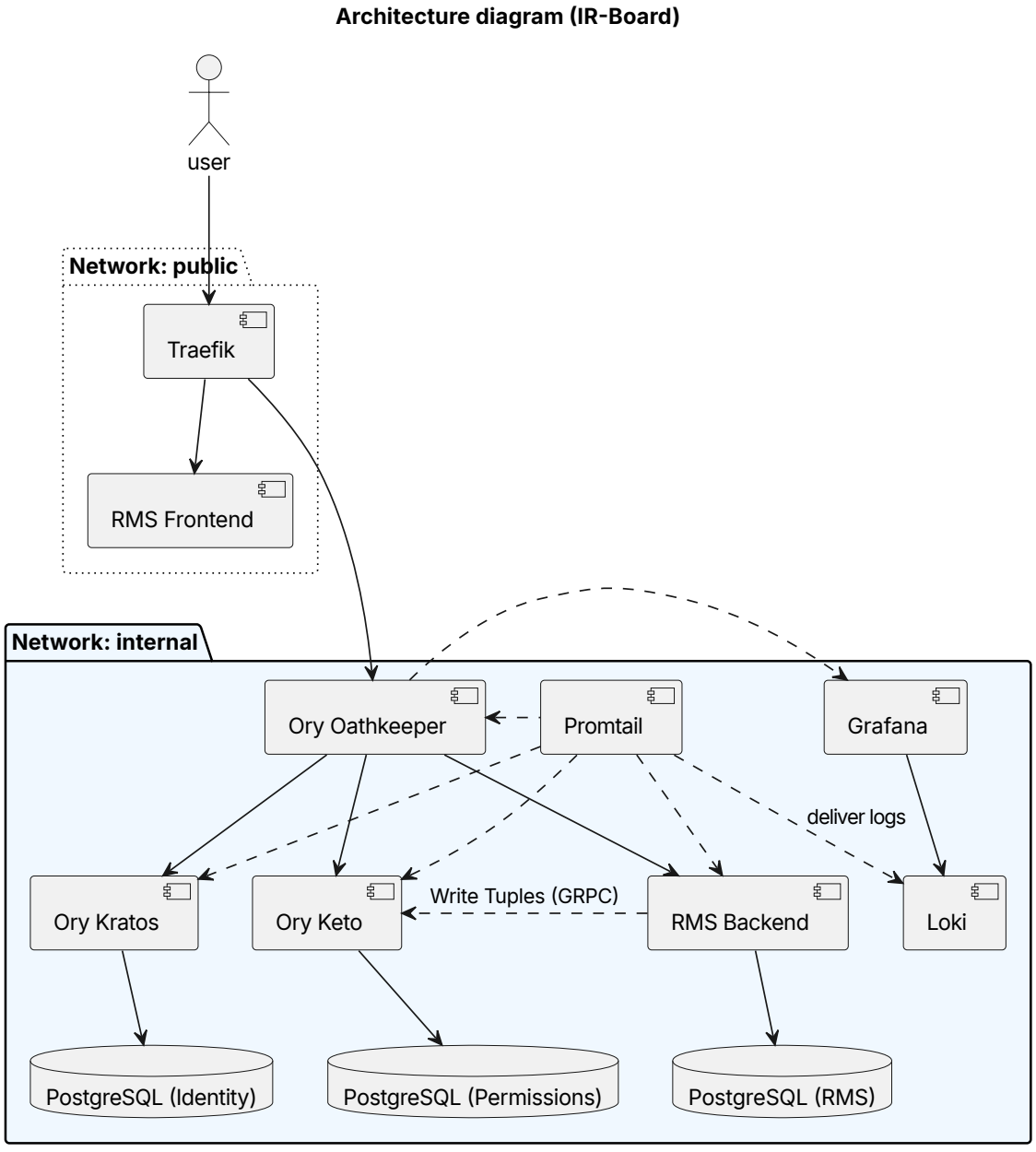

Figure 3: Architecture C2 component diagram

Traefik - Acts as the system's entry point and TLS Termination Proxy. It handles dynamic routing and load balancing, effectively hiding the internal network topology and eliminating the need to expose multiple ports to the public internet.

RMS Frontend - Built with React and TypeScript, served as static content. It executes within the user's browser and communicates with the backend services through the API Gateway.

Ory Oathkeeper - A policy-enforcement engine that acts as a gatekeeper between the public and internal networks. It intercepted every request to validate session integrity (via Kratos) and fine-grained permissions (via Keto) before allowing traffic to reach the internal services.

Ory Keto - A relationship-based access control (ReBAC) server inspired by Google's Zanzibar. It manages permission tuples, allowing the system to verify complex authorization rules (e.g., checking if a user is linked to a specific project).

Ory Kratos - Manages the full identity lifecycle, including user registration, multi-factor authentication, and session management, ensuring that sensitive credentials are handled by a specialized security component.

RMS Backend - The core service developed using Spring Boot, containing the domain-specific business logic and data persistence.

Promtail - An agent that ships local logs from the various microservices to the central store. and sends them to Loki.

Loki - A horizontally scalable, highly available log aggregation system.

Grafana - A visualization platform used to build observability dashboards. In this architecture, it is placed within the internal network and accessed through the Identity Proxy to ensure that system logs are only visible to authorized personnel.

6.2 Real Use Case Design

6.3 Class Design

6.4 Database Design

6.5 User Interface Design

6.6 Test Plan Specification

Chapter 7. System Implementation

Chapter 8. Test Plan Execution

8.1 Unit Testing

8.2 Integration and System Testing

8.3 Usability Testing

8.4 Accessibility Testing

8.5 Load Testing

8.6 Acceptance Testing

Chapter 9. System manuals

9.1 Installation Guide

9.2 User Manual

9.3 Developer Guide

Chapter 10. Project Closure

10.1 Final Schedule

10.2 Final Risk Report

10.3 Final Budget

10.4 Project Closure Analysis

Chapter 11. Conclusions and Future Work

Chapter 12. References

Chapter 13. Appendices

13.1 Supplementary Material