Aveiro University

Department of Electronics, Telecommunications and Informatics

Rework of the information system of DETI Maker Lab

1st cycle degree in Computer and Informatics Engineering

Project Members:
André Silva
Manuel Mendonça
João Martins
Jakub Suliga
Laura Gabryjańczyk

Supervisor:
Prof. Diogo Gomes

March 2026

Abstract

The DETI Maker Lab serves as an open workspace for students, faculty, and staff to develop electronics, telecommunications, and informatics projects. Currently, the lab's operations are heavily based on a Wiki-based platform, which has proven to be cumbersome for managing equipment requests. Furthermore, the use of Markdown for project creation leads to inconsistent data entry, and tracking borrowed equipment is particularly difficult for the lab technician, who is not an IT specialist.

This report details the ongoing development of a modernized, self-managed information system designed to resolve these inefficiencies. Although the project initially aimed to rework and extend the existing platform, implementation analysis showed that meeting the defined goals would require rebuilding most core components from scratch. The new solution preserves useful domain knowledge from the previous system while integrating inventory management with Snipe-IT, an open-source asset management tool, and introducing key improvements such as form-based project creation, a streamlined requisition workflow (request, approve, assign, return), and integration with the university's universal user authentication system.

Additionally, this project encompasses the enhancement of existing mobile applications for remote requisitions, being the unfinished iOS side and the Android application renewal. By prioritizing intuitive user guidance and reusing existing open-source resources, this new iteration aims to significantly improve usage rates and simplify lab management for all stakeholders.

Chapter 1

Introduction

The DETI Maker Lab is intended to be an open and collaborative space where students, teachers, and staff can carry out projects in various informatic fields. The laboratory is project-oriented: users are organized around projects, and each project may require equipment, components, workspaces, and other resources throughout its lifecycle. This means that the information system supporting the laboratory is not a secondary tool, but a core operational element — responsible for project registration, requisition control, and documentation.

At present, the Maker Lab relies on a Wiki-based platform that already supports project management, equipment information, requisitions, and documentation. This installed base is valuable because of the time already invested in it and the operational knowledge it contains. However, during development we observed that many of its limitations are structural rather than isolated issues: the requisition process is cumbersome for students and project groups, equipment tracking is difficult and error-prone, and project creation through Markdown leads to inconsistent data that is ill-suited for structured storage or later querying.

Although the initial vision was to rework the existing information system, implementation work showed that meeting the project goals would effectively require rebuilding almost everything from scratch. The current approach therefore preserves the useful domain knowledge and practices from the previous platform while introducing a new implementation centered on four directions: integrating inventory management with Snipe-IT, simplifying the requisition workflow, replacing Markdown-based project creation with structured forms, and unifying projects, groups, users, requisitions, and lent equipment within one coherent system.

A second important motivation is usability. The laboratory technician must be able to manage inventory and requisitions without depending on tools that are overly technical or difficult to operate. In parallel, student groups need a clearer and faster process to request, track, and return equipment across the multiple iterations that naturally occur over a project's lifetime. This emphasis on practical usability strongly influences both the requirements and the architecture proposed in this report.

This document corresponds to the elaboration phase of the project. Following the guidance for Milestone 2, it consolidates the project context, related work, tools and technologies, requirements elicitation results, and the current architectural baseline. It also defines the expected outcomes that will guide implementation in the subsequent milestones.

1.1 Main Objectives

- Integrate the Maker Lab system with Snipe-IT, moving inventory authority away from the current Wiki-based equipment management.
- Make it a straightforward and simple process for the technician to manage inventory and requisitions
- Enable a simple and repeatable requisition workflow that project groups can use throughout the entire project lifecycle.
- Replace Markdown project creation with form-based input to ensure consistent, database-ready records.
- Unify projects, groups, students, requisitions, and lent equipment within a single coherent information system.
- Integrate authentication with the university's universal identity provider across all system components.
- Complete the unfinished iOS application and renew the Android application to support the updated requisition and project management workflows.

Chapter 2

Related Work, Tools and Technologies

This chapter presents the existing work that directly influenced the project, the external tools and platforms adopted, and the technologies selected for development. The guiding principle throughout was to avoid reinventing the wheel: where a mature, well-supported solution already exists, the project adopts it rather than building a custom alternative.

2.1 Related Work

The most directly relevant piece of prior work is the existing Maker Lab platform itself. Built on a Wiki-based solution, it already supports project registration, equipment information, requisitions, and documentation, and it serves as the operational backbone of the laboratory. Rather than discarding this foundation, the current project treats it as the starting point, identifying its shortcomings — cumbersome requisition flows, inconsistent Markdown-based project data, and limited inventory control — and redesigning around them.

A second relevant piece of prior work is the mobile ecosystem developed in the previous project iteration. Two applications were produced, one for Android and one for iOS, though only the Android side reached a completed state. The current project inherits this partial implementation and incorporates mobile support as a first-class concern, ensuring that students can request and track equipment through both web and mobile interfaces.

No directly comparable academic systems were identified in the literature for small-scale university laboratory management. However, the broader field of asset management systems and inventory control platforms informed the decision to integrate an existing open-source solution rather than develop one from scratch, as discussed in the following section.

2.2 External Tools and Platforms

2.2.1 Snipe-IT

Snipe-IT is an open-source asset management platform designed specifically for tracking physical assets, consumables, and accessories within an organisation. It provides a web-

based interface for managing stock levels, assigning assets to users, recording check-in and check-out events, and generating audit trails. Its adoption in this project was recommended by the project supervisor, who identified it as a mature and well-supported solution aligned with the laboratory's inventory needs.

The key architectural decision is to use Snipe-IT as the *authoritative* system for all physical inventory — devices, components, and equipment — while the Maker Lab application layer retains responsibility for projects, groups, users, requisitions, and their history. This separation of concerns avoids duplicating inventory logic and allows the laboratory technician to manage stock through a tool purpose-built for that task, rather than through ad hoc entries in the Wiki.

2.2.2 Wiki Platform

The existing Wiki platform is retained as part of the system where it continues to add value, particularly for documentation and informational content. The project does not aim to migrate or replace it entirely, but rather to offload the workflows it handles poorly — namely structured project creation and equipment requisitions — to more appropriate components.

2.3 Development Technologies

2.3.1 Frontend

The web frontend is built with **Next.js**, a React-based framework that supports both server-side rendering and static generation. Next.js was chosen for its strong ecosystem, its ability to handle routing and server-side logic within the same project, and its widespread adoption in modern web development. Using React as the underlying library also ensures access to a large component ecosystem and simplifies the development of dynamic, interactive interfaces such as requisition forms and project dashboards.

2.3.2 Backend

The backend is developed using **FastAPI**, a modern Python web framework designed for building APIs with high performance and automatic documentation generation. FastAPI's use of Python type hints enables request validation and OpenAPI specification generation with minimal boilerplate, which is particularly useful for a project that must expose a well-defined API consumed by both web and mobile clients.

Database interaction is handled through **SQLModel**, which integrates naturally with FastAPI by combining SQLAlchemy's query capabilities with Pydantic's data validation. This allows the same model definitions to be used for both database schema and API serialisation, reducing duplication and keeping the codebase consistent.

2.3.3 Database

The project uses **PostgreSQL** as the relational database for the Maker Lab application layer. This database stores all entities that are not managed by Snipe-IT: users, projects, groups, requisitions, and the history of lent equipment. PostgreSQL was chosen for its

robustness, its strong support for relational integrity constraints, and its compatibility with SQLModel and the broader Python ecosystem.

It is important to note that this database is entirely separate from Snipe-IT's internal database. The two systems communicate through Snipe-IT's REST API, meaning the Maker Lab backend queries Snipe-IT for inventory data rather than accessing its database directly. This keeps the integration clean and ensures that future updates to Snipe-IT do not break the application layer.

2.3.4 Mobile Applications

The mobile applications are developed using **React Native**, an open-source framework maintained by Meta that enables the development of natively compiled applications for both Android and iOS from a single shared JavaScript codebase. React Native renders using each platform's own native components, meaning the applications behave like native apps while sharing the vast majority of their logic and interface code.

React Native was chosen primarily because the team is already working with React on the web frontend through Next.js, meaning no additional language or paradigm shift is required. This overlap in technology reduces friction and makes it more feasible for a small team to maintain both platforms in parallel — which is particularly relevant given that the iOS application from the previous project iteration was left incomplete and must be brought to a finished state alongside the Android renewal.

2.3.5 Collaboration and Project Management

The team's collaboration toolset was chosen to be lightweight and familiar. **GitHub** is used for source code management, pull request reviews, and version control across all repositories. **GitHub Projects** provides a Kanban-style board for task tracking and milestone planning. **Google Docs** serves as the shared team notebook for meeting notes, drafts, and shared documents. **Slack** and **WhatsApp** are used for day-to-day communication, with Slack handling more structured team discussions and WhatsApp serving for quick coordination.

2.4 Development Methodology

The team follows the **OpenUP** (Open Unified Process) methodology. OpenUP is a lean, iterative software development process that scales down the Rational Unified Process for use by small teams. It organises work into four phases — Inception, Elaboration, Construction, and Transition — each with defined goals and deliverables.

OpenUP was selected over alternatives such as Scrum or Kanban for several reasons. First, its phase structure maps naturally onto the course milestone structure, where each milestone corresponds to a different stage of maturity in the project. Second, unlike Scrum, OpenUP does not require fixed-length sprints or dedicated roles such as a Scrum Master, which suits a small academic team where all members share responsibilities. Third, OpenUP places explicit emphasis on architectural decisions and risk mitigation during the elaboration phase, which is directly aligned with the goals of this milestone: establishing a stable architectural baseline before committing to full implementation.

Chapter 3

Requirements Elicitation and Architecture

3.1 Requirements Gathering

The requirements presented in this report were gathered from several complementary sources. First, the original project brief defines the application context, the need to preserve the previous system, the importance of integrating inventory management with Snipe-IT, the intention to improve mobile requisitions, and the principle of reusing existing solutions whenever possible. Second, the work carried out during Milestone 1 included a direct meeting with the laboratory technician, who is one of the primary users of the system. This meeting was aimed at understanding how he currently uses the platform, what his day-to-day pain points are, and what improvements he would most value in a reworked solution. His input proved particularly relevant for the requisition workflow and inventory management requirements, as he provided first-hand insight into where the current system falls short in practice. Third, the elaboration work for Milestone 2 refined these points further through analysis of the current workflow, discussion of stakeholder roles, user stories, and simulation of the intended requisition process.

This combination of sources is consistent with the elaboration-phase guidance. The purpose at this stage is not only to list features, but also to understand how the system will actually be used, what constraints already exist, and which architectural decisions are needed early in order to keep the project viable. For this reason, requirements were organised around users, workflow, data consistency, inventory integration, and traceability.

3.2 Actors and Personas

Actor	Role in the system
Students / project groups	Create or join projects, request equipment, track requisition status, and return borrowed items.
Lab technician	Manage inventory, approve or reject requisitions, assign equipment, and receive returned items.
Supervisor	Review project progress and maintain visibility over how the system supports ongoing work.

Table 3.1: System actors and their roles.

3.3 Functional Requirements

Access and Roles

- The system must authenticate users through the university Single Sign-On (SSO) service, ensuring that only members of the university community can access the platform.
- The system must distinguish between three primary roles: student, lab technician, and supervisor. Students may create projects, join groups, and submit requisitions. The lab technician may manage inventory, process requisitions, and assign or receive equipment. Supervisors may consult active projects and associated requisitions but may not intervene directly in the requisition workflow.
- All actions must be restricted according to the authenticated user's role, preventing users from performing operations outside their designated responsibilities.

Project and Group Management

- The system must allow users to create new projects using a structured form, replacing the previous Markdown-based approach in order to ensure consistent and database-ready records.
- The system shall allow a lab technician or supervisor to review newly submitted projects and approve or reject them before they become active.
- Users must be able to join an existing project and be associated with its group, so that requisitions and equipment assignments can be correctly linked to both the project and its members.
- Each requisition must be traceable to a specific project and group, ensuring that equipment usage is always contextualised within an ongoing piece of work.

Requisition Workflow

- The system must allow project groups to submit equipment requisitions at any point during the project lifecycle, supporting multiple requisitions per project as needs evolve.

- Each requisition must follow a defined workflow with four stages: request, approval or rejection by the lab technician, equipment assignment, and return. The system must enforce this sequence and prevent stages from being skipped.
- Students must be able to view the current status of any requisition they have submitted, so that they are aware of whether it is pending, approved, rejected, assigned, or closed.
- The lab technician must be able to approve or reject incoming requests, assign specific items to approved requisitions, and register the return of equipment when it is brought back.

Inventory Integration

- The system must integrate with Snipe-IT as the authoritative source for all inventory data, including equipment availability, stock levels, and asset details. No duplicate inventory management logic should exist within the Maker Lab application layer.
- Equipment availability shown to students during requisition must reflect the current state of Snipe-IT in order to prevent requests for items that are already assigned or unavailable.

Traceability and History

- The system must store a complete history of all requisitions, assignments, and returns, associated with the relevant project, group, and user.
- The system must support queries over this historical data, including equipment currently on loan per project, all active requisitions, and items whose return deadline has passed.

3.4 Non-Functional Requirements, Assumptions and Dependencies

- **Security:** The system must protect user data and enforce controlled access according to user role.
- **Integrity:** Project, requisition, and assignment data must remain consistent across the application and the inventory layer.
- **Maintainability:** The architecture should remain understandable and modular enough to support further development and integration.
- **Response time:** Core requisition operations should have acceptable response times for normal laboratory usage.

Assumptions and Dependencies

- University SSO integration will be available for authentication.
- Snipe-IT will be treated as the authoritative inventory system.
- The existing Maker Lab database must be fully imported to our system.

3.5 Use-Case Core and Representative User Stories

Core Requisition Workflow

1. Authenticate with the university account.
2. Submit to create a project or join one.
3. Approve or reject the project.
4. View available equipment.
5. Submit a requisition.
6. Approve or reject the requisition.
7. Assign equipment.
8. Track requisition status.
9. Return equipment.

Representative User Stories

- **Join a project group:** Ana wants to create a project using a form so that her group can organise the work and request equipment.
- **Request equipment:** João wants to request equipment for the project so that his group can build and test the prototype.
- **View available equipment:** A student wants to see the list of available equipment so that the group can plan around the resources offered by the lab.
- **Track request status:** Filipe wants to track the status of his requisitions so that he knows whether they were approved or rejected.
- **Return equipment:** Bernardo wants to mark equipment as returned so that other groups can use it again.
- **Manage inventory:** The lab technician wants to approve requests, assign items, and eventually add or remove equipment from the inventory.
- **Supervise equipment:** Professor Silva wants to consult the list of his active projects and their associated requisitions so that he can monitor whether his students are making appropriate use of the laboratory resources.

3.6 System Architecture

The architectural baseline defined during the elaboration phase reflects the need to separate concerns clearly while keeping the system cohesive and integrated. Users access the platform through two main client types: a web browser on desktop or laptop computers, and mobile applications on Android and iOS devices. Both client types interact with the application through secure HTTP-based requests, exposing a consistent API surface regardless of the access method.

Incoming requests pass through a reverse proxy deployed within the university network. This proxy acts as the single entry point into the system, handling request routing and providing a controlled boundary between external clients and internal services. Behind it, the application server hosts the core Maker Lab logic, encompassing project and group management, requisition handling, business rules, and coordination with the external services the platform depends on.

The data responsibilities are intentionally divided between two separate systems. The Maker Lab database, built on PostgreSQL, stores all application-domain data: users, projects, groups, requisitions, and their full history. Inventory authority is delegated entirely to Snipe-IT, which acts as the single source of truth for physical assets, stock levels, and equipment availability. The Maker Lab backend communicates with Snipe-IT exclusively through its REST API, meaning the two systems remain loosely coupled — updates to Snipe-IT's internals do not affect the application layer, and inventory logic does not need to be reimplemented within the project. This separation is one of the most consequential architectural decisions of the project, as it allows the team to focus development effort on workflow, traceability, and usability rather than asset management infrastructure.

Authentication is fully delegated to the university's Single Sign-On service. No credentials are stored or managed within the Maker Lab system itself, which simplifies security and ensures that user access remains consistent with the university's own identity management policies.

The architecture also accounts for the transition from the legacy Wiki platform. A one-time migration module will be used during the initial deployment to import existing data — projects, equipment records, and relevant historical content — from the Wiki into the new PostgreSQL database and Snipe-IT. Once this migration is complete, the module will no longer be needed, and the legacy Wiki will be gradually superseded by the new system. This approach preserves the value of the installed base without permanently coupling the new architecture to the old one.

Component	Responsibility
Web and mobile clients	User entry points for students, technicians, and supervisors.
Reverse proxy	Routes incoming requests to the application server inside the university network.
Application server	Implements project, group, requisition, and integration logic.
Maker Lab database	Stores structured project, group, user, requisition, and history data.
Snipe-IT	Acts as the single authoritative source for physical inventory and asset management.
University authentication service	Provides single sign-on for all user authentication.
Migration module	Used once during initial deployment to import legacy Wiki data into the new system.

Table 3.2: System components and their responsibilities.

This architecture supports the elaboration-phase objective of demonstrating an executable baseline. The most critical communication paths — client to application, application to database, application to Snipe-IT, and application to authentication service — must be validated early and working together before full implementation begins. Establishing these paths reduces the principal architectural risks of the project and gives the team confidence to proceed with feature development in subsequent milestones.

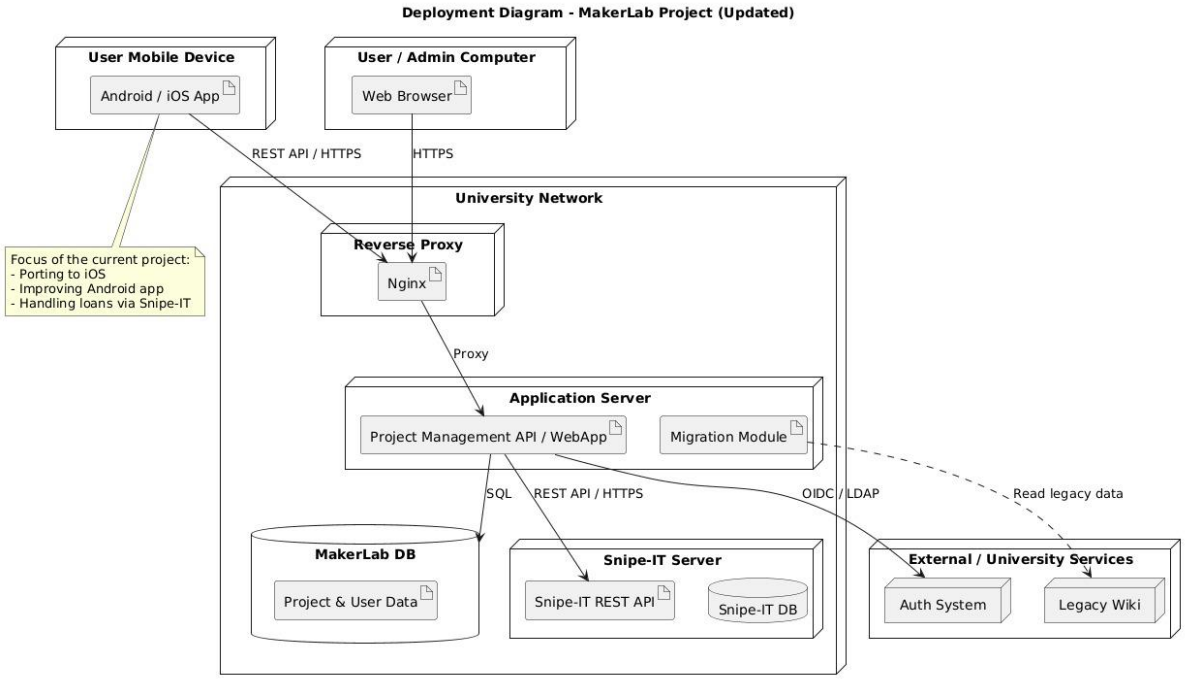

Figure 3.1: Deployment diagram of the proposed Maker Lab architecture.

3.7 Architecture Baseline and Current Achievements

The work completed to date already provides concrete evidence of an architectural baseline rather than a purely theoretical design. The most significant achievements at this stage are the successful integration of the core communication paths identified in the architecture: the frontend can communicate with the backend, the backend connects to the PostgreSQL database, and initial Snipe-IT integration work has begun. These are not trivial milestones — they confirm that the fundamental components of the system can work together, which is precisely the goal the elaboration phase sets out to achieve.

Beyond integration, a number of supporting project elements are also in place. The repository structure has been established and is actively used for version-controlled development and code reviews. The project website is live and the development blog is updated weekly, providing a transparent record of progress. Mock-ups have been produced for the main user-facing flows, including project creation and the requisition workflow, and user stories have been evaluated against the requirements to verify coverage.

A virtual machine environment has been provisioned within the university infrastructure to serve as the deployment target, and Snipe-IT has been installed and configured within it. This means the inventory system is already operational in the target environment, and integration work can proceed against a real instance rather than a simulated one.

Taken together, these achievements demonstrate that the project has moved past the stage of planning and design and into active, integration-oriented development. The skeleton of the system exists, the main risks related to component compatibility have been addressed, and the team is in a position to proceed with full feature implementation during the construction phase.

Chapter 4

Expected Results

The expected outcome of this project is a working information system in which projects and groups become the natural entry point for all equipment management throughout the lifecycle of laboratory work. Rather than relying on a requisition process that is detached from its project context, student groups should be able to request equipment repeatedly, track their requests transparently, and return items in a way that keeps records consistent and up to date.

At the process level, the most important expected result is a complete and reusable requisition workflow covering four clearly defined stages: request, approval or rejection, assignment, and return. Reusability is a key property here — a group should be able to initiate multiple requisitions at different points during the same project without generating fragmented or inconsistent records. The workflow must also improve traceability significantly, making it straightforward to monitor pending requests, overdue returns, and the full set of equipment currently associated with each active project.

At the data and integration level, the project is expected to establish Snipe-IT as the single authoritative source for inventory, with equipment status kept in sync with the requisition state in the Maker Lab application. Project creation should transition from free-form Markdown to structured forms, producing consistent and database-ready records that support both day-to-day operation and potential future extensions such as reporting or analytics. Throughout this transition, the solution must remain compatible with the existing Maker Lab base, preserving continuity with previous work and reusing relevant content rather than discarding it.

From a usability perspective, the expected results are equally important. The laboratory technician should gain a simpler and more appropriate set of tools for managing inventory and processing requisitions — tools that do not require technical expertise to operate confidently. Student groups, in turn, should encounter a clearer and more guided experience when creating projects, submitting requests, and tracking their status across both web and mobile interfaces. If these goals are achieved, the project will not only modernise the underlying technical infrastructure but also drive meaningful improvements in real adoption and day-to-day operation of the Maker Lab.

Chapter 5

References

1. M2 - Elaboration, course material provided during the classes, 2026.
2. Milestone_2_DETIMakerLab, project presentation, 2026.
3. Information system for projects, groups and equipment requisitions, project description document.
4. M1 - Inception and project planning, course material provided during the classes, 2026.
5. Snipe-IT, open-source IT asset management platform and official documentation. Available at: https://snipe-it.readme.io/docs/introduction
6. React Documentation. Available at: https://react.dev
7. React Native Documentation. Available at: https://reactnative.dev/docs/getting-started
8. Next.js Documentation. Available at: https://nextjs.org/docs
9. FastAPI Documentation. Available at: https://fastapi.tiangolo.com
10. PostgreSQL Documentation. Available at: https://www.postgresql.org/docs