ThesisFlow: A Web Platform for Managing and Visualizing Undergraduate Theses at DCIC

Ignacio Joaquín Dotta

Departamento de Ciencias e Ingeniería de la Computación (DCIC)
Universidad Nacional del Sur, Argentina
ij.dotta@gmail.com

Abstract. This paper presents *ThesisFlow*, a web platform that digitizes the lifecycle of final projects and theses at the DCIC. The system integrates administrative workflows, professor self-service, and public analytics into a single solution. We describe the problem context motivating the system, the architecture and implementation (Spring Boot + Kotlin, React + TypeScript, PostgreSQL), the data and control flows (authentication, data ingestion and curation, analytics pipelines, and backup/restore), and the end-user interfaces. Using the live deployment, we analyze real data to uncover trends about topics, supervision, and collaboration. Results show improved data accessibility, transparency, and exploratory insights for academic decision-making. We discuss limitations (e.g., analytics scalability, testing gaps) and propose future work to extend analytics efficiency, authorization granularity, and interoperability with institutional systems.

Keywords: Academic information systems · Data visualization · Software engineering · Web applications

1 Introduction

Universities require trustworthy and accessible records of academic production to support administration, transparency, and strategic planning. At the DCIC, the management of undergraduate final projects and theses historically involved fragmented tools and manual processes, limiting visibility into trends (topics, supervision, careers) and complicating day-to-day operations. *ThesisFlow* addresses these issues with an integrated platform that (i) streamlines the registration and curation of projects; (ii) offers professors a self-service view to maintain their projects; and (iii) provides a public analytics portal to explore the department's outputs.

Objectives. The project aims to design and develop a comprehensive web system for the management, follow-up, and visualization of final projects and theses, delivering clear value for administrative staff, professors, and the broader community.

Contributions. The paper contributes: (1) a domain-driven, production-ready architecture and implementation; (2) a set of analytics visualizations (timeline, topic heatmap, collaboration network, and career statistics); (3) an evaluation based on the live instance and dataset; and (4) an open roadmap grounded in observed limitations and user feedback.

Paper structure. Section 2 reviews related work. Section 3 details requirements, architecture, and interfaces. Section 4 reports results and discoveries from live analytics. Section 5 discusses impact and limitations. Section 6 concludes and outlines future work.

2 Related Work

We review academic project management systems and analytics portals, highlighting three gaps our system addresses: (1) alignment between administrative workflows and public analytics, (2) professor-centric self-service without compromising data integrity, and (3) a minimal deployment footprint enabling departmental ownership.

3 System Proposal

3.1 Requirements

Functional. Manage projects, people (students/professors), careers, tags/domains; curate and import approved proposals (CSV); provide a public analytics portal (filters, drill-down); support professor logins and project ownership; enable backup/restore.

Non-functional. Security (JWT, role-based access), maintainability (modular layering, typed DTOs), scalability (DB-centric aggregation as future work), usability/accessibility (responsive UI), and operational reliability (health checks, containerized deployment).

3.2 Architecture and Implementation

High-Level Architecture ThesisFlow follows a three-tier architecture with clear boundaries between backend, frontend, and database. Infrastructure is containerized for reproducible deployments.

Backend (Spring Boot + Kotlin) The backend exposes REST APIs for administrative and public endpoints, enforces authentication and authorization, implements the analytics pipeline, and orchestrates bulk import and backups. Key components include: `AnalyticsService` (in-memory aggregations for timelines, heatmaps, networks, and dashboards), repositories and services for core entities, and backup/restore services (table export/import with dependency order).

Frontend (React + TypeScript) The SPA provides an admin console and a public analytics portal. It uses React Query for data fetching/caching, a consistent UI library for tables and forms, and ECharts/Recharts/vis-network for visualizations. Public endpoints back the analytics portal; authenticated routes power admin workflows.

Database (PostgreSQL + Flyway) A normalized schema stores projects, participants, careers, tags, and application domains. Integer primary keys support relations; externally-exposed UUIDs preserve stable identifiers. Schema migrations are tracked with Flyway.

Authentication and Authorization Admins log in with credentials; professors authenticate via magic links that issue JWTs. Authorization checks ensure professors can only access their projects. Public analytics are readable without authentication.

Data and Control Flow Data originates from CSV imports or manual creation; public endpoints expose curated subsets for analytics. The analytics pipeline performs aggregations and joins on repository queries; backup services export/import JSON snapshots.

3.3 Interfaces

Administrative Interfaces Tables to list, filter, create, and edit projects, people, careers, domains, and tags. Import views support CSV uploads with validation, row-level statuses, and summaries. Backup manager provides stepwise restore with safety checks.

Professor Self-Service Professors view and filter their own projects; the system constrains results by professor identity to ensure ownership and privacy.

Public Analytics Portal We present four visualizations: (i) timeline of projects per professor per year; (ii) heatmap of topics (tags) over time; (iii) professor collaboration network where node size reflects activity and edge width reflects co-supervision frequency; and (iv) career-by-year statistics with intensity shading. Each view shares filter controls (career, professor, date range, topic) and complementary summary cards (counts, percentages).

4 Results and Discoveries

We evaluate ThesisFlow using the live instance and dataset. All figures in this section are derived from production analytics endpoints at the time of writing.

4.1 Dataset Overview

We report the number of projects, unique professors, students, application domains, and topics (tags), and describe the distribution across careers and years.

4.2 Trends Over Time

The timeline shows the evolution of projects per professor per year. We highlight peaks and sustained supervision activity, and discuss whether topic diversity correlates with supervision volume.

4.3 Topic Popularity

The heatmap reveals growth or decline in certain topics (e.g., AI, data science, systems). We identify emerging areas and long-term staples of the curriculum.

4.4 Collaboration Network

The professor network uncovers co-supervision communities. We compute degree, weighted degree, and connected components to characterize collaboration patterns.

4.5 Career Participation

A table summarizes counts and proportions by career and year, enabling comparisons across programs and cohorts.

4.6 Operational Outcomes

We summarize deployment status, health checks, email pipeline (magic links), import/backup behavior, and any reliability incidents observed during evaluation.

5 Discussion

5.1 Impact Versus Objectives

We map the delivered modules and features to the initial objectives, discussing benefits for administration, professors, and the public community. We describe observed improvements in transparency and decision support.

5.2 Limitations

Current analytics rely on in-memory computation; with larger datasets, DB-level aggregations or a caching layer would be beneficial. Authorization checks are service-level; finer-grained policies could be explored. Testing coverage is improving but not yet comprehensive.

5.3 Threats to Validity

We discuss data completeness (historical gaps), measurement biases (tagging practices), and interface learnability for new users.

6 Conclusions and Future Work

ThesisFlow demonstrates that a department-owned platform can unify academic project management and analytics, offering immediate value with modest operational complexity. Future work prioritizes (i) pushing aggregations into the database and adding caching; (ii) enhancing authorization granularity and auditability; (iii) extending public APIs and export/reporting; and (iv) integrating with institutional services to improve data freshness and coverage.

Acknowledgments We thank the DCIC administrative staff and professors for their feedback and time during requirements elicitation and acceptance.

A Appendix: Screenshots and Diagrams (Optional)

Include here UI screenshots (admin console, analytics views) and architecture/ER/UML diagrams as figures. Ensure high-resolution images and captions.