Dedication

To Those Who Shape Dreams Into Reality

In the journey of life, some moments become milestones,
and some people become the reason for those milestones.

To my beloved parents,
whose endless love and sacrifices have been the wind beneath my wings.

To my siblings and close friends,
whose constant support, laughter, and shared moments
have been my anchor in challenging times.

To the exceptional university at ENSAM Casablanca,
whose wisdom and mentorship have transformed
challenges into opportunities for growth.

To my fellow students at ENSAM,
whose camaraderie and shared dreams
have made this journey truly special.

To the remarkable team at VOID,
whose trust and guidance have helped me
discover new horizons of possibility.

This work is a reflection of all the love,
support, and inspiration you have given me.

- Hamza

Acknowledgement

As I reflect on this transformative journey, I am filled with profound gratitude for the countless individuals who have contributed to the success of this internship and the completion of this report. Their guidance, support, and wisdom have been instrumental in shaping not just this project, but my professional growth.

My deepest appreciation goes to **Professor Mustapha Hain** of ENSAM Casablanca, whose academic mentorship has been nothing short of transformative. His insightful guidance, rigorous feedback, and unwavering encouragement have not only enhanced the quality of this work but have also profoundly influenced my approach to problem-solving and research.

I am equally indebted to **Mr. Hamza Bahlaouane** at VOID Digital Agency, whose visionary leadership and technical expertise have been pivotal to this project's success. His trust in my capabilities, willingness to share knowledge, and commitment to fostering innovation have created an environment where learning and growth flourished.

The VOID team has been an exceptional source of inspiration and support. I am particularly grateful to:

- **Mr. Yahya Chahine** (DevOps Engineer) for his invaluable technical guidance and patient mentorship
- **Mr. Mjid** (Project Director) for his strategic vision and leadership
- **Mr. Khalid** (Technical Lead) for his technical expertise and mentorship
- **Mr. Salah** (Full Stack Developer) for his collaborative spirit and technical insights
- **Mr. Karim** and **Mr. Mounssif** (Frontend Developers) for their teamwork and support

The foundation of this practical work rests upon the solid theoretical knowledge imparted by the distinguished university at ENSAM, whose dedication to excellence in teaching has been fundamental to my academic journey. To my family and friends, whose unwavering support and encouragement have been my constant source of strength your belief in me has made this achievement possible and this work is as much yours as it is mine.

Abstract

This report documents the comprehensive work undertaken during my final-year internship at VOID Digital Agency, a prominent independent UX and digital solutions firm in Morocco. The internship focused on two primary objectives: the implementation of an advanced CI/CD pipeline and the development of an innovative Intern Management Platform.

The CI/CD pipeline implementation leveraged modern DevOps practices, including Docker containerization, resulting in a 60% reduction in deployment time and enhanced development workflow efficiency. The Intern Management Platform, built on a headless Drupal architecture with React/Next.js frontend, introduced a sophisticated solution for managing intern applications, onboarding processes, and performance tracking.

The technical stack encompassed PHP, Docker, React, and Next.js, demonstrating the integration of modern web technologies with robust development practices.

The project faced several technical challenges, including container networking complexities, test reliability issues, and headless CMS integration hurdles. These challenges were systematically addressed through innovative solutions and best practices.

Keywords: DevOps, CI/CD, Docker, Headless CMS, Drupal, React, Next.js, Intern Management, Containerization, Web Development, PHP

Résumé

Ce rapport présente le travail effectué lors de mon stage de fin d'études chez VOID Digital Agency, une agence indépendante spécialisée dans l'expérience utilisateur et les solutions digitales au Maroc. Le stage s'est concentré sur deux objectifs principaux : la mise en place d'un pipeline CI/CD avancé et le développement d'une plateforme innovante de gestion des stagiaires.

Le pipeline CI/CD, basé sur les pratiques DevOps modernes comme la conteneurisation Docker, a permis de réduire de 60% le temps de déploiement et d'améliorer l'efficacité du développement. La plateforme de gestion des stagiaires, construite sur une architecture Drupal headless avec un frontend React/Next.js, a facilité la gestion des candidatures, l'intégration et le suivi des performances.

La stack technique comprenait PHP, Docker, React et Next.js, illustrant l'intégration de technologies web modernes avec des pratiques de développement robustes.

Le projet a rencontré plusieurs défis techniques, notamment la gestion du réseau des conteneurs, la fiabilité des tests et l'intégration du CMS headless. Ces difficultés ont été surmontées grâce à des solutions innovantes et aux bonnes pratiques.

Mots-clés: DevOps, CI/CD, Docker, Headless CMS, Drupal, React, Next.js, Gestion des stagiaires, Conteneurisation, Développement web, PHP

Contents

List of Figures

List of Acronyms

API *Application Programming Interface*

CI/CD *Continuous Integration/Continuous Deployment*

CMS *Content Management System*

CTA *Call To Action*

DevOps *Development and Operations*

Drush *Drupal Shell*

GitOps *Git Operations*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hypertext Transfer Protocol Secure*

JWT *JSON Web Token*

OAuth *Open Authorization*

OWASP *Open Web Application Security Project*

PHP-FPM *PHP FastCGI Process Manager*

QA *Quality Assurance*

SDLC *Software Development Life Cycle*

SEO/SEA *Search Engine Optimization/Search Engine Advertising*

SMTP *Simple Mail Transfer Protocol*

SSL *Secure Sockets Layer*

TLS *Transport Layer Security*

UUID *Universally Unique Identifier*

UX/UI *User Experience/User Interface*

XHProf *XHProf PHP Profiler*

General Introduction

In today's digital landscape, building robust, scalable, and user-centric web platforms is crucial for staying competitive. Since 2005, **VOID Digital Agency** has established itself as a leader in UX and digital solutions, serving major clients across banking, telecom, healthcare, retail, and cultural sectors.

During my final-year internship at **VOID**, I developed an **Intern Management Platform** using headless Drupal CMS and React/Next.js. This platform streamlines intern applications, onboarding, training, and evaluations. Additionally, I contributed to DevOps improvements, server migrations, and custom Vactory module development.

This report is structured as follows:

Chapter 1: Context The first chapter presents the general context of the project, providing an overview of VOID Digital Agency, their services, and technical infrastructure.

Chapter 2: Design This chapter details the analysis and design phase, covering requirements gathering, system architecture, and interface design for the Intern Management Platform.

Chapter 3: Development The third chapter describes the implementation phase, focusing on frontend and backend development, along with CI/CD pipeline setup.

Chapter 4: Results Chapter four evaluates the implemented solution and its impact on the internship management process.

Chapter 5: Future The final chapter examines technical challenges encountered and proposes future improvements.

Throughout this project, I applied agile Scrum methods and DevOps practices, including Bitbucket pipelines and Docker containerization. This report demonstrates how modern web architectures and automation contribute to building efficient, scalable digital platforms.

1 General Context of the Project

1.1 Introduction

In today's fast-paced digital economy, organizations must deliver **seamless**, **scalable**, and **engaging** web experiences to stay ahead. **VOID Digital Agency**, established in 2005, has become a recognized leader in **user experience** and **digital solutions** for banking, healthcare, telecom, retail, and cultural clients.

During my final-year internship at **VOID**, I undertook two parallel streams of work:

- **Core Project:** Design and implementation of a **CI/CD pipeline** alongside a headless-Drupal-based **Intern Management Platform**, automating build-test-deploy workflows and centralizing trainee onboarding, progress tracking, and reporting.
- **Side Projects:** Infrastructure enhancements, including server migrations from production to test environments, building custom **PHP-FPM Docker images**, and developing an automated dependency-update tool integrated into the CI/CD workflow.

By embedding these initiatives within **VOID's** agile framework, this report demonstrates how modern DevOps practices and decoupled architectures can drive both **efficiency** and **quality**, delivering robust, maintainable digital solutions.

1.2 Presentation of VOID

Figure 1.1: Logo of VOID Digital Agency

Founded in 2005, **VOID Digital Agency** is an independent Moroccan firm specializing in **user experience** and **digital transformation**. With offices in Casablanca and Paris, VOID serves clients across banking, insurance, healthcare, telecom, retail, and culture. The agency's mission is to design **efficient digital experiences** and build **robust, future-proof architectures** enhanced by **reactive interfaces**.

Over nearly two decades, VOID has grown to a team of 35 multidisciplinary experts, delivering end-to-end solutions from ideation to maintenance. Key offerings include:

- **Strategic Consulting:** Digital strategy, brand audits, and security assessments.
- **Content & Media:** Transmedia storytelling, institutional films, and social-media activations.
- **Digital Platforms:** Headless CMS integrations, custom web/mobile applications, and e-learning solutions.

1.2.1 Core Values

VOID Digital Agency operates on three core principles: **User-Centricity**, ensuring solutions are built around user needs; **Excellence**, maintaining high standards in quality and security; and **Innovation**, constantly exploring new technologies and methods.

1.2.2 Technical Expertise

VOID's technical expertise spans across modern technologies. The agency uses **React/Next.js** for frontend development, **Drupal** and **Symfony** for backend systems, **Docker** and **Bitbucket** for DevOps, and **Figma** for UI/UX design.

1.2.3 Company Solutions

VOID's strategic offerings are structured around four pillars:

- **Ideation Workshops**: Co-creation sessions to align business goals with user needs.
- **Digital Strategy**: Brand audits, market positioning, and roadmap definition.
- **User Research**: Focus groups, usability testing, and data-driven persona development.
- **Security & Compliance Audits**: Technical security reviews and OWASP-aligned assessments OWASP Foundation [2025].

1.2.4 Company Services

VOID delivers end-to-end digital platforms and media services:

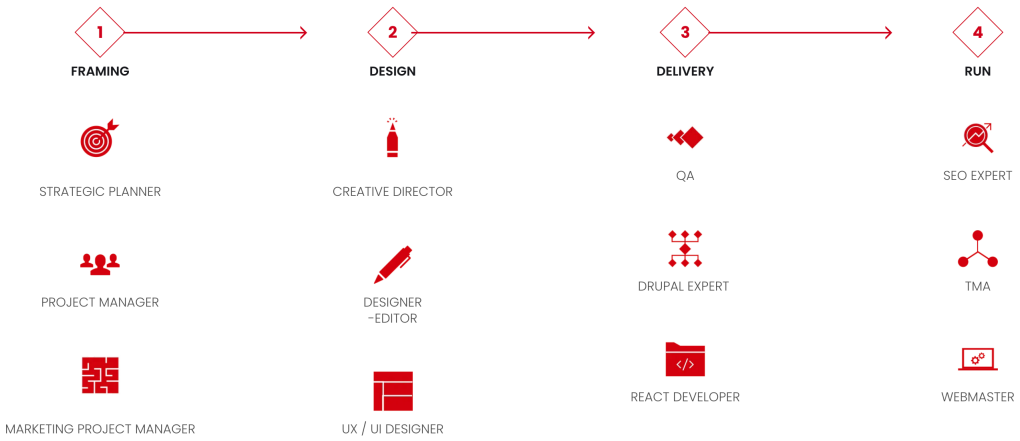

Figure 1.2: VOID Digital Agency Services

- **Transmedia Content** - Corporate videos, product demonstrations, and editorial production.
- **Social Media Activation** - Campaign design, content calendars, and performance analytics.
- **Web & Mobile Platforms** - Headless CMS (Drupal) sites, single-page applications (React/Next.js), and e-learning intranets.
- **SEO & Analytics** - On-page optimization, Google Core Web Vitals tuning, and dashboard reporting.
- **Hosting & Maintenance** - Infogérance, 24/7 support, and periodic security updates.

1.2.5 Clients and Projects

Over nearly two decades, **VOID Digital Agency** has earned the trust of a wide range of prestigious clients from various sectors. Its portfolio reflects an exceptional capacity to deliver tailor-made solutions that meet the highest standards of performance and user experience.

The agency's major clients include:

Beyond the private sector, **VOID** collaborates with governmental and cultural institutions to craft digital ecosystems that are inclusive, scalable, and centered around user engagement. This diversity in projects demonstrates VOID's ability to adapt its technical expertise and creative methodologies to a variety of contexts, while consistently maintaining excellence and innovation at the core of its services.

Figure 1.3: VOID Digital Agency Clients

1.2.6 Awards and Recognitions

Over the years, **VOID Digital Agency** has been honored with numerous national and international awards, recognizing its commitment to innovation, creativity, and digital excellence. These accolades reflect the agency's dedication to delivering outstanding user experiences and robust technological solutions.

The awards include distinctions from prestigious institutions such as:

Figure 1.4: Awards and recognitions received by VOID Digital Agency.

1.3 The Main Project

The core of the internship was centered around the design and development of an **Intern Management Platform**, an internal solution for **VOID Digital Agency**. The goal was

to streamline the onboarding, tracking, evaluation, and reporting processes for interns, while showcasing modern development practices such as **headless CMS architecture**, **agile workflows**, and **continuous deployment pipelines**.

Although the initial division of tasks assigned me the responsibility of **designing the UX/UI** of the platform, **meeting with the head of VOID** to align with the business needs, **developing the frontend** using **React/Next.js**, **integrating it with a headless Drupal backend**, in practice we adopted a **dynamic collaboration system**. This system was based on **weekly shifts**, allowing both my colleague and me to contribute to all aspects of the platform: including frontend, backend, DevOps infrastructure, and CI/CD pipelines. This approach ensured a complete understanding of the entire digital ecosystem and promoted continuous knowledge sharing.

1.3.1 Training Phase at VOID

Before engaging in the development of the Intern Management Platform, VOID Digital Agency organized a two and a half months internal training phase. This training covered the main technologies and methodologies required for the project, including Headless Drupal CMS, Next.js development, Agile workflows, DevOps practices, and CI/CD pipelines. The training ensured that all team members were aligned on technical standards and development best practices.

This foundational training was crucial in building the necessary skills and preparing for the successful delivery of the project.

1.3.2 Problematic

VOID, like many digital agencies, manages a significant number of interns each year across various departments development, UX/UI design, marketing, project management, and more. The traditional intern management relied heavily on manual tracking, disconnected communication channels, and dispersed data storage (Excel sheets, emails, local files), resulting in several critical issues:

- Lack of centralized data regarding intern profiles, progress, and evaluations.
- Difficulty for supervisors to monitor onboarding status, project allocations, and intern deliverables.
- Absence of real-time reporting on intern activity, skill assessments, and final evaluations.
- Inconsistencies and inefficiencies in document generation (attestation letters, certificates, evaluation reports).

These limitations not only increased administrative workload, but also slowed decision-making and reduced the overall quality of intern experiences at VOID. A scalable, structured, and automated solution was thus necessary to improve operational efficiency and enhance the agency's capacity to support and evaluate its interns effectively.

1.3.3 Objectives

The Intern Management Platform was designed with the following main objectives:

- **Centralization:** Create a unified platform to store and manage all intern-related data, from applications to evaluations.
- **Automation:** Automate key workflows such as onboarding tracking, evaluation form generation, and certificate issuance.
- **Scalability:** Build the system with a scalable architecture using **headless Drupal** and **Next.js**, ensuring future extensibility (adding new modules like training plans, skill assessments, etc.).
- **User Experience:** Offer an intuitive and efficient user experience for both **candidates** and **employers** through a clean and responsive UX/UI design.
- **Monitoring and Reporting:** Equip supervisors with real-time dashboards and reporting tools to track intern progression, performance, and document generation.
- **DevOps and CI/CD:** Ensure that the deployment of the platform follows best practices with a robust CI/CD pipeline, enabling automated testing, building, and deployment processes.

While I was initially focused on the UX/UI design and the development of the **Candidate Space** (frontend and backend), thanks to our rotation system, I actively contributed across all project dimensions, including DevOps setup, backend integrations, and global system testing.

1.3.4 Proposed Solution

To meet the objectives and address the problems identified, the proposed solution was to design and build a comprehensive **Intern Management Platform** structured into three interconnected modules:

- **Website:** A public-facing site aimed at presenting VOID's internship program, its philosophy, stages, and benefits. It includes a **Call-to-Action (CTA)** allowing candidates to apply directly. This site serves as the primary entry point to attract and engage potential interns.

- **Candidate Space (Espace Candidat):** A dedicated portal where candidates can:
  - Browse internship offers posted by VOID.
  - Apply to internships by submitting personal information, CVs, cover letters, and any other required documents.
  - Track the status of their application (in progress, accepted, rejected).
  - Participate in tests and interviews conducted as part of the recruitment process.

  This space was built using **React/Next.js** for the frontend and connected to a **headless Drupal CMS** backend serving structured data via APIs, ensuring flexibility, scalability, and performance.

- **Training Space (Espace Formation):** Once accepted, interns gain access to this space, modeled after modern online learning platforms such as Coursera. Here, they can:
  - Follow predefined training roadmaps aligned with VOID's standards.
  - Access courses, videos, assignments, and learning materials.
  - Complete projects and quizzes over a three-month structured program.
  - Monitor their own progression through dashboards and receive feedback from supervisors.

  This module supports VOID's mission of continuous learning and ensures that interns are fully prepared for their assigned roles.

The technical implementation was based on a **headless architecture**, where Drupal managed the content models, permissions, and workflows, while the frontend was fully decoupled, using **Next.js** for server-side rendering and dynamic interaction. The backend exposed structured data through **JSON:API** endpoints, and all user interactions were designed to be smooth and reactive, respecting the best practices of modern UX/UI design.

Additionally, the entire project was deployed using a full **CI/CD pipeline** to automate testing, building, and production delivery.

Although my initial responsibilities mainly covered the **UX/UI design**, **meetings with the head of VOID**, the development of the **Candidate Space** (frontend and headless backend integration), our **rotation system** allowed me to work on all areas, including the **Training Space** and infrastructure tasks. This dynamic approach enhanced my skills across the complete technological stack and provided a holistic understanding of the system lifecycle.

1.4 Project Management Methodology

1.4.1 Agile Methodology

VOID adopts an **Agile methodology**, specifically the **Scrum framework**, to manage and deliver its digital projects efficiently. The iterative approach ensures flexibility, faster feedback loops, and continuous improvement throughout the development lifecycle.

Each project follows the **Software Development Life Cycle (SDLC)** principles, combined with **GitOps** practices to automate deployment pipelines and infrastructure operations. Regular sprint planning, daily stand-ups, sprint reviews, and retrospectives form the core rhythm of project execution.

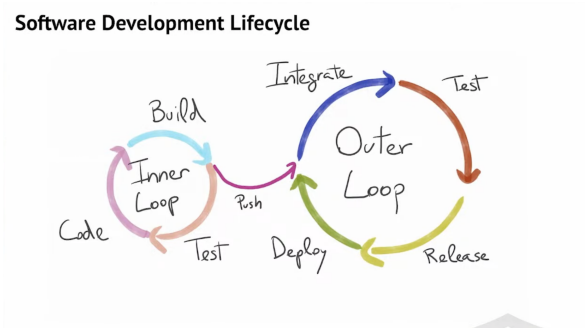

Figure 1.5: Software Development Life Cycle (SDLC)

1.4.2 Organization and Communication Tools

VOID leverages a combination of tools to streamline project tracking, team communication, and task management:

- **Redmine:** Internal project management tool similar to Jira, featuring Gantt charts and Kanban boards for visual task tracking.

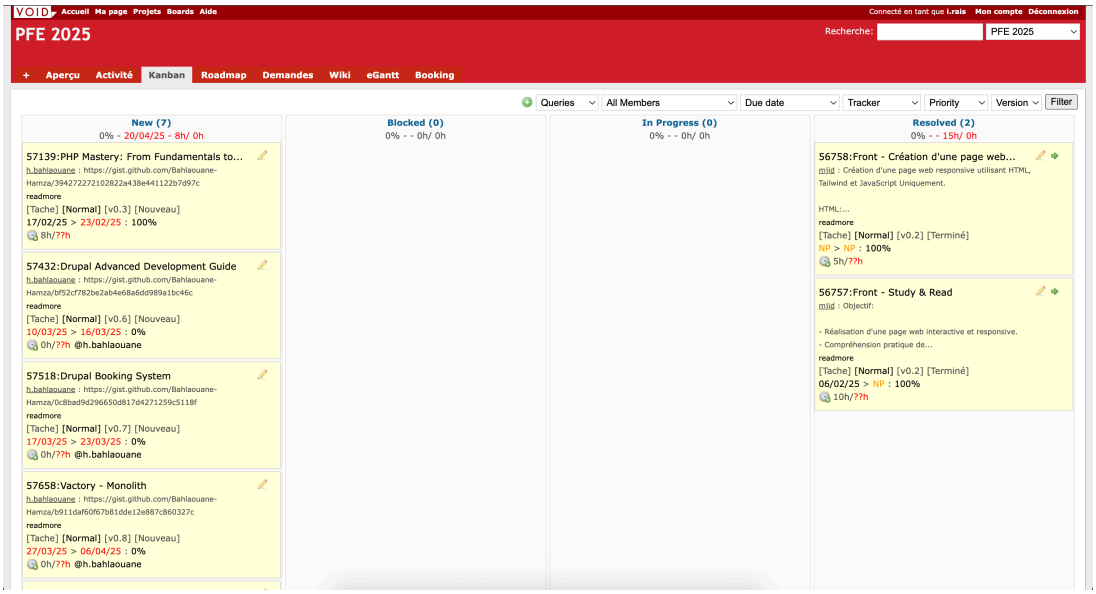

Figure 1.6: Redmine project management tool

- **Communication Tools:**

Figure 1.7: Slack

Figure 1.8: Loom

Figure 1.9: Google Meet

Figure 1.10: Google Calendar

Figure 1.11: Communication and collaboration tools used at VOID

The combination of these tools enables smooth communication, clear project tracking, and efficient collaboration across all teams at VOID.

1.5 Conclusion

This first chapter has established the general framework of the project by presenting **VOID Digital Agency**, its organizational structure, and its technical ecosystem. We have also defined the context and objectives of the main project, highlighted the problems it aims to solve, and outlined the chosen implementation approach and project management methodology.

This foundation sets the stage for a deeper exploration of the project requirements. The next chapter will focus on the detailed analysis and specification of the functional and technical needs essential for the successful realization of the platform.

2 Project Analysis and Design

2.1 Introduction

This chapter presents a comprehensive analysis of the platform's development, focusing on the requirements gathering process and design decisions. We begin by examining the current workflow and identifying key challenges, followed by establishing clear project objectives and success criteria. Through user research and requirements gathering, we define both functional and technical specifications. The chapter concludes with an overview of the system architecture and user interface design concepts, providing a complete picture of the platform's development approach.

2.2 Study of the Existing System

Before the introduction of the Intern Management Platform, VOID Digital Agency VOID Digital Agency [2025] managed internships using a patchwork of manual processes. Applications were tracked in Excel sheets, communications handled through scattered emails, and evaluations stored in separate documents. This fragmented approach created a number of persistent challenges for the organization.

2.2.1 Operational Inefficiencies

- **Process Delays:** Manual data entry and document handling often led to delays of several days in processing applications.
- **Resource Intensive:** The HR team spent a significant amount of time on repetitive administrative tasks.
- **Error Prone:** A notable percentage of applications required manual correction due to data entry mistakes.
- **Inconsistent Documentation:** Evaluation forms and reports frequently lacked standardization, making it difficult to compare or aggregate results.

2.2.2 Technical Limitations

- **Data Fragmentation:** Information was dispersed across multiple systems, making it hard to maintain a single source of truth.
- **Version Control Issues:** Multiple versions of documents circulated, leading to confusion and inconsistencies.
- **Limited Accessibility:** Accessing intern information remotely or collaboratively was often challenging.
- **Integration Challenges:** The lack of integration with existing HR and project management tools further complicated workflows.

2.2.3 Impact on Stakeholders

- **Interns:** Experienced delays in feedback and had limited visibility into their own progress.
- **Supervisors:** Found it difficult to monitor and support multiple interns effectively.
- **HR Team:** Faced increased workloads and reduced overall efficiency.
- **Management:** Had limited insight into the overall performance and return on investment of the internship program.

As the number of interns grew each quarter, these issues became more acute. Following the Software Development Life Cycle (SDLC) GeeksforGeeks [2025], the situation underscored the urgent need for a centralized, automated, and scalable platformone capable of supporting VOID's growth while ensuring data integrity, process efficiency, and a better experience for all stakeholders.

2.3 Specification Document

This specification document outlines the essential requirements and criteria for the Intern Management Platform, as defined through stakeholder collaboration and technical analysis. The document is structured to provide a clear reference for both development and evaluation.

2.3.1 Functional Requirements

- **Main Website:** Provide a public-facing portal that presents VOID's internship program, company values, and available opportunities, with clear calls-to-action redirecting candidates to the application process.

- **Candidate Space:** Enable candidates to create accounts, submit applications, upload required documents (CV, cover letter, etc.), and track their application status in real time.
- **Automated Technical Testing:** Integrate an automated technical test system that assigns, grades, and filters candidates based on their results, ensuring only competent applicants proceed.
- **Challenge Submission and Review:** Allow selected candidates to receive and submit coding or project challenges, with manual review and feedback by supervisors.
- **Video Interview Integration:** Facilitate asynchronous or scheduled video interview submissions, enabling supervisors to assess communication and motivation.
- **Selective Progression:** Implement a workflow where only candidates who pass all evaluation stages (technical test, challenge, video) are accepted as pre-hire interns (pre-embauche).
- **Training Space:** Provide a dedicated area for accepted interns to access learning resources, follow personalized training paths, complete quizzes, and track their progress.
- **Supervisor and Admin Tools:** Equip supervisors and administrators with dashboards for monitoring candidate progress, managing challenges, reviewing submissions, and generating evaluation reports and certificates.
- **Communication and Notification:** Ensure timely, automated notifications and messaging between candidates, supervisors, and HR throughout all stages.
- **Data Security and Privacy:** Enforce role-based access, secure authentication, and compliance with data protection standards for all user data and documents.

2.3.2 Non-Functional Requirements

- **Scalability:** The system must support at least 50 concurrent users and handle 100+ intern records simultaneously.
- **Security:** Sensitive data must be protected through encryption, secure authentication (OAuth 2.0 or equivalent), and audit logging.
- **Performance:** Application response times should not exceed 2 seconds for standard operations under normal load.
- **Reliability:** The platform should maintain 99.5% uptime, with automated backups and disaster recovery procedures in place.

2.3.3 Success Criteria

- The Main Website effectively communicates VOID's internship program and drives candidate engagement, as measured by application conversion rates.
- The Candidate Space enables at least 90% of users to complete the application process without support, and provides real-time status updates at every stage.
- The technical test and challenge system successfully filters out unqualified candidates, with only those meeting predefined competency thresholds advancing to the next stage.
- Only candidates who pass all evaluation stages (technical test, challenge, video) are accepted as pre-hire interns, ensuring a high standard of competency and motivation.
- The Training Space is accessed by 100% of accepted interns, with all interns completing assigned learning paths and quizzes.
- Supervisors and administrators report improved efficiency in candidate management, with a reduction in manual tracking and paperwork by at least 60%.
- The platform maintains data integrity and security, with no reported breaches or unauthorized access incidents during the evaluation period.

This specification document serves as a foundation for the design, implementation, and evaluation of the Intern Management Platform, ensuring that the solution meets both the immediate operational needs and the long-term strategic goals of VOID Digital Agency.

2.4 User Research and Methodology

To ensure the Intern Management Platform would genuinely address the needs of its users, a thorough user research phase was conducted at the outset of the project. This phase combined direct engagement with stakeholders, analysis of existing workflows, and benchmarking against industry best practices. The insights gained were instrumental in shaping both the functional and technical design of the platform.

2.4.1 Research Methods

- **Stakeholder Interviews:** In-depth interviews were held with HR staff, technical leads, former interns (us, and our colleagues). These conversations provided a nuanced understanding of the pain points, expectations, and day-to-day realities of each group involved in the internship process.

- **User Journey Mapping:** The team mapped out the complete candidate journeyfrom discovering the program on the main website, through application, technical testing, challenge submission, video interview, onboarding, and training. This exercise highlighted critical touchpoints and potential friction points.

2.4.2 User Journey Analysis

The research revealed several critical stages in the candidate and intern journey that required special attention:

- **Application Phase:** Candidates needed a simple, intuitive process for discovering opportunities, submitting applications, and uploading documents. Real-time status tracking was essential to reduce anxiety and uncertainty.
- **Selection Process:** Automated technical tests, challenge submissions, and video interviews were identified as effective ways to assess candidate skills and motivation. Clear communication of next steps and timely feedback were crucial for maintaining engagement.
- **Internship Period:** Once accepted, interns valued having a dedicated training space with structured learning paths, progress tracking, and easy access to resources. Supervisors needed tools to monitor progress, provide feedback, and manage evaluations efficiently.
- **Evaluation:** Both interns and supervisors required a streamlined process for performance assessment, documentation, and certification, with the ability to track outcomes and support future opportunities.

2.5 UML Diagrams Overview

To provide a clearer understanding of the system's functionality and structure, two key UML diagrams were developed: a Use Case Diagram and a Sequence Diagram. These diagrams illustrate the system's behavioral aspects and the interactions between different actors and components.

2.5.1 Use Case Diagram

The use case diagram (Figure 2.1) illustrates the major interactions between the different actors and the Void Training Platform system. It identifies six primary actors with distinct roles and responsibilities:

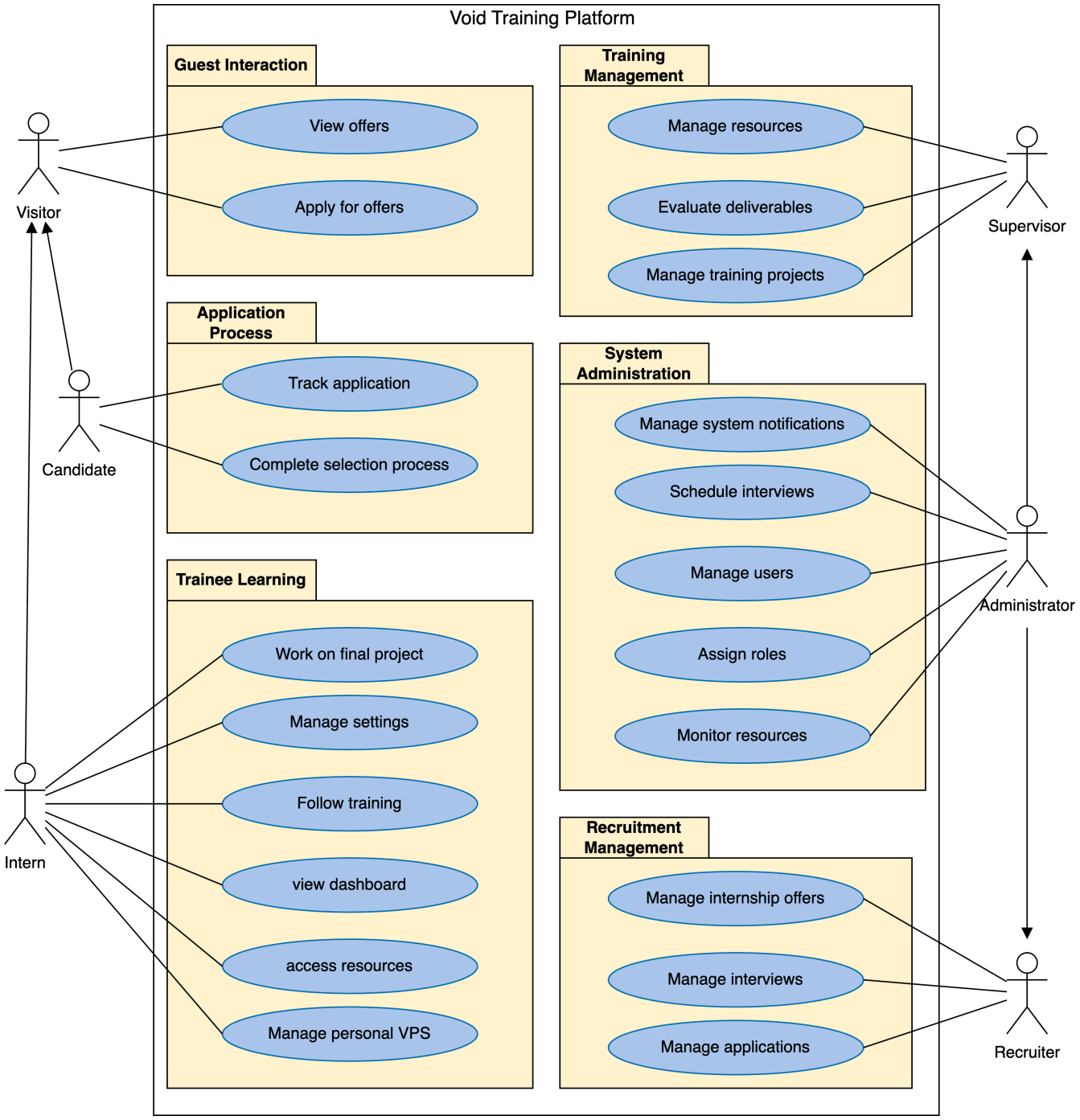

Figure 2.1: Use Case Diagram for the Intern Management Platform

The Use Case Diagram provides a comprehensive overview of the system's functional requirements and user interactions within the Intern Management Platform. This diagram identifies six primary actors with distinct roles: Visitors who can browse and apply for opportunities, Candidates who track applications and complete selection processes, Trainees who access training resources and submit projects, Supervisors who manage educational content and evaluate deliverables, Administrators who handle system configuration and user management, and Recruiters who manage internship offers and interviews. The diagram illustrates key use cases including account management, training program partici-

pation, resource access, project submission and evaluation, and administrative oversight. This comprehensive view demonstrates how the platform supports the complete intern lifecycle from initial application through training completion.

2.5.2 Sequence Diagram

The sequence diagrams represents the detailed interaction flow of the intern management platform, showing the communication between different system components. This diagram illustrates the technical architecture and data flow.

Note: All API calls in the following sequence diagrams use an API proxy to facilitate communication between the frontend and backend services.

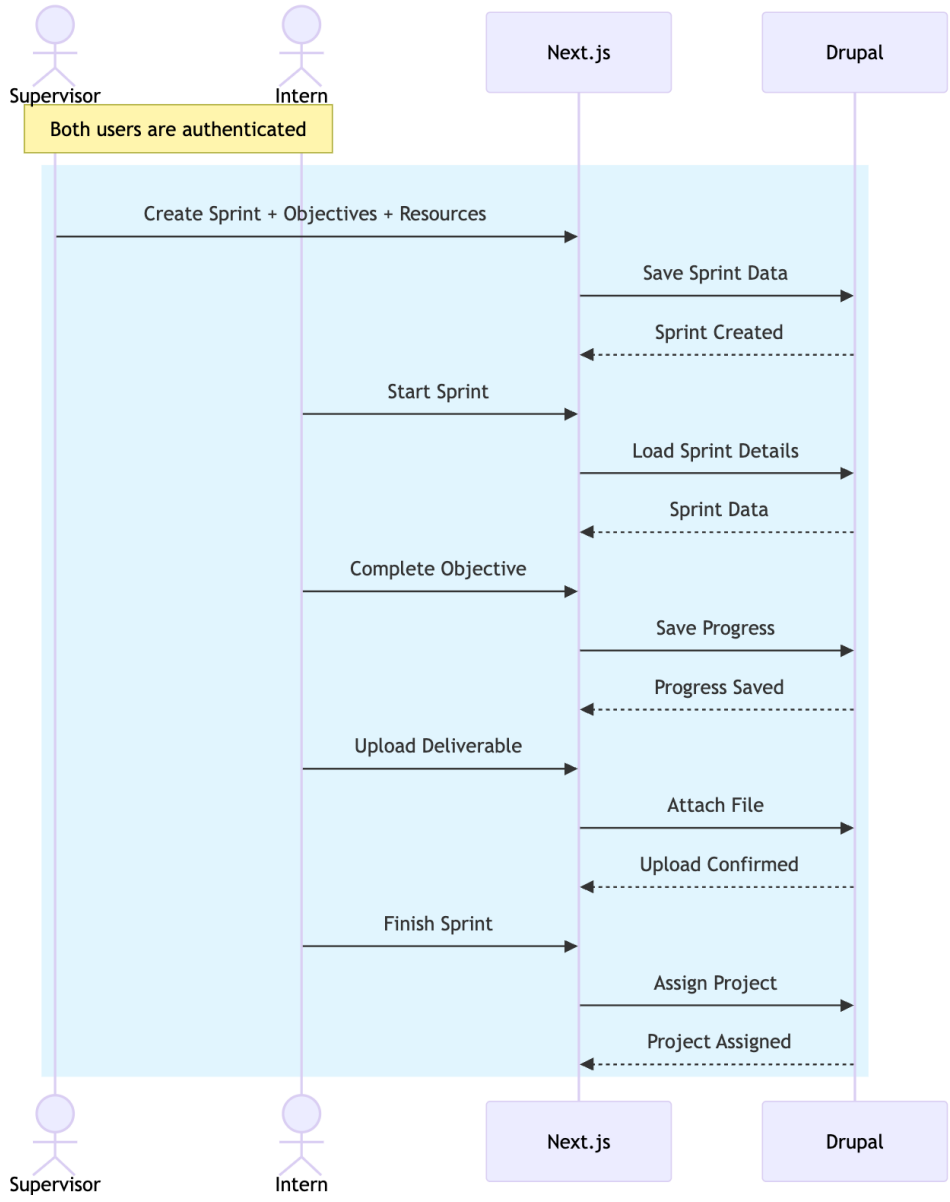

Figure 2.2: Sequence Diagram - Sprint Management Process Flow

The Sprint Management Process Flow sequence diagram illustrates the complete lifecycle of sprint management within the training platform. This diagram demonstrates how supervisors create and manage training sprints, how trainees interact with sprint content, and how the system handles progress tracking and evaluation. The flow begins with sprint creation by supervisors, followed by trainee enrollment, content delivery, progress monitoring, and final evaluation. Key interactions include automated notifications, progress updates, and the integration between the frontend training interface and backend sprint management system.

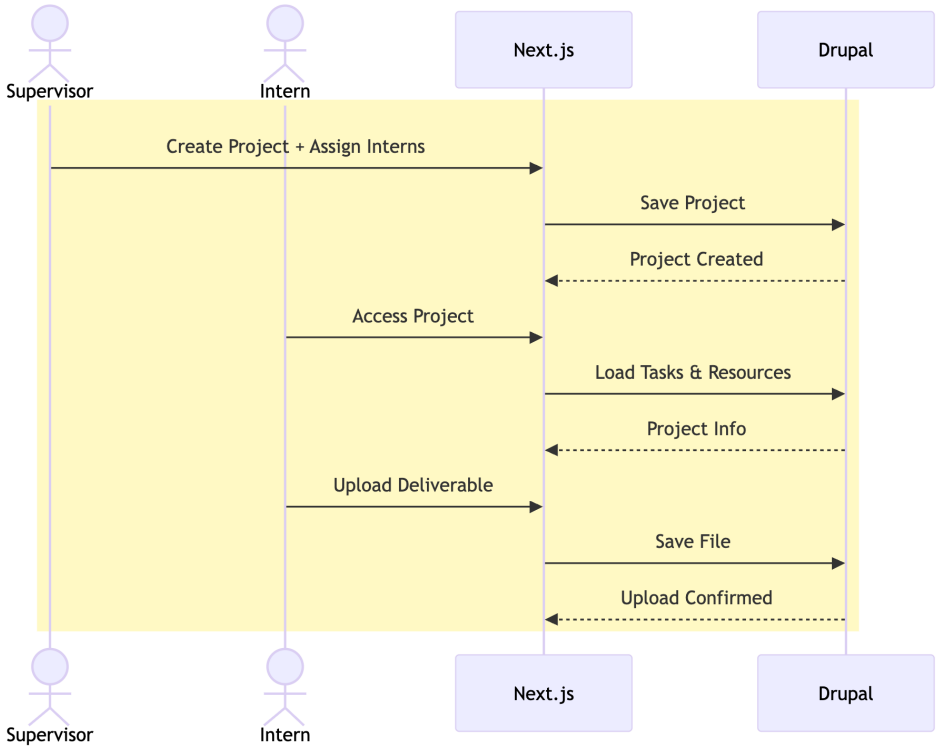

Figure 2.3: Sequence Diagram - Project Management Process Flow

The Project Management Process Flow sequence diagram showcases the comprehensive project lifecycle management system. This diagram details how trainees submit project deliverables, how supervisors review and evaluate submissions, and how the system manages project status updates and feedback loops. The process includes project creation, milestone tracking, submission workflows, review processes, and final project evaluation. The diagram highlights the automated status updates, email notifications, and the seamless integration between the project submission interface and the backend evaluation system.

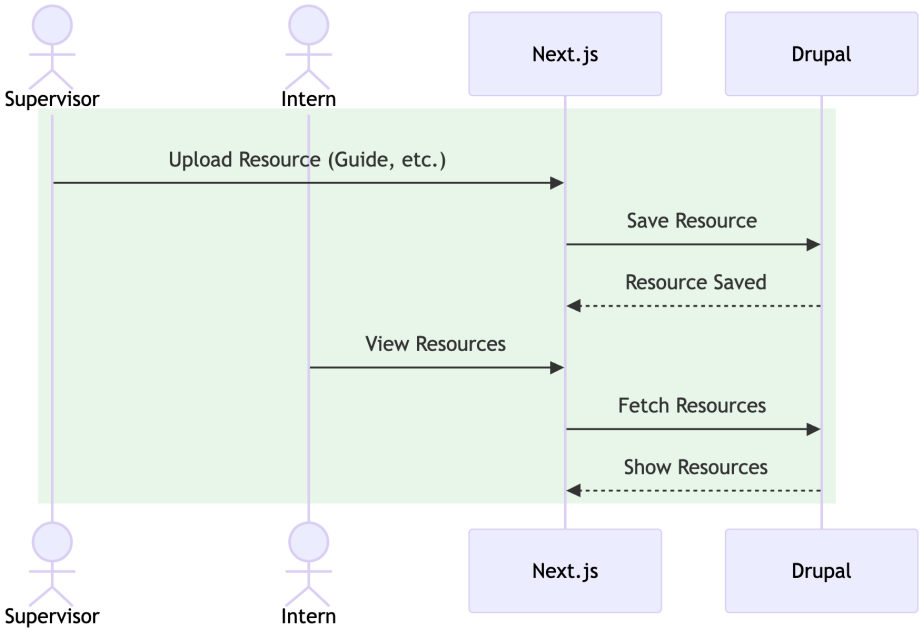

Figure 2.4: Sequence Diagram - Resource Management Process Flow

The Resource Management Process Flow sequence diagram demonstrates the complete resource lifecycle management within the training platform. This diagram illustrates how supervisors upload and organize educational resources, how trainees access and utilize these materials, and how the system manages resource categorization and access control. The flow encompasses resource creation, categorization, access management, usage tracking, and resource updates. The diagram emphasizes the role-based access control system, automated resource organization, and the integration between the resource library interface and the backend content management system.

2.5.3 Class Diagram

The class diagram (Figure 2.5) presents the complete data model and entity relationships within the Intern Management Platform. This diagram illustrates the core entities, their attributes, and the relationships between different components of the system, providing a comprehensive view of the platform's structural design.

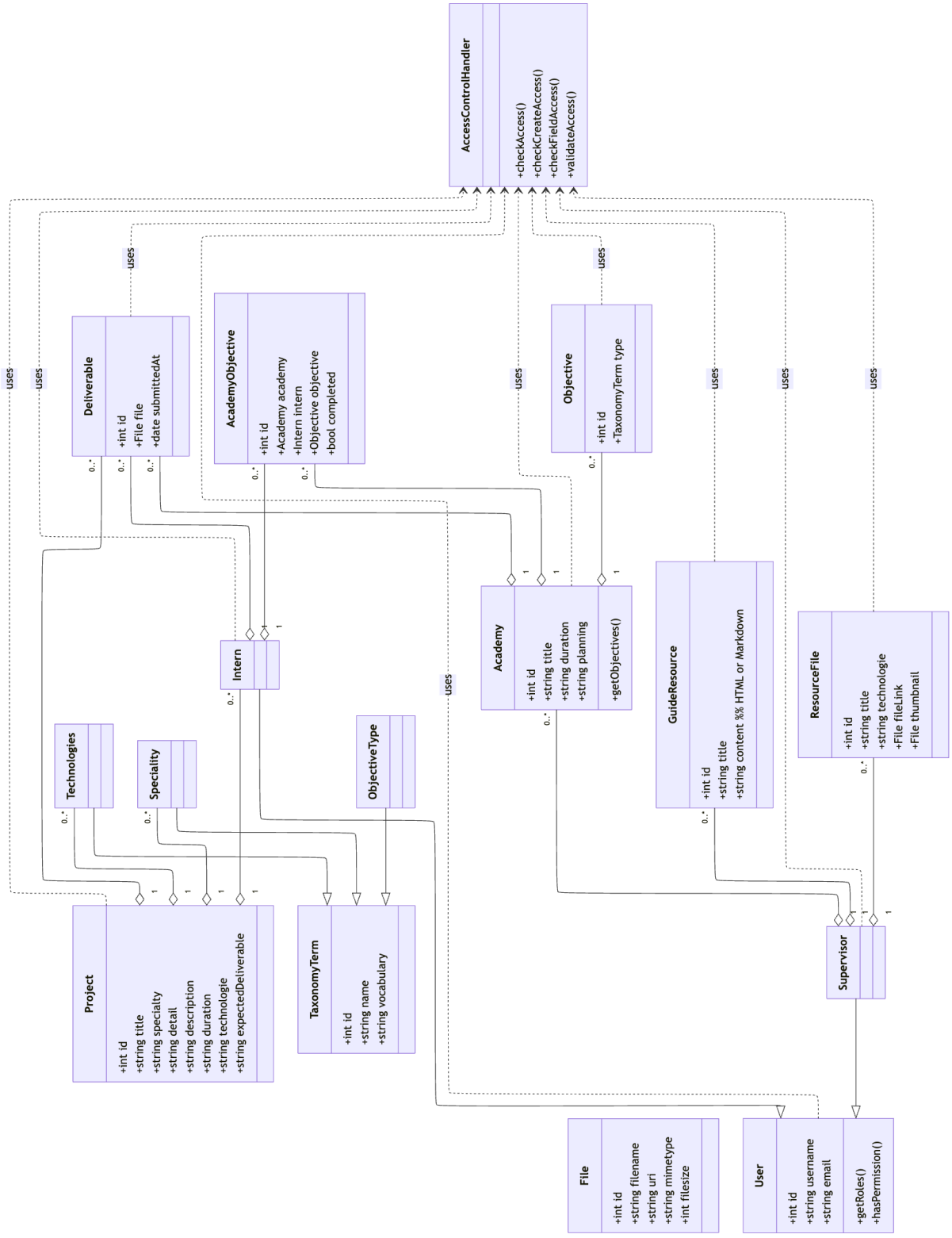

Figure 2.5: Class Diagram - System Entity Relationships and Data Model

The Class Diagram provides a comprehensive view of the system's data architecture and entity relationships within the Intern Management Platform. This diagram illustrates the core entities including User management (with different roles like Trainee, Supervisor,

Administrator), Training entities, and supporting entities. The diagram shows the relationships between entities, such as how Trainees are associated with multiple Sprints and Projects, how Supervisors manage Resources and evaluate Projects, and how the system handles Status tracking and automated Notifications. Key design patterns include role-based access control, hierarchical resource organization, and flexible status management systems that support the platform's training workflow requirements.

2.5.4 Project Timeline Diagrams

Before diving into the main project implementation, it's important to note that a significant portion of time was dedicated to training and preparation. The team underwent an intensive two and a half months training phase to master the required technologies and methodologies, including Headless Drupal CMS, Next.js development, Agile workflows, DevOps practices, and CI/CD pipelines. This training phase was crucial for ensuring all team members were aligned on technical standards and development best practices. Following the training phase, the main project implementation was structured into six focused sprints, each targeting specific deliverables and building incrementally on previous work:

- **Sprint 1: Setup** Initial setup of the Vactory distribution and Drupal environment, along with the DevOps stack (OrbStack, Docker, EasyPanel).
- **Sprint 2: Main Website** Development of the main website, including widget creation (Hero, FAQ, CTA, Process), job ads content types, taxonomies, and job offers listing modules.
- **Sprint 3: Training Platform (Backend)** Implementation of backend features including training entities, project modules, resources management, access control, status taxonomy, and automated progress tracking.
- **Sprint 4: Training Platform (Frontend)** Development of the training platform frontend, featuring the training space interface, resource library, status and objective trackers, project submission system, and progress monitoring dashboard.
- **Sprint 5: Admin Tools and Integration** Development of admin dashboards, iframe review tools, role and permission setup, and the email notification system. Integration of all modules and finalization of admin features.
- **Sprint 6: Testing, Documentation, and Deployment** Comprehensive testing and debugging (frontend and backend), CI/CD pipeline setup, Docker Compose finalization, and preparation of final documentation and delivery.

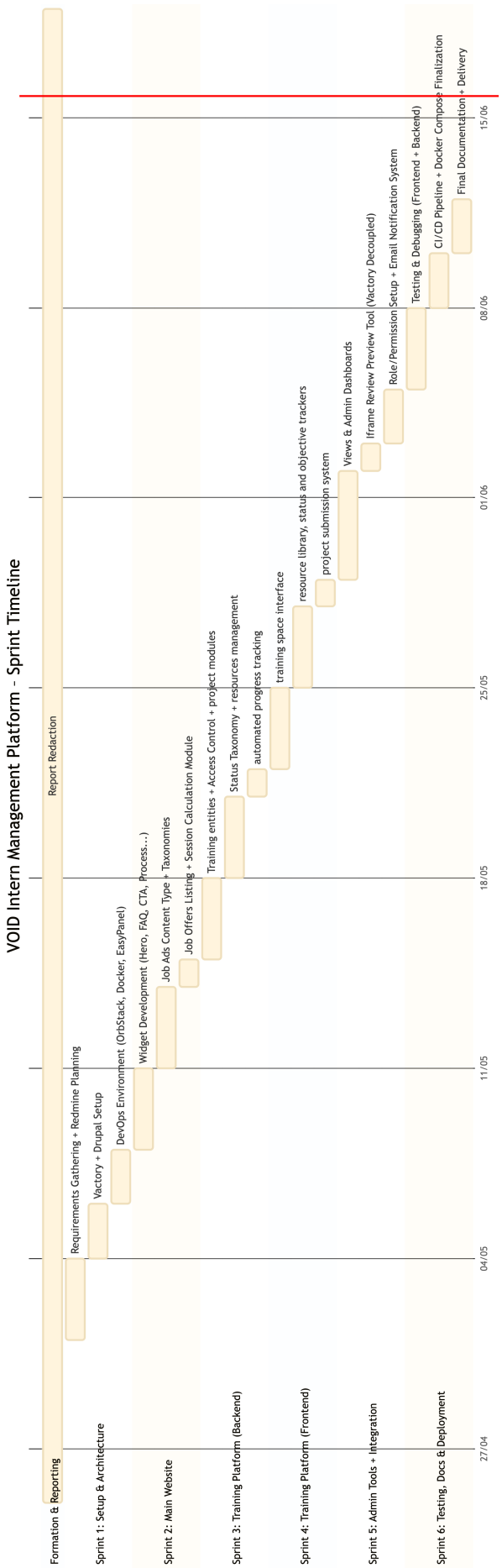

Figure 2.6: Project Implementation Gantt Chart - VOID Intern Management Platform Sprint Timeline (6 Weeks)

2.6 Mockups and Wireframes

Based on the insights gathered during the research phase, a first version of the mockups and wireframes was designed to visualize the user interfaces and main flows.

The mockups were centered around the three core spaces of the platform:

- **Landing Website:** Homepage presenting internship opportunities, company culture, and a call-to-action to apply.

Figure 2.7: High-Fidelity Mockups Landing Website

- **Candidate Space:** Area dedicated to candidates for profile creation, job application, test session participation, and training follow-up.

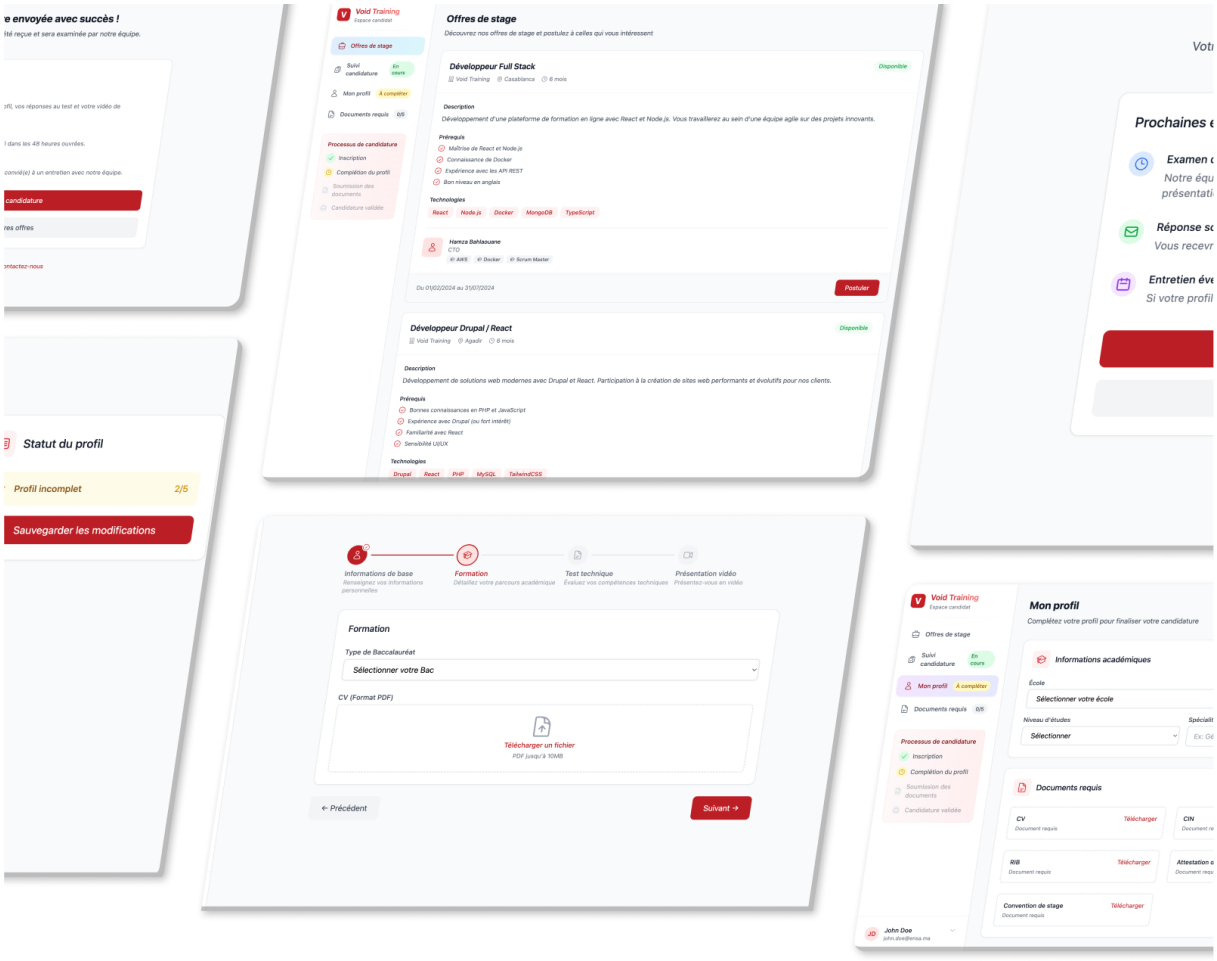

Figure 2.8: High-Fidelity Mockups Candidate Space

- **Training Space:** Educational portal where accepted interns can access courses, complete quizzes, and track their training progress.

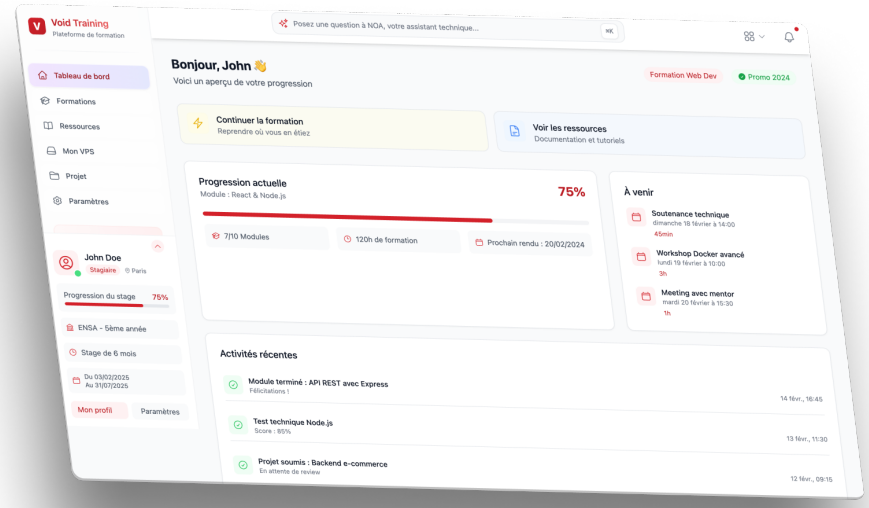

Figure 2.9: High-Fidelity Mockups Formation Space

- **Admin Panel:** Interface for HR managers and supervisors to manage offers, monitor intern progression, and generate reports and certifications.

The UX design prioritized simplicity, clarity, and responsiveness to ensure a smooth experience across desktop and mobile devices.

2.7 Conclusion

This chapter laid the foundation for the technical and functional understanding of the Intern Management Platform. By outlining the problem, conducting a user journey analysis, identifying system functionalities, and modeling the architecture using UML diagrams, the analytical groundwork was established.

This phase not only clarified user needs but also guided design choices that ensured alignment with business goals and technical feasibility. The next chapter will delve into the practical implementation of these components, covering the architecture setup, technologies used, integration strategies, and component development.

3 Main Project Implementation

3.1 Introduction

This chapter details the implementation and development phase of the Intern Management Platform. The development process followed an agile methodology, with one-week sprint cycles with the encadrant and two-week sprint cycles with the CEO & CTO. The implementation phase spanned over six weeks, focusing on delivering a robust, scalable, and user-friendly platform.

3.2 Programming Languages and Frameworks

3.2.1 Programming Languages

PHP, HTML5, JavaScript: Our project uses PHP for server-side development and Drupal CMS integration, HTML5 for web page structure, and JavaScript for interactive features and client-side functionality.

Figure 3.1: Core Programming Languages: PHP, HTML5, and JavaScript

3.2.2 Frameworks and Libraries

Next.js, Tailwind CSS, Drupal: We use Next.js for frontend development and SEO optimization, Tailwind CSS for responsive design and styling, and Drupal as our content management system.

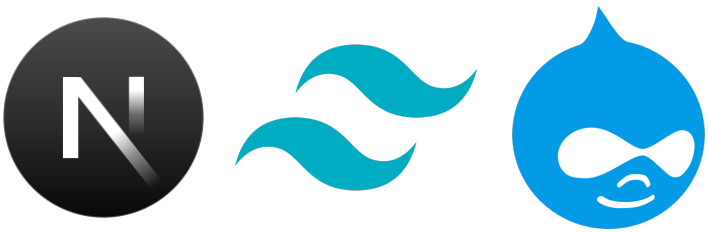

Figure 3.2: Core Frameworks and Libraries: Next.js, Tailwind CSS, and Drupal

3.2.3 Database and Caching

MySQL, Redis, Memcached, Mailhog: MySQL serves as our primary database, Redis and Memcached handle caching and performance optimization, while Mailhog manages email testing in development.

Figure 3.3: Database and Caching Tools: MySQL, Redis, Memcached, and Mailhog

3.3 Development Tools

VS Code & Docker: VS Code for development with web technology support, Docker for containerized environments.

Orbstack & Git: Orbstack for Docker management on macOS, Git for version control and collaboration.

Bitbucket & Docker Hub: Bitbucket for repository hosting, Docker Hub for container image management.

EasyPanel & Nginx: EasyPanel for staging server management, Nginx for web serving and traffic routing.

Figure 3.4: Development Tools: VS Code, Docker, Orbstack, Git, Bitbucket, Docker Hub, EasyPanel, and Nginx

3.4 Development Environment

Our development environment is built for consistency, scalability, and security. We use Orbstack to manage all containers, making setup and code syncing easy.

3.4.1 Backend Containers

The backend is structured as follows:

- **Nginx + PHP-FPM + XHProf Container:** Runs the Drupal code, serves the application, handles all web requests, and provides profiling/debugging with XHProf. Code is synced from the host using Orbstack.
- **Memcached Container:** Connected to the main container for backend caching, improving speed and performance.
- **MySQL Container:** Stores all structured data for Drupal, connected internally to the main container and PhpMyAdmin.
- **PhpMyAdmin Container:** Provides a GUI for managing the MySQL database.
- **Mailhog Container:** Used for email testing. Drupal is configured to send emails via SMTP to Mailhog, which catches them for testing.

- Ports are exposed as follows: Drupal (8080), PhpMyAdmin (8085), Mailhog (8002). - Database and Memcached containers are only accessible via the internal Docker network for security.

3.4.2 Frontend Containers

The frontend setup includes:

- **Node.js Container:** Runs the Next.js application.
- **Redis Container:** Used for caching and boosting frontend performance.
- **Redis Commander Container:** Allows management of Redis cache via terminal or GUI.

The Next.js frontend communicates with the backend, but a reverse proxy is set up so the backend is not directly exposed. All API requests go through the frontend, making the backend invisible to the outside.

Widget-Based Component Binding

To empower non-technical users, we built a widget system in Drupal. Each widget comes with a **screenshot**, a **.twig** template for monolithic deployments, and a **settings.yml** for the decoupled frontend. On the Next.js side, a webpack plugin scans for ***Widget.jsx** files, reads their metadata, and links them to backend widgets. This setup allows for lazy loading and keeps the bundle size in check.

Route Translation and UUID Resolution

SEO-friendly URLs are a must. Using Drupal's Pathauto, every alias (like /internship-offers) is mapped to a UUID and content type. The frontend then fetches the right data via JSON:API, so routing stays flexible and isn't tied to backend structure. Here is an example of a route translation:

1. The alias (e.g., `/internship-offers`) is passed to a translation endpoint.
2. A response returns the `UUID` and `content type`.
3. The UUID is used to fetch detailed JSON:API data for rendering.

3.4.3 Secure API Proxying

Direct backend exposure is risky. All API calls go through a Next.js proxy endpoint, which lets us tweak headers, strip cookies, and add caching or rate limits if needed. This extra layer keeps things secure and gives more control over traffic.

3.4.4 Caching Architecture

Performance matters, especially as the platform grows. We set up Memcached on the Drupal side for dynamic content and Redis on the frontend for frequently accessed collections and sessions. Redis Commander makes it easy to inspect and debug cache contents.

3.4.5 Mail Testing Infrastructure

Testing email flows without spamming real users is crucial. MailHog catches all outgoing mail in the dev stack, so we can check templates and triggers safely.

3.4.6 Component Development with Storybook

Storybook runs in its own container, letting us and the team build and preview UI components in isolation. Mocked backend data and stakeholder previews help us catch design issues early.

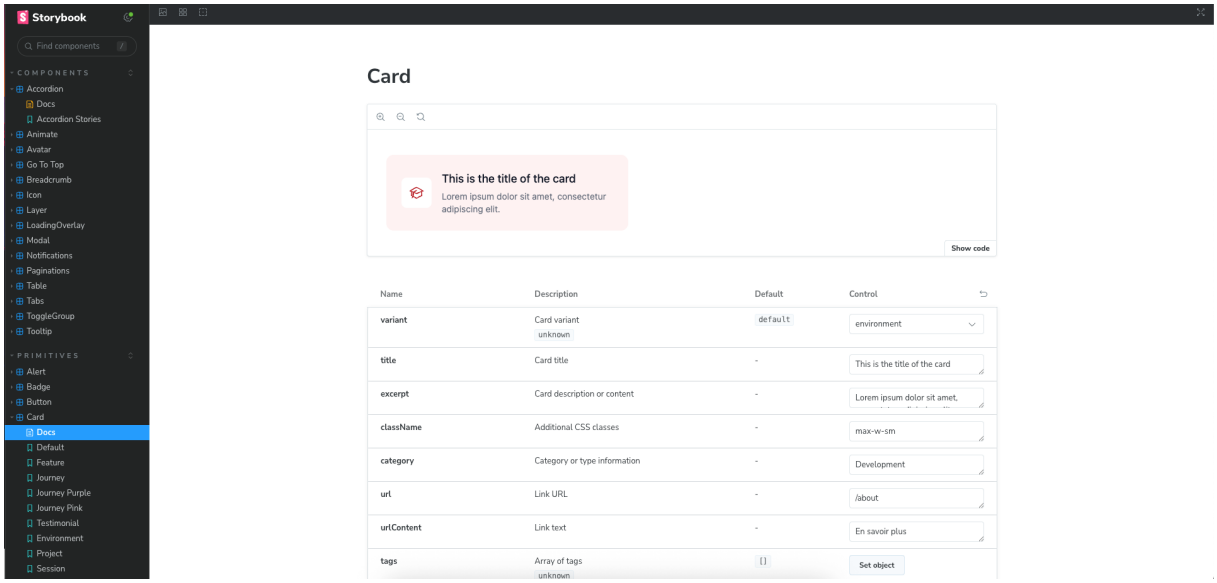

Figure 3.5: Storybook UI

3.4.7 Routing Strategy

We opted for Next.js Pages Router over the newer App Router. The Pages Router is stable, well-documented, and better supported by the modules we needed. Data fetching is straightforward, and the learning curve is lower for new contributors.

3.5 Backend Implementation

3.5.1 Introduction

The backend of the Intern Management Platform is built on Drupal 10, using VOID Agency's Vactory distribution. We chose this stack for its modularity and JSON:API support, which enables seamless frontend integration. The backend consists of three main components: the Main Website, Candidate Space, and custom modules for recruitment automation.

3.5.2 Drupal Architecture and API Strategy

What is Drupal?

Drupal is an open-source content management system (CMS) used to build and manage websites. It provides a flexible back office for content creation, user management, and site configuration.

Our Use of Drupal

In our project, we use Drupal mainly as a backend service. The Drupal back office is used for content management, while the frontend is built separately. We do not use Drupal's default theming or page rendering for the public site.

Exposing Content with JSON:API

To connect Drupal with our frontend, we enabled and configured the **JSON:API** module. This module allows us to expose Drupal content, users, menus, and other entities as RESTful API endpoints. These endpoints are configurable and can be consumed directly by the frontend application.

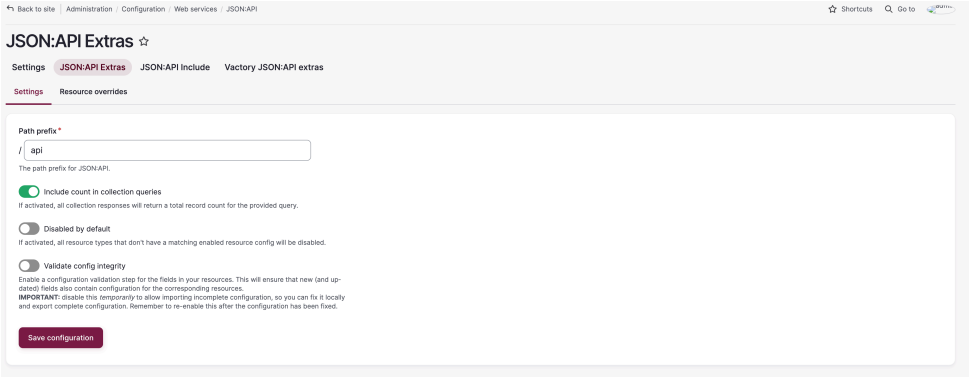

Figure 3.6: JSON:API endpoint configuration with path prefix /api

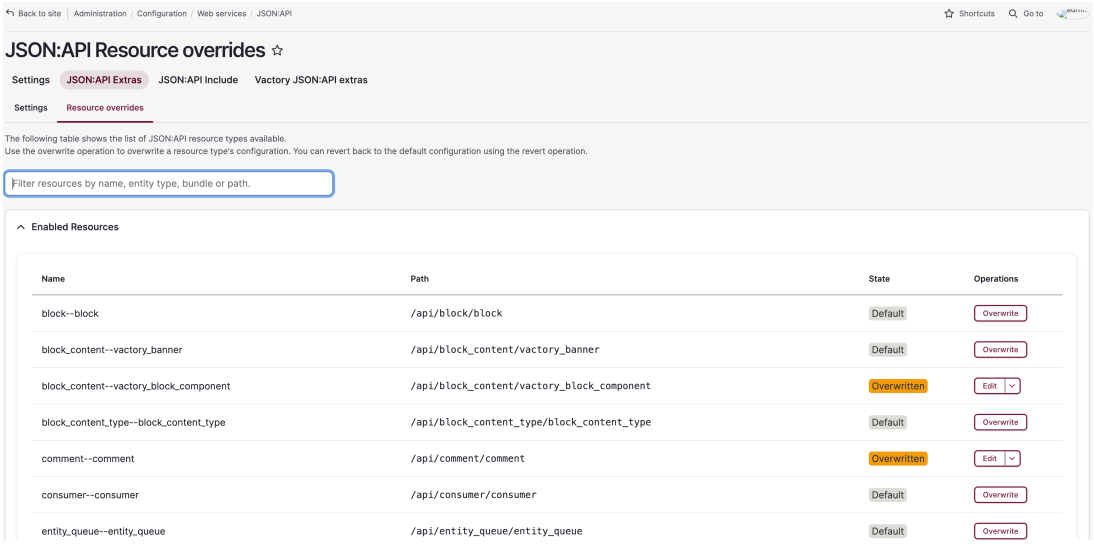

Figure 3.7: Exposed JSON:API resources in Drupal

Custom API Endpoints for Menus and Translations

For essential features like menus and translations, we configured two custom API endpoints:

- `/api/_translations` Returns translation data for the site.
- `/api/_menus?menu_name={main,footer,...}` Returns menu structures for navigation.

Building Website Structure with Dynamic Fields

To enable flexible page construction, we use the **Dynamic Field** module and a modular development approach. Website componentssuch as headers, footers, and card listsare implemented as reusable Drupal entities, which we refer to as *widgets*. These widgets are defined in the codebase and can be placed visually within page layouts using the Drupal back office.

The typical workflow is as follows:

1. Create widget templates in the codebase, specifying the data to be exposed.
2. Use the Drupal interface to assign widgets to specific page regions.
3. The JSON:API module automatically exposes the page structure and widget data as API endpoints.
4. Consume these endpoints to render the final user interface.

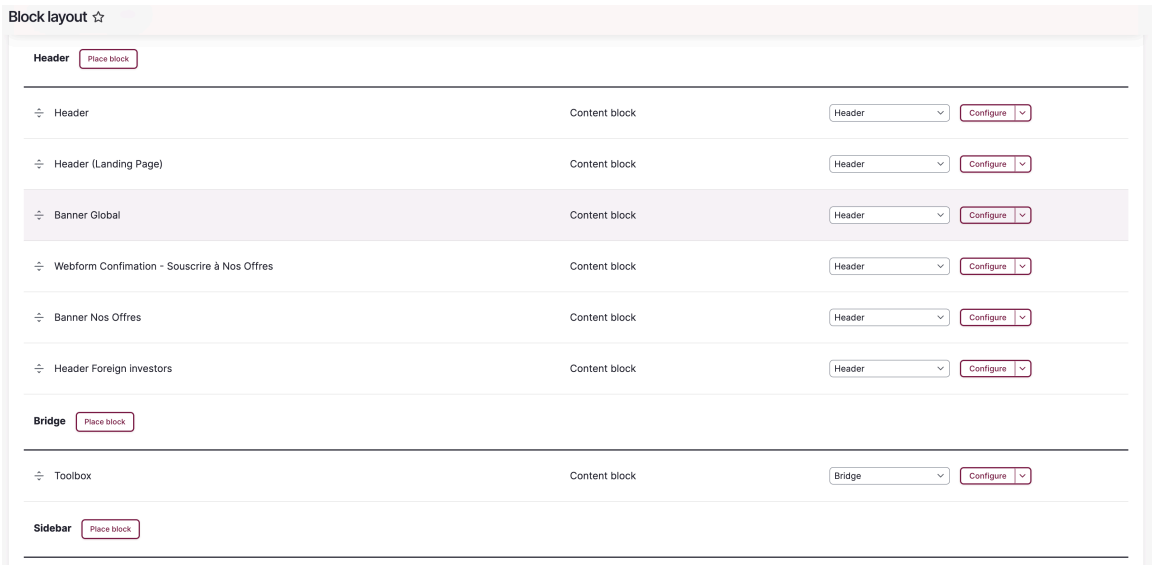

Figure 3.8: Visual placement of widgets (blocks) in the Drupal back office

This decoupled approach allows for rapid content updates and consistent data delivery to the frontend, while maintaining a clear separation between content management and presentation.

3.5.3 Performance Optimization with Memcached

To enhance backend response time and reduce database load, we implemented Memcached as our caching solution. In our Docker-based architecture, Memcached runs as a dedicated container, communicating directly with Drupal through our internal Docker network.

Memcached improves our platform's performance by:

- Caching frequently accessed data in memory (entities, views, routes)
- Reducing database queries and disk I/O operations
- Providing near-instant access to cached content
- Enabling horizontal scaling of the caching layer

This caching strategy is particularly effective for our recruitment platform, where multiple users frequently access the same content like job listings and candidate profiles. During peak recruitment periods, Memcached helps maintain fast response times even under high traffic loads.

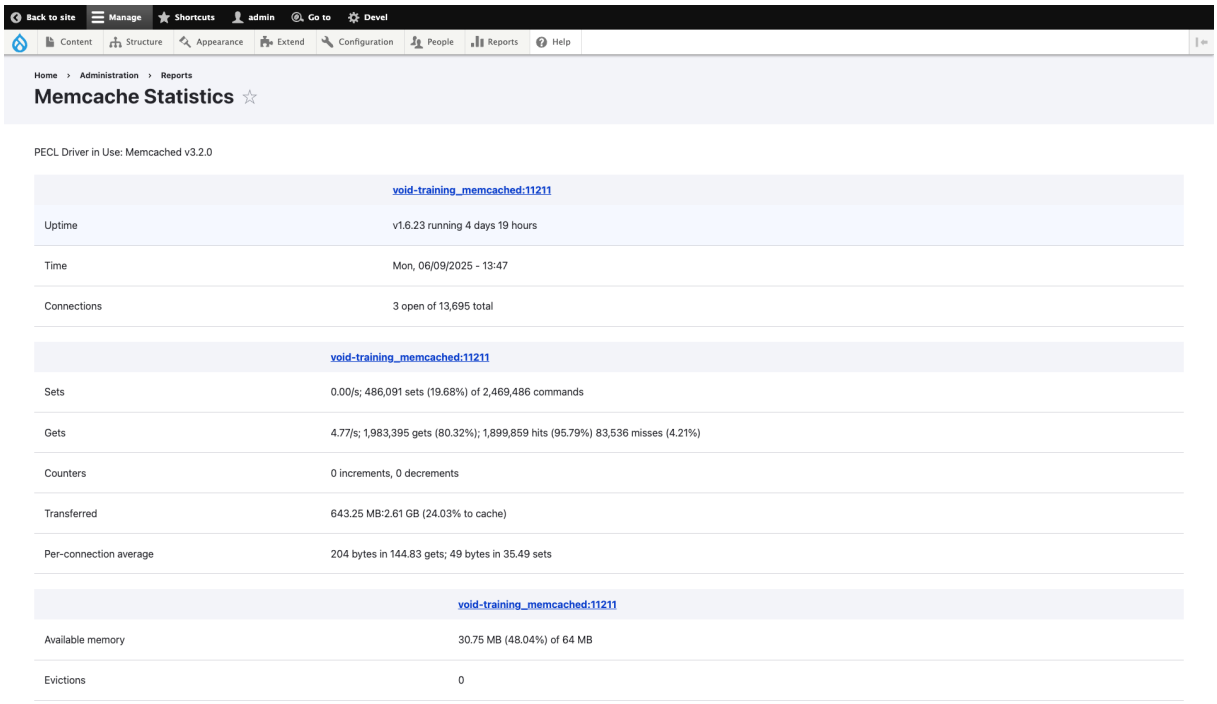

Figure 3.9: Memcached UI

3.5.4 Email Debugging with MailHog

We use MailHog as a Docker container for email testing in development. It provides a web interface at port 8025 for viewing emails and an SMTP server at port 1025 for intercepting outgoing messages.

Drupal is configured to use MailHog's SMTP server through the Symfony Mailer module, which handles all system emails including account notifications, interview scheduling, evaluation reports, and password resets. This containerized setup ensures consistent email testing across all development environments.

3.5.5 Widget System Architecture and Discovery Process

The backbone of our content management strategy relies on Vactory's widget system, which implements dynamic, reusable content components that bridge backend configuration with frontend rendering. Understanding the widget lifecycle reveals how Drupal automatically discovers, categorizes, and exposes these components to content editors.

The widget discovery process follows a structured technical workflow:

1. **Widget Definition**: Each widget consists of a directory containing a `settings.yml` file that defines metadata, JSON:API queries, field structures, and categorization

information. This YAML configuration acts as the widget's blueprint, specifying data sources, field types, and display preferences.

2. **Automatic Discovery**: During Drupal's module discovery phase, the system scans all enabled modules for widget directories. The presence of a `settings.yml` file signals to Drupal that the directory contains a valid widget definition.

3. **Plugin Registration**: Drupal's plugin system registers each discovered widget as a paragraph type, making it available through the administrative interface. The widget's metadata from the YAML file determines its category, id, and content.

4. **Frontend Mapping**: A corresponding Webpack plugin on the frontend side scans for widget files matching specific patterns (`*Widget.jsx`). This plugin extracts configuration IDs from each React component and generates a mapping file that links backend widget IDs to their frontend implementations.

5. **Template Selection Interface**: Content editors access widgets through Drupal's paragraph interface, where widgets are organized by categories (Academy, Banners, Content, Void Training, etc.). This categorization is defined in each widget's YAML configuration.

When content editors create or edit pages, they encounter a streamlined widget selection process. The template chooser displays widgets organized by functional categories, allowing editors to quickly locate appropriate components for their content needs.

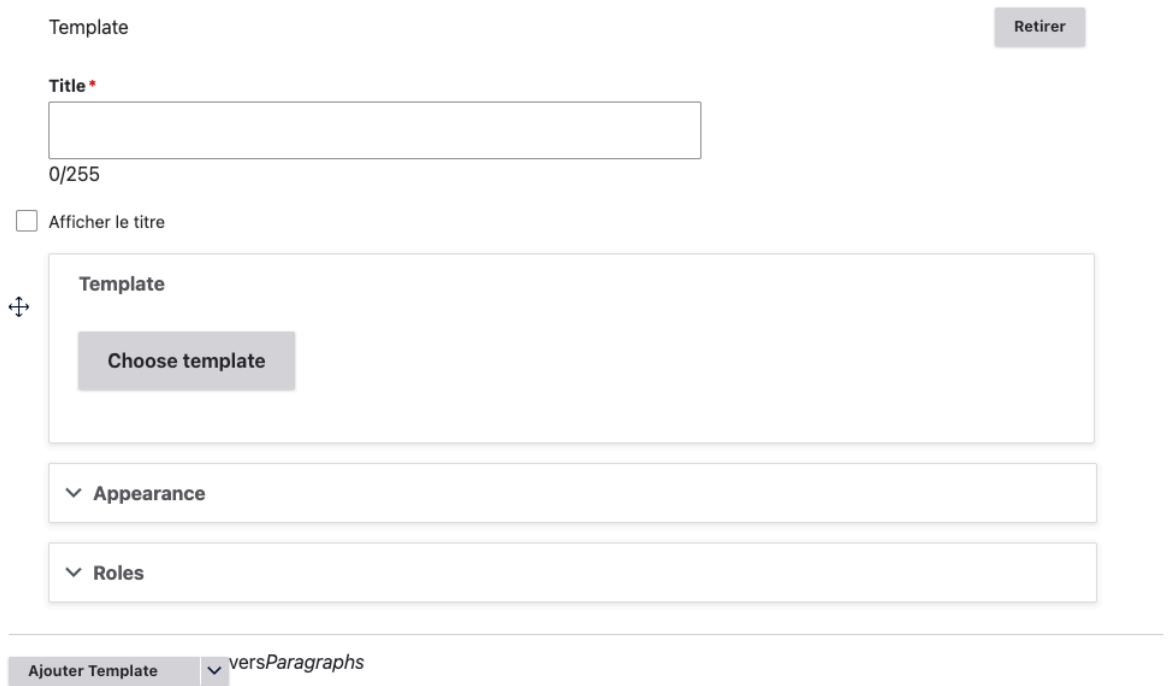

Figure 3.10: Widget Selection Interface with Category Tabs

Within each category, editors can browse available widget templates. For the Void Training project, custom widgets were developed specifically for recruitment and training content, including process flows, call-to-action sections, and testimonial displays.

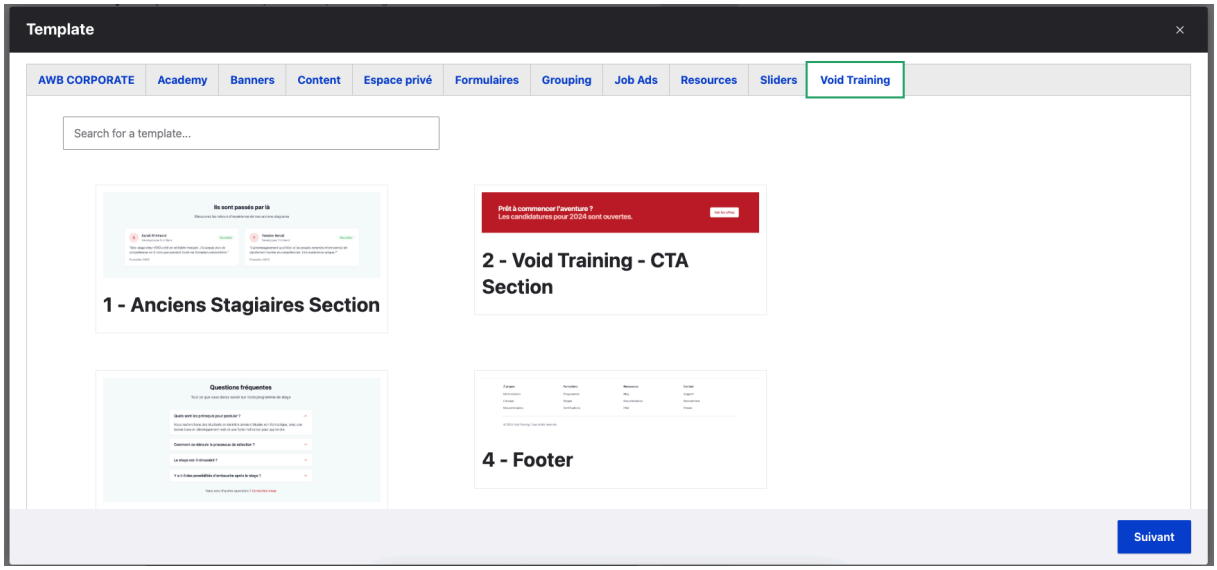

Figure 3.11: Available Widget Templates in Void Training Category

Once a widget is selected, editors access a dynamic configuration interface. Each widget exposes different field types and structures based on its YAML definition. For complex widgets like the Recruitment Process widget, editors can configure multiple components, titles, descriptions, and process steps through an intuitive form interface.

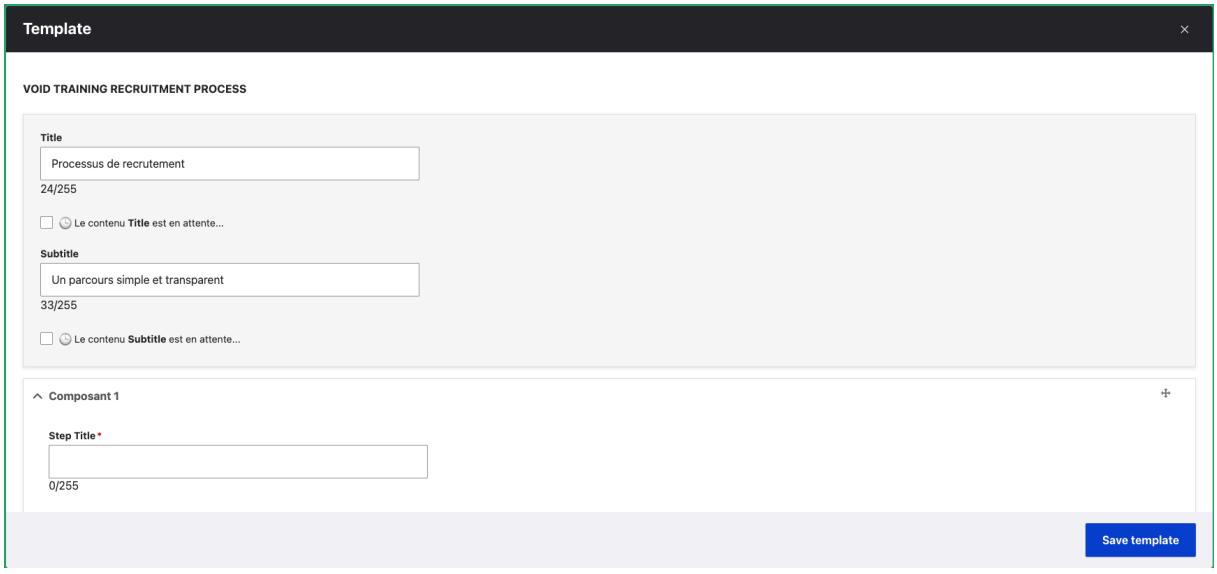

Figure 3.12: Widget Configuration Interface for Recruitment Process

3.5.6 Main Website Backend Structure

The main website's backend is built using a widget-based architecture, where each widget is a Drupal paragraph type that can be configured and reused across the site. These widgets are exposed through JSON:API for frontend consumption, making them easily accessible and maintainable.

The platform includes two main categories of widgets: Core Layout Widgets and Content Display Widgets. Core Layout Widgets handle the fundamental structure of the site. The Header Widget manages navigation and user authentication, while the Footer Widget provides company information and quick links. The Hero Widget creates engaging landing sections with customizable backgrounds and call-to-action elements.

Content Display Widgets focus on presenting information in various formats. Card Listing Widgets offer three variations: Tagged Cards for filtered content, Duration Cards for time-based content, and Numbered Cards for sequential information. The Testimonial Widget displays user feedback with ratings and profile information, while the Slider Widget creates dynamic content carousels with responsive controls.

A key feature is the Project Showcase Widget, which displays intern projects with details like technologies used, duration, and links to repositories. All widgets are designed to be fully configurable through the Drupal interface, responsive across devices, and consistent in their styling and behavior.

This widget-based approach allows content editors to create engaging layouts without technical intervention while maintaining a consistent presentation across the site.

3.5.7 Training and Resource Management Custom Modules

The platform's training and development capabilities are powered by three specialized custom modules that work together to create a comprehensive learning environment. These modules handle everything from sprint-based progression to resource management and project tracking.

The **void_training_academy** module forms the foundation of the training system. It implements a sprint-based progression framework that manages the relationship between supervisors and interns. The module tracks learning objectives and handles deliverable submissions, making it adaptable for various training contexts from internship programs to team workshops.

Project management is handled by the **void_training_project** module. This module oversees the final capstone projects assigned to interns after their training phases. It manages project assignments, tracks progress, and handles final deliverables. A key feature is its ability to link projects to specific technologies and specialties through taxonomy terms.

The **void_training_resources** module serves as the platform's knowledge repository. It provides both sprint-specific and global learning resources, supporting various

content formats. Resources can be tagged with relevant technologies and made available either globally or attached to specific training phases.

The platform implements a structured progression system with three main phases:

Sprint (Academy) Phase Supervisors create and configure training sprints with specific objectives, attach relevant learning resources, and monitor intern progress. Interns access their assigned sprints, track progress by marking completed objectives, and submit deliverables. The system automatically tracks completion status and provides real-time progress updates.

Project Phase During this phase, supervisors create and assign capstone projects, define requirements, and evaluate submissions. Interns work on their assigned projects, submit deliverables, and access project-specific resources. The system maintains clear communication channels between supervisors and interns throughout the project lifecycle.

Resource Management The platform provides a comprehensive resource management system where supervisors can upload and organize learning materials. Resources are categorized by technology or topic and can be made available globally or restricted to specific sprints. Interns can easily access, search, and download relevant materials based on their current training phase.

The entire process is automated through the platform's API layer, which handles:

- Real-time progress tracking and updates
- Automated project assignment upon sprint completion
- Resource access control and delivery
- Deliverable submission and management

This structured approach ensures a consistent training experience while allowing supervisors to customize learning paths based on individual intern needs and project requirements.

Progress Tracking with Objectives Taxonomy

The platform implements a sophisticated progress tracking system using taxonomy terms to manage learning objectives within sprints. This system provides granular control over intern progress and ensures comprehensive skill development.

Objectives Taxonomy Structure

- **Hierarchical Organization**:
  - Primary objectives (e.g.,"React Components", "State Management", "Context API")
- **Progress Tracking**:

  - Each objective is associated with a completion status
  - Progress is tracked at both individual and sprint levels
  - Automated calculation of overall completion percentage
  - Visual progress indicators in the frontend interface

- **Objective Management**:
  - Supervisors can create and modify objectives
  - Objectives can be reused across different sprints

Implementation Details

- **Progress Tracking**:
  - Entity reference field linking objectives to sprints
  - Custom field for tracking completion status
  - Timestamp tracking for objective completion
- **Automation Features**:
  - Automatic progress calculation
  - Sprint completion triggers

Integration with Sprint System

- **Sprint Configuration**:
  - Objectives are assigned to specific sprints
  - Each sprint has a defined set of required objectives
- **Intern Progress**:
  - Real-time progress updates
  - Visual progress indicators
  - Achievement tracking

This taxonomy-based approach to progress tracking ensures:

- Structured learning progression
- Clear success criteria
- Measurable outcomes
- Flexible adaptation to different learning paths
- Comprehensive progress monitoring

3.5.8 Security and Access Control

Simple OAuth was used for secure JWT-based token issuance. Private claims were modified to include role and user-specific fields.

AccessControlHandler class restricts access based on role, entity ownership, and operation. JSON:API collection access was filtered using:

Filtering Collection per Connected User:

```
function void_training_json_api_collection_alter(...) {
  // Custom logic to expose only relevant records per token
}
```

Role-Based Access Control and Permissions To maintain security and content workflow integrity, user roles and permissions were carefully structured within the backend. Two primary roles were defined:

- Intern: This role is automatically assigned to candidates once their CVs are approved and accounts are programmatically created. These users can authenticate via the frontend and are allowed to view, update, and track their own candidature entity. They have access to their personal dashboard, application status, and can submit required documents.
- Supervisor: This role includes advanced permissions to manage job offers, moderate content, review applications, evaluate technical tests and challenges, and update candidature status. Supervisors have access to content moderation workflows and administrative views, allowing them to oversee the entire internship process and make informed decisions about candidate selection.

3.6 Frontend Implementation

The Candidate Platform's frontend is built on a decoupled architecture, using Drupal 10 as a headless CMS and Next.js 13+ for the user interface Newman [2015]. This setup separates content management from presentation, making the system more flexible and maintainable. The integration relies on JSON:API Buytaert [2019], which exposes all backend data as RESTful endpoints, so there's no need to write custom APIs for every entity or field.

3.6.1 Decoupled Architecture and Backend Integration

The frontend consumes data from Drupal's JSON:API endpoints. Each content type, taxonomy, user, and webform is available as a resource. This approach keeps the backend and frontend loosely coupled, allowing each to evolve independently. The main benefits: faster page loads, easier updates, and a clear separation of concerns.

JSON:API Integration

All CRUD operations and relationships are handled through JSON:API. Filtering, sorting, and pagination are built in. For example, fetching a candidate's application and related quiz results or challenge submissions is a single request.

Widget-Based Content Delivery

A central innovation in the platform is the widget system, which bridges Drupal's content management capabilities with Next.js's component-based rendering. Each widget encapsulates a reusable content component, configurable via Drupal's administrative interface and rendered dynamically on the frontend.

The widget architecture comprises three principal elements:

1. **Backend Widget Configuration**: Widgets are defined in Drupal using `settings.yml` files, specifying JSON:API queries, field filters, and sorting logic.
2. **Frontend Widget Components**: React components that receive and render data provided by the backend.
3. **Webpack Build-Time Mapping**: An automated system that maps backend widget configurations to their corresponding frontend components during the build process and inject the data into the components.

Webpack Plugin Architecture

The mapping between backend widget configurations and frontend components is orchestrated by a custom Webpack plugin. This plugin automates the discovery and registration of widget components, ensuring seamless integration and maintainability. The core responsibilities of the `WidgetsPlugin` include:

1. **Component Discovery**: Scanning the codebase for files matching the `*Widget.jsx` and `*Node.jsx` pattern.
2. **Configuration Extraction**: Parsing each component's `config.id` to establish unique identifiers.
3. **Dynamic Import Generation**: Supporting both lazy and direct imports for optimal performance.
4. **Mapping File Creation**: Generating a mapping file (`widgets.js`) that links widget IDs to their respective components.

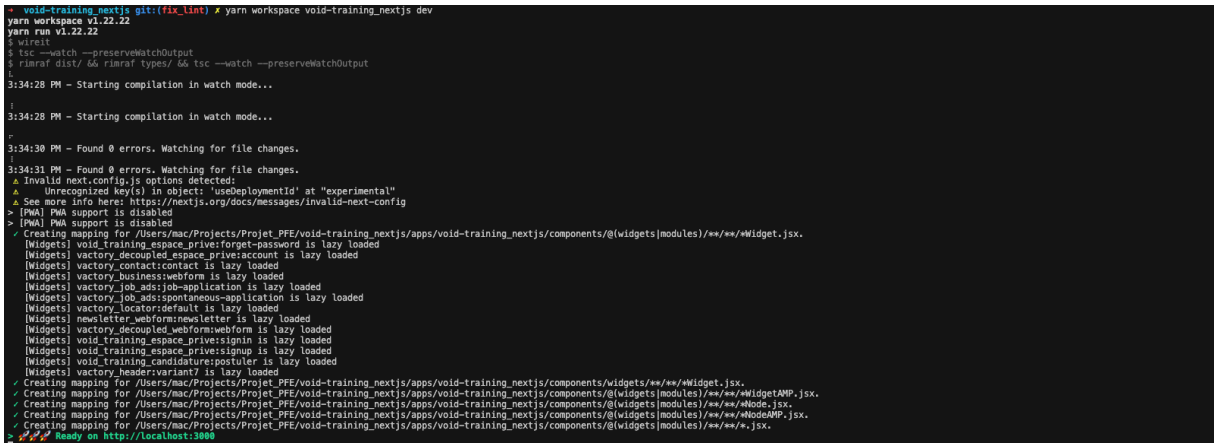

Figure 3.13: Webpack Mapping Process

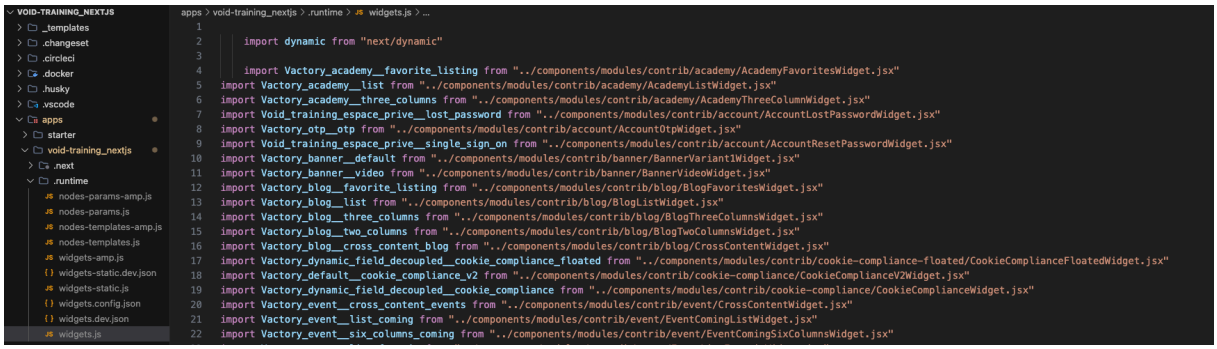

Figure 3.14: Webpack Listing File

3.6.2 Routing and Path Resolution

Slug-Based Routing

The frontend employs a catch-all route pattern (`[...slug].jsx`) to intercept and process all navigation requests. The routing workflow is as follows:

1. **Path Translation**: The frontend transmits the requested path to Drupal's translate route endpoint.
2. **UUID Resolution**: The Pathauto module resolves the alias to the corresponding content UUID.
3. **Content Fetching**: The UUID is used to retrieve the full content entity via JSON:API.
4. **Component Rendering**: The retrieved content is passed to the appropriate page template for rendering.
5. **UUID Caching**: The UUID is cached in redis to avoid unnecessary API calls.

API Proxy Implementation

To further abstract backend infrastructure and enhance security, all API requests from the frontend are routed through a Next.js API proxy. This proxy is responsible for authentication, caching, and forwarding requests, thereby maintaining a clear separation of concerns.

Key features of the proxy include:

- **URL Rewriting**: Requests to `/api/proxy/*` are transparently forwarded to the Drupal backend.
- **Response Caching**: GET requests are cached using Redis to improve performance and reduce backend load.
- **Header Management**: Authentication and other headers are managed and forwarded as required.
- **Error Handling**: Robust error handling ensures appropriate HTTP status codes and user feedback.

3.6.3 Monorepo Architecture and Starter Kit Integration

The project is structured as a monorepo, promoting code reuse and consistency across platform components. The monorepo is anchored by the Vactory starter kit, which provides foundational components and utilities.

Workspace Organization

The monorepo is organized into distinct workspaces, each serving a specific function:

1. **Vactory Starter Kit**: Contains core components, utilities, and design system elements.
2. **Project Workspace**: Houses custom components and modules specific to the intern management platform.
3. **Shared Dependencies**: Centralized configuration for linting, TypeScript, and other shared tooling.

A schematic of the directory structure is provided below:

```
[caption={Monorepo Directory Structure}, captionpos=b]
void-training_nextjs/
 packages/
```

```
    core/
    console/
apps/
    starter/ # Vactory starter kit that is synchronized with the Vactory starter kit
    void-training_nextjs/
        .next/
        .storybook/
        hooks/
        lib/
        pages/
        public/
        components/
            elements/           # Atomic components
            widgets/            # Content widgets
            modules/            # Complex modules
        pages/
        public/
shared/
    eslint-config/
    typescript-config/
```

3.6.4 Design System and Storybook Integration

Storybook is central to our design system. It lets us build and test UI components in isolation, document usage, and share previews with stakeholders. Each component has stories showing its different states and props. The Button component, for example, is documented with all its variants and sizes.

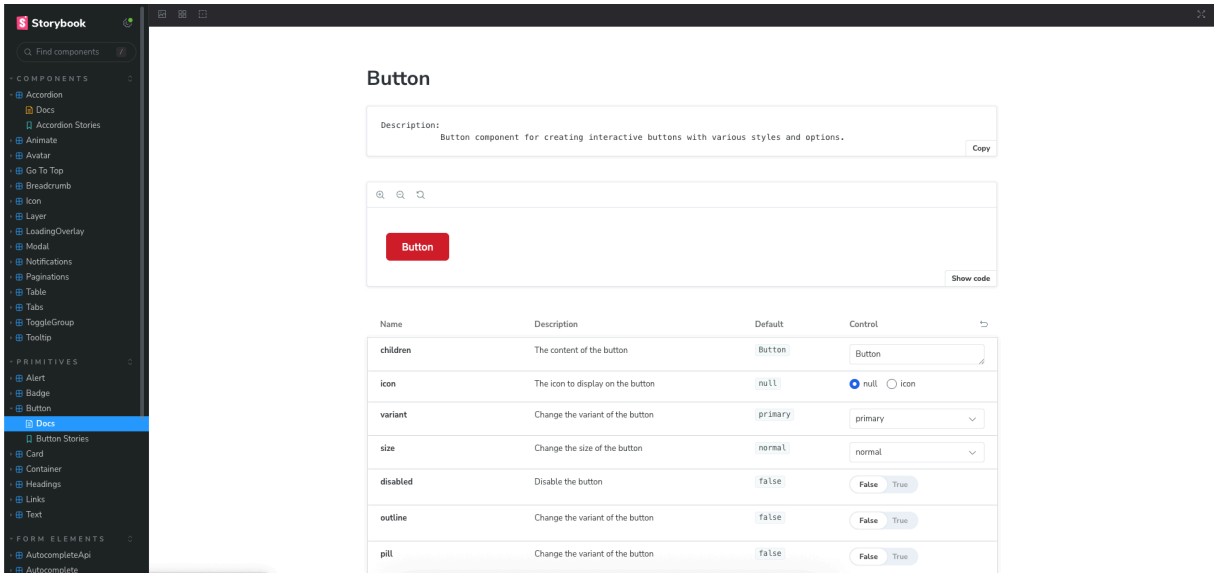

Figure 3.15: Storybook Button Component

3.6.5 Icon System and Asset Management

Icons are managed as an SVG sprite, generated at build time. This keeps asset delivery efficient and usage flexible. The build script optimizes SVGs, tidies up the code, and aggregates everything into a single sprite in the public directory.

```
{
  "scripts": {
    "icons:build": "npx --yes @vactorynext/svg-spreact-cli
       ./icons --optimize true --tidy true > ./public/icons.svg"
  }
}
```

Listing 3.1: Icon Build Script Configuration

This process encompasses:

- **SVG Optimization**: Removal of extraneous metadata and path optimization.
- **Code Tidying**: Cleanup of SVG markup for minimal file size.
- **Sprite Generation**: Aggregation of all icons into a single sprite file.
- **Public Exposure**: Placement of the sprite in the public directory for direct access.

3.6.6 Custom Widgets and Modules Implementation

Custom widgets and modules power both content presentation and interactive features. The platform implements two main categories of widgets:

Core Layout Widgets

- **Header Widget**: Implements the site's navigation structure with:
  - Main menu integration
  - User authentication status
  - Responsive mobile menu
- **Footer Widget**: Provides site-wide footer content including:
  - Company information
  - Social media links
  - Quick navigation links
- **Hero Widget**: Creates impactful landing sections with:
  - Background image/video support
  - Call-to-action buttons

  - Customizable text content

Content Display Widgets

- **Card Listing Widgets**:
  - **Tagged Cards**: Displays content cards with taxonomy term filters
  - **Duration Cards**: Shows cards with time-based labels (e.g., "2 weeks", "3 months")
  - **Numbered Cards**: Presents content in a numbered sequence
- **Testimonial Widget**: Showcases user feedback with:
  - User profile information
  - Rating system
  - Quote content
  - Optional profile images
- **Slider Widget**: Creates dynamic content carousels with:
  - Auto-play and manual navigation controls
  - Responsive image optimization
  - Custom transition effects
  - Mobile-friendly touch gestures
- **Project Showcase Widget**: Displays past intern projects with:
  - Project title and description
  - Technologies used (with icons)
  - Project duration and completion date
  - Link to project repository or demo
  - Intern's name and role

3.6.7 Performance Optimization and Testing

Performance and reliability are ensured through a combination of optimization techniques and rigorous testing.

Redis Integration for Performance Optimization

Redis and Redis Commander play a crucial role in optimizing the platform's performance through efficient caching mechanisms. Redis serves as an in-memory data structure store, while Redis Commander provides a web interface for managing Redis instances.

The integration of Redis in our project serves several key purposes:

1. **API Response Caching**: Redis caches JSON:API responses from Drupal, significantly reducing response times for frequently accessed data.
2. **Session Management**: User sessions and authentication tokens are stored in Redis, enabling fast retrieval and validation.
3. **Rate Limiting**: Redis helps implement rate limiting for API endpoints, preventing abuse and ensuring system stability.
4. **Real-time Updates**: Redis pub/sub functionality enables real-time updates for features like quiz submissions and status changes.
5. **Offline Mode Support**: Redis caches enable offline functionality when the backend is unavailable.

Redis Commander is utilized for:

1. **Monitoring**: Real-time visualization of Redis operations and memory usage.
2. **Data Management**: Easy inspection and modification of cached data.
3. **Performance Analysis**: Tracking cache hit rates and response times.
4. **Debugging**: Quick access to cached content for troubleshooting.

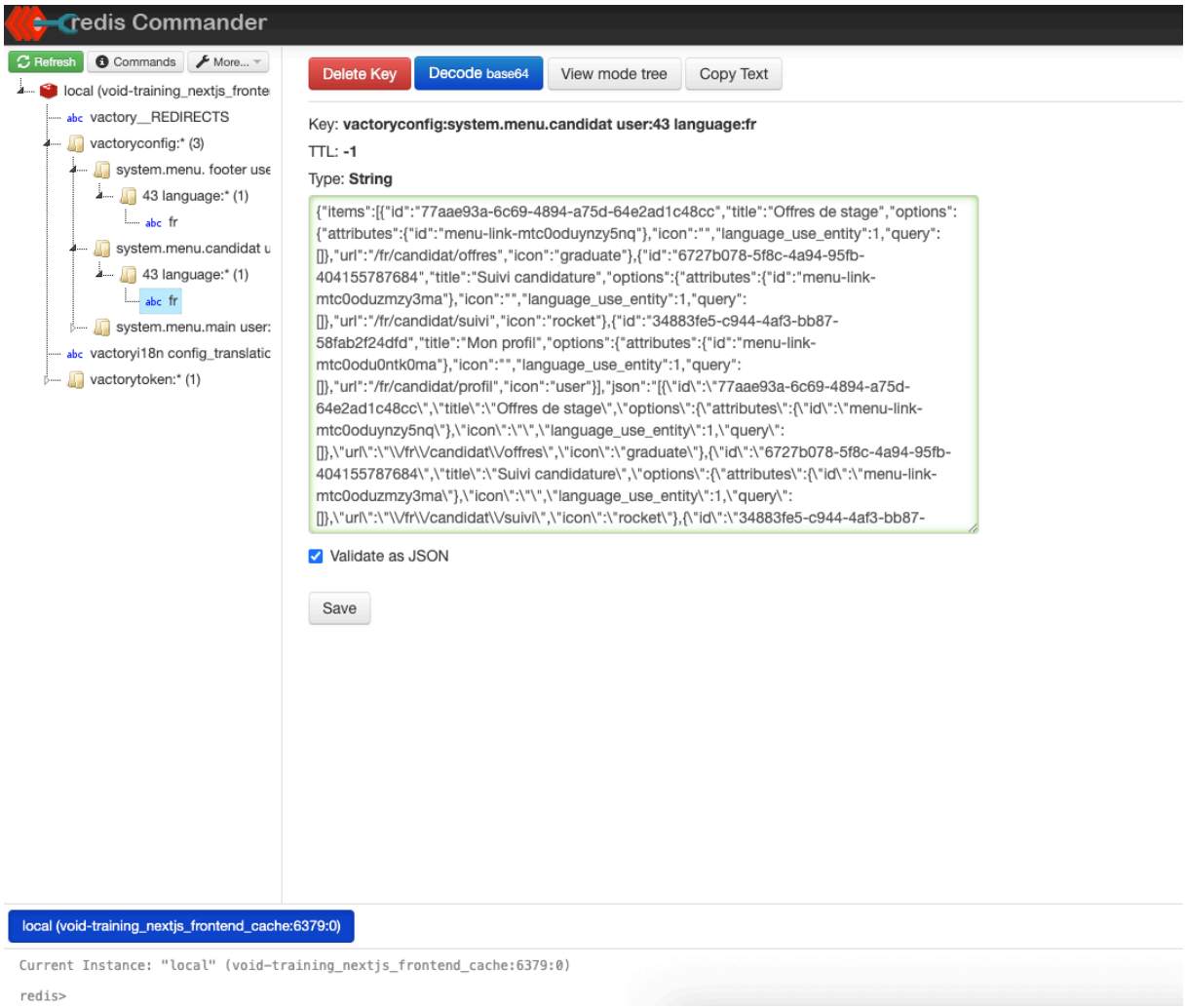

Figure 3.16: Redis Commander UI

Offline Mode Implementation

A critical feature of our platform is the offline mode capability, which ensures users can access content even when the backend is unavailable or there's no internet connection. This functionality leverages Redis caching to maintain platform availability during service interruptions.

When offline mode is enabled, the system operates as follows:

1. **Cache Detection**: The frontend detects when the backend API is unreachable through failed request monitoring.
2. **Redis Fallback**: Instead of displaying error messages, the system automatically switches to serve cached content from Redis.
3. **Content Delivery**: Users continue to access previously cached pages, widgets, and essential platform functionality.

4. **User Notification**: A subtle indicator informs users they're viewing cached content in offline mode.

5. **Automatic Recovery**: When backend connectivity is restored, the system seamlessly transitions back to live data.

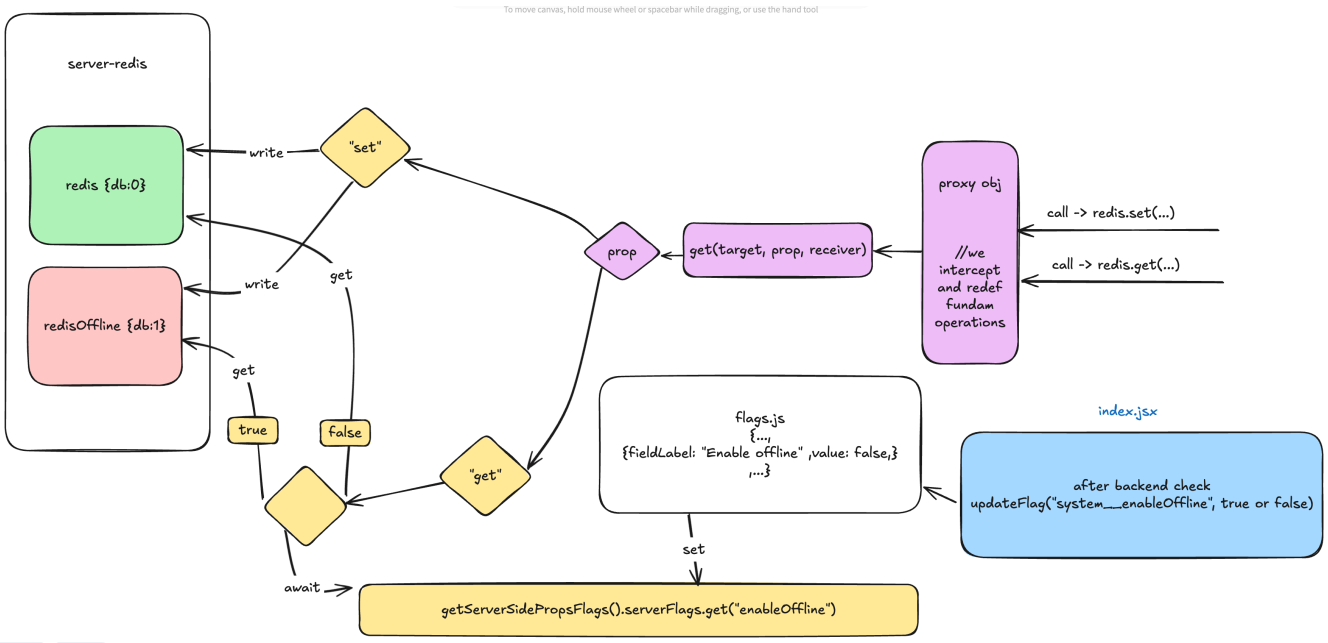

Figure 3.17: Offline Mode Implementation with Redis Cache Schema

This approach ensures business continuity and maintains user experience quality even during technical difficulties or maintenance windows. Candidates can still access their application status, view job offers, and navigate the platform when the backend is temporarily unavailable.

The frontend is designed for maintainability, scalability, and a smooth user experience. Decoupled architecture, component-driven design, and robust state management keep the platform flexible and ready for future growth.

3.7 DevOps and Deployment Strategy

Getting our platform from development to production required setting up a solid DevOps pipeline. We needed something that could handle both our local development needs and production deployment without breaking anything along the way. The setup we built uses containerization for consistency and automated pipelines for reliability.

3.7.1 EasyPanel Infrastructure

We have used EasyPanel running on a dedicated Mac Studio server. This self-hosted approach gives us complete control over our deployment process and keeps costs reasonable

and the Mac Studio provides plenty of power for our needs. The setup handles automatic SSL certificates through Let's Encrypt and Traefik Traefik [2025], container orchestration, and provides a clean interface for managing deployments.

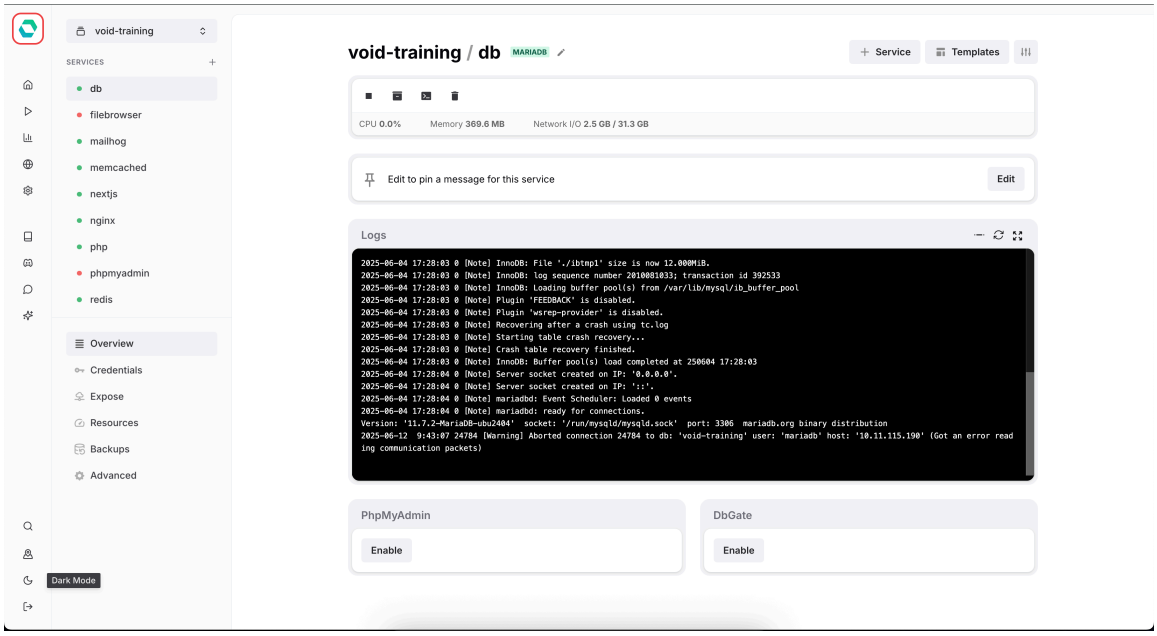

Figure 3.18: EasyPanel Production Environment

3.7.2 Continuous Integration and Deployment Pipeline (CI/CD)

Besides that, we have built a CI/CD pipeline that automates everything from code changes to production deployment. We have used a tag-based deployment system that automatically builds and deploys our code when we created release tags. Here's how it works:

1. We commit code changes to feature branches
2. After code review, changes get merged to main
3. When ready for release, we create a tag (like v1.2.3)
4. Bitbucket webhook triggers a DockerHub build automatically
5. DockerHub builds and tags container images (build fails if ESLint or Prettier tests don't pass)
6. EasyPanel pulls the updated images by tag
7. Rolling deployment happens automatically

Below are the DockerHub build processes for the frontend and backend:

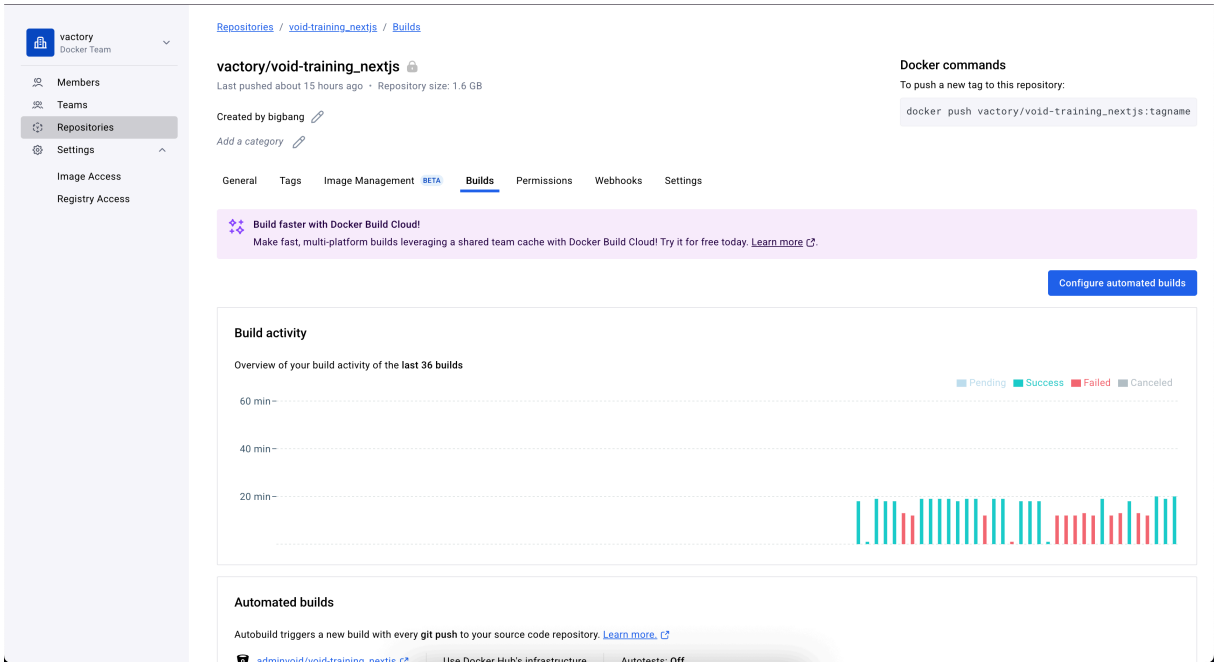

Figure 3.19: DockerHub Frontend Build Process

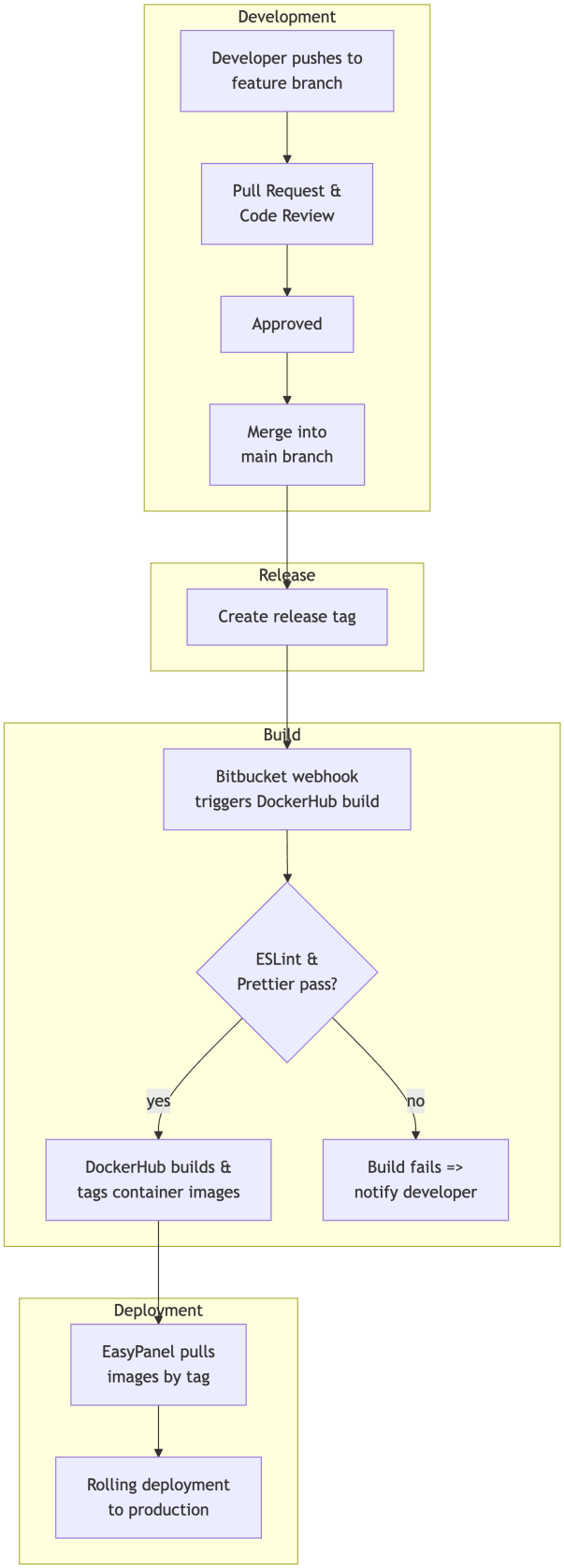

Figure 3.20: CI/CD Pipeline Overview - Complete workflow from code commit to production deployment

3.8 Conclusion

The implementation of the Intern Management Platform at **VOID** represents a comprehensive solution that successfully addresses the complex challenges of managing internship programs in a modern digital environment. Through the adoption of cutting-edge technologies and best practices, the project demonstrates how thoughtful architecture and user-centered design can create powerful, scalable systems.

The platform's modular architecture, built on Drupal 10 and React, provides the flexibility needed to accommodate diverse business requirements while maintaining performance and security standards.

The successful deployment and adoption of the platform within **VOID** validates the effectiveness of the chosen technical approach and implementation strategy. The project not only meets the immediate needs of internship management but also establishes a solid foundation for future enhancements and scalability.

This implementation serves as a testament to the importance of thorough planning, iterative development, and close collaboration between technical and business stakeholders in delivering enterprise-grade solutions that drive organizational efficiency and user satisfaction.

4 Project Realization

4.1 Introduction

The Candidate Platform has been successfully implemented and deployed, providing a comprehensive solution for VOID Digital Agency's intern recruitment process. This chapter presents a detailed walkthrough of the platform's functionality, demonstrating how it streamlines the entire recruitment workflow from job posting to final candidate selection.

4.2 Project Results and Showcase

4.2.1 Main Page Components

The main page showcases key information through various components:

Figure 4.1: Main Page Cards Display

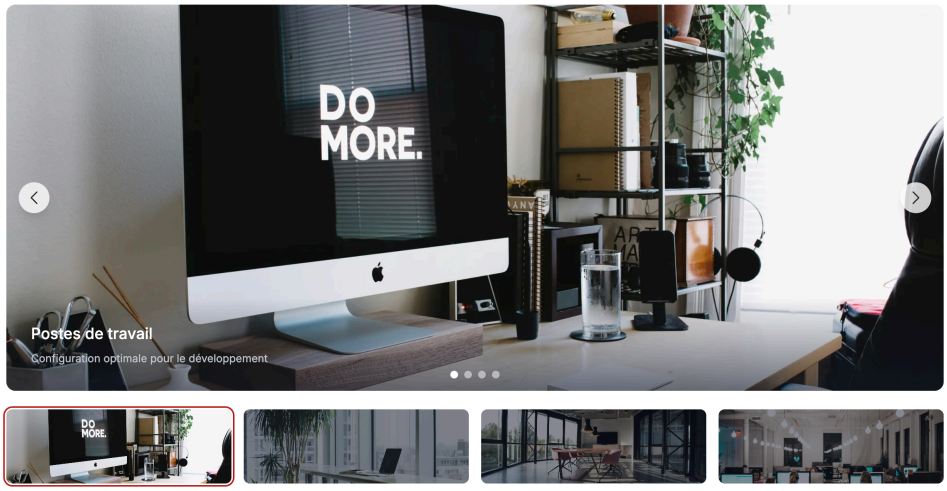

Figure 4.2: Main Page Slider

4.2.2 Intern Dashboard Overview

The intern dashboard provides a comprehensive overview of the internship experience with clear navigation and timeline tracking:

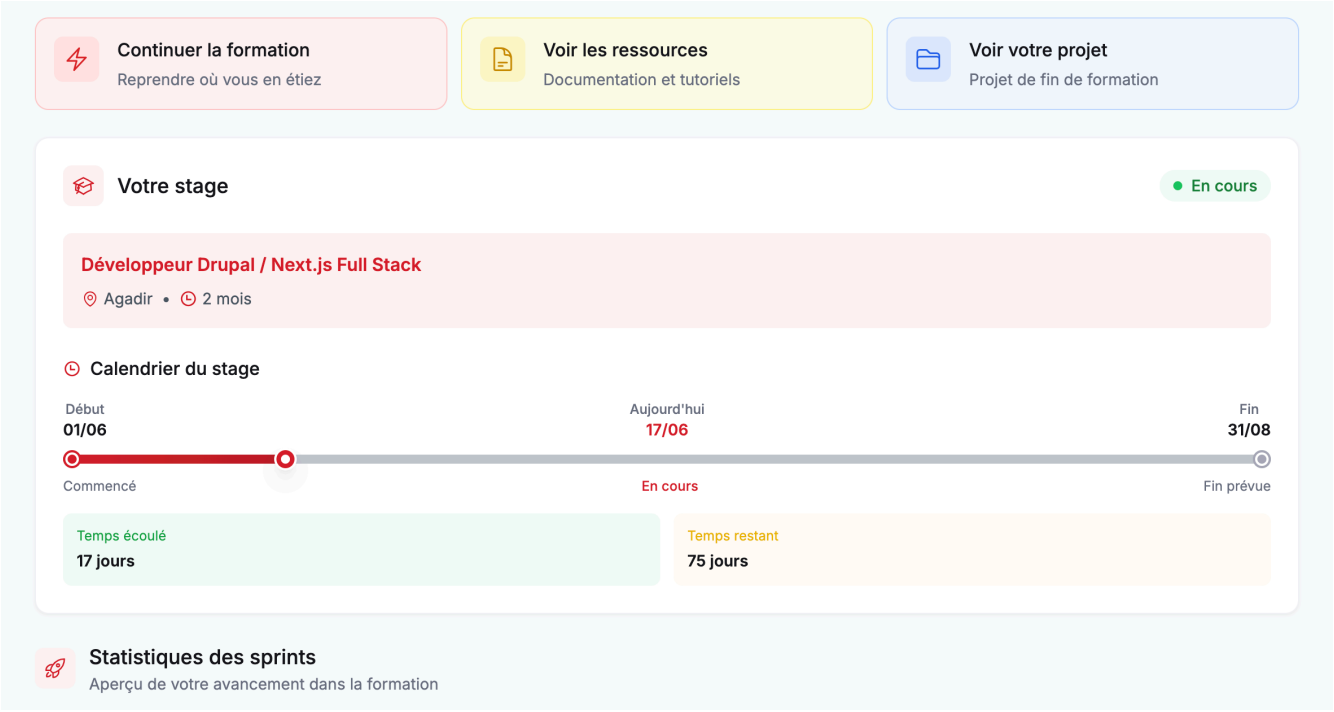

Figure 4.3: Intern Dashboard - Access and Navigation with Timeline

This dashboard illustrates:

- Complete internship timeline and progress

- Easy navigation to all platform features
- Current status and upcoming milestones
- Quick access to assigned tasks and resources

4.2.3 Sprint Statistics Dashboard

A dedicated sprint statistics dashboard provides detailed insights into sprint performance and management:

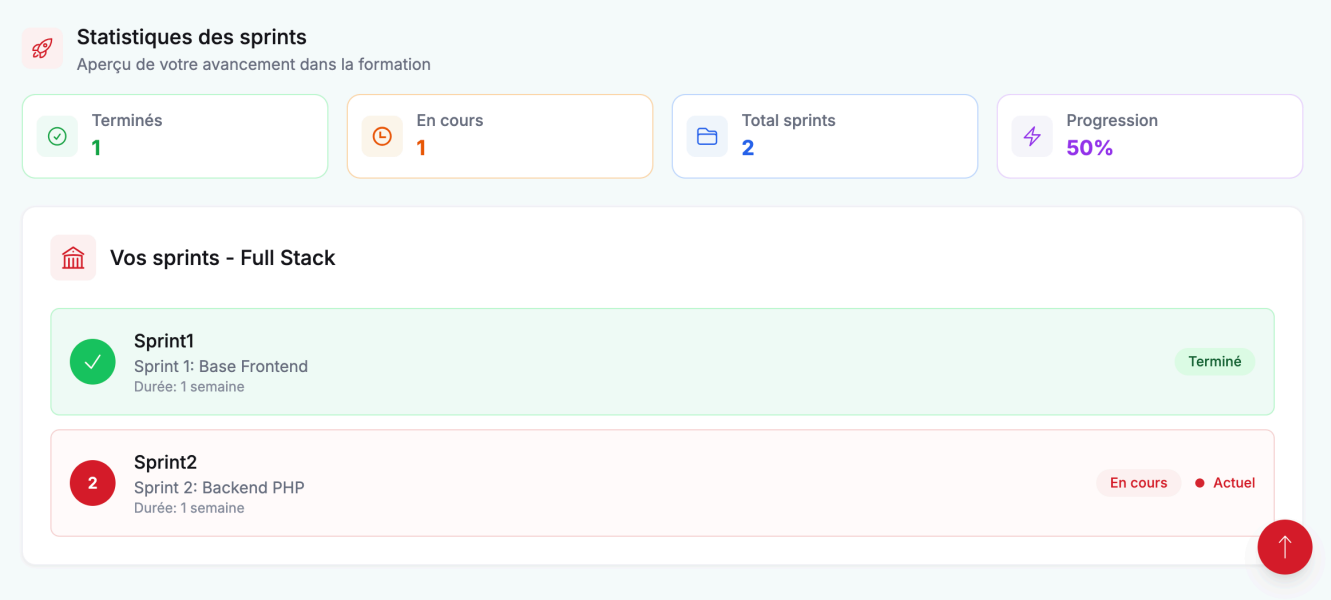

Figure 4.4: Sprint Statistics and Sprint Listings Dashboard

The sprint dashboard displays:

- Comprehensive sprint statistics and metrics
- Small sprint listings with key information
- Progress tracking across all sprints
- Performance analytics and completion rates

4.2.4 Intern Sprint Management

Each intern has access to their assigned sprints with clear progress tracking:

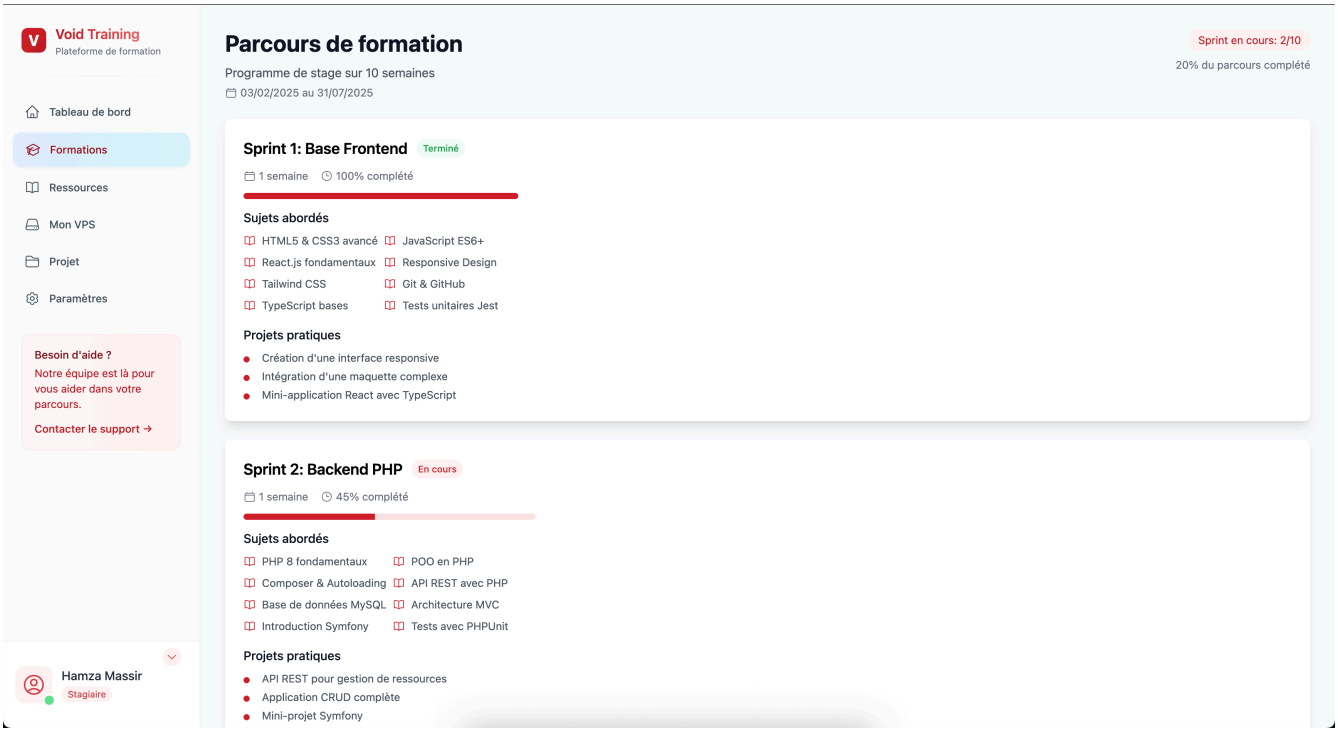

Figure 4.5: Intern Sprint Listing

The sprint listing shows:

- All assigned sprints
- Current progress status
- Start and end dates
- Completion percentage

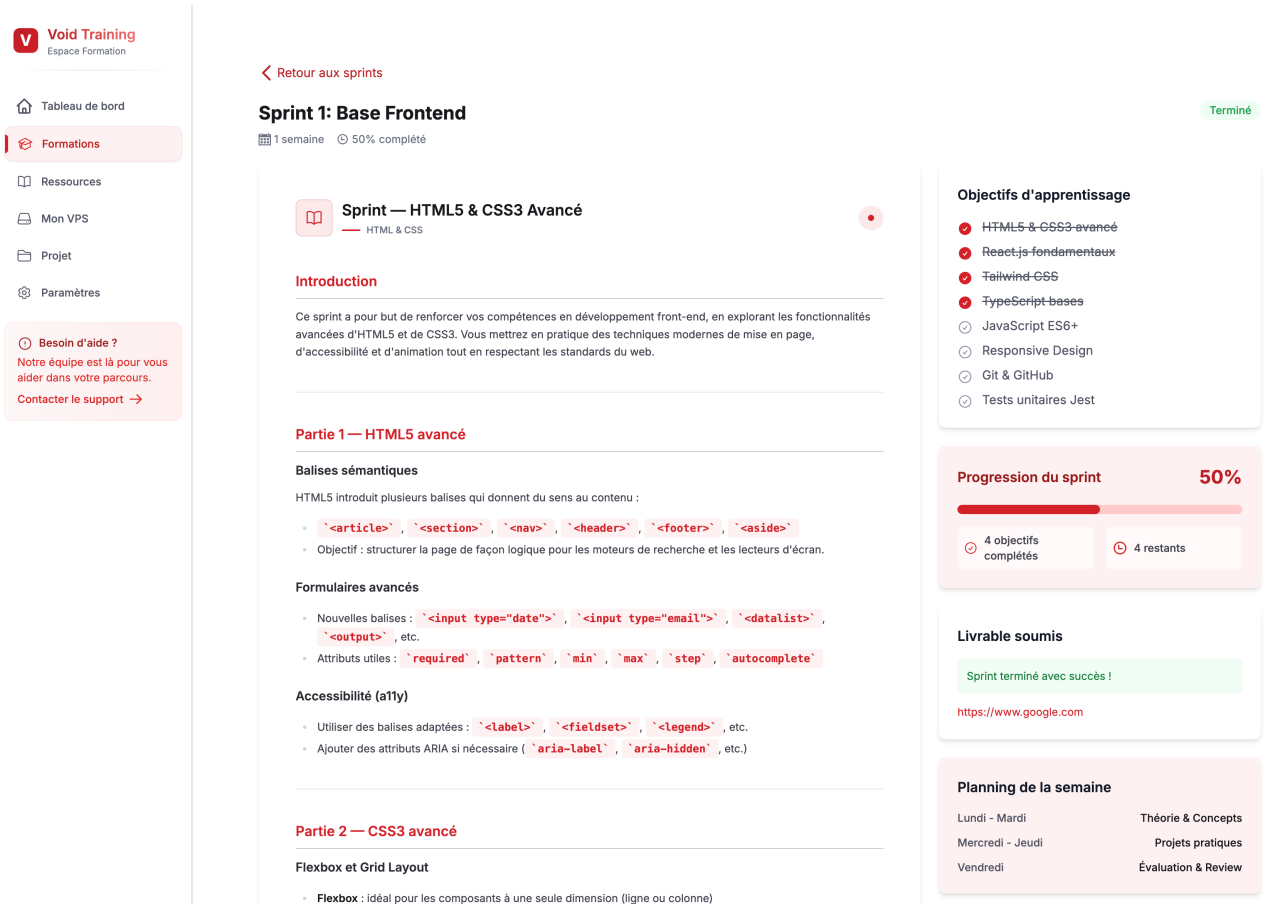

Figure 4.6: Sprint Details Page

Each sprint page includes:

- Learning objectives with progress tracking
- Clear deadlines for each task
- Required resources and materials
- Supervisor feedback section

4.2.5 Learning Resources

A dedicated resources page provides additional learning materials:

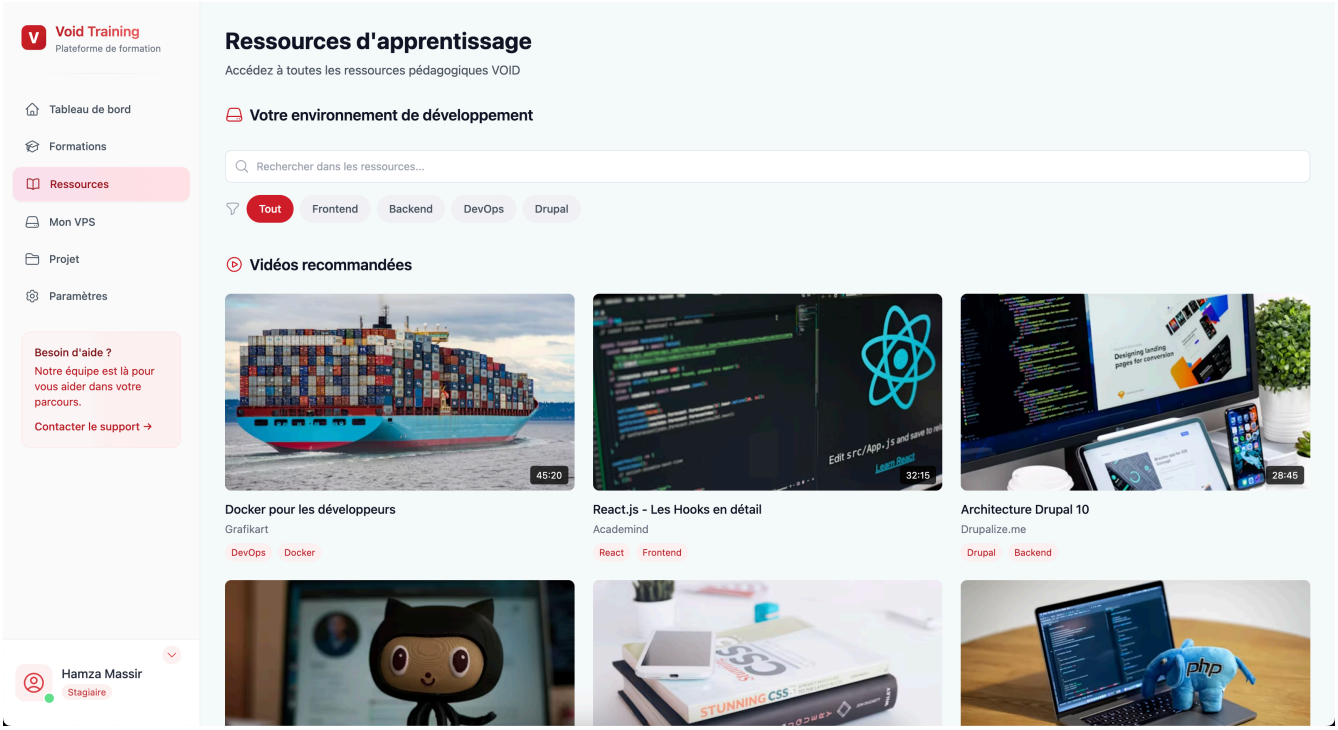

Figure 4.7: Learning Resources Page

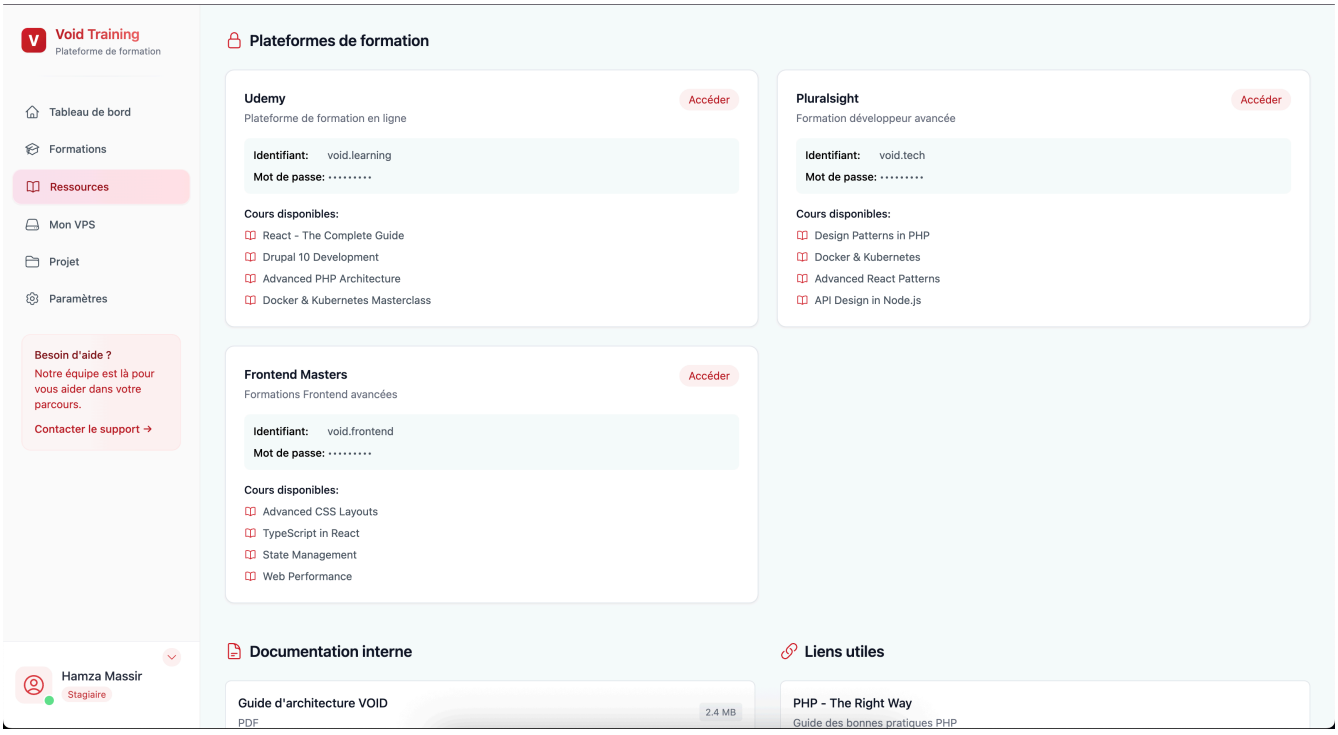

Figure 4.8: Learning Resources Page (Accounts)

The resources section offers:

- Technical documentation

- Tutorial videos
- Code examples
- Best practices guides

4.2.6 Project Management

Each intern has a dedicated project page:

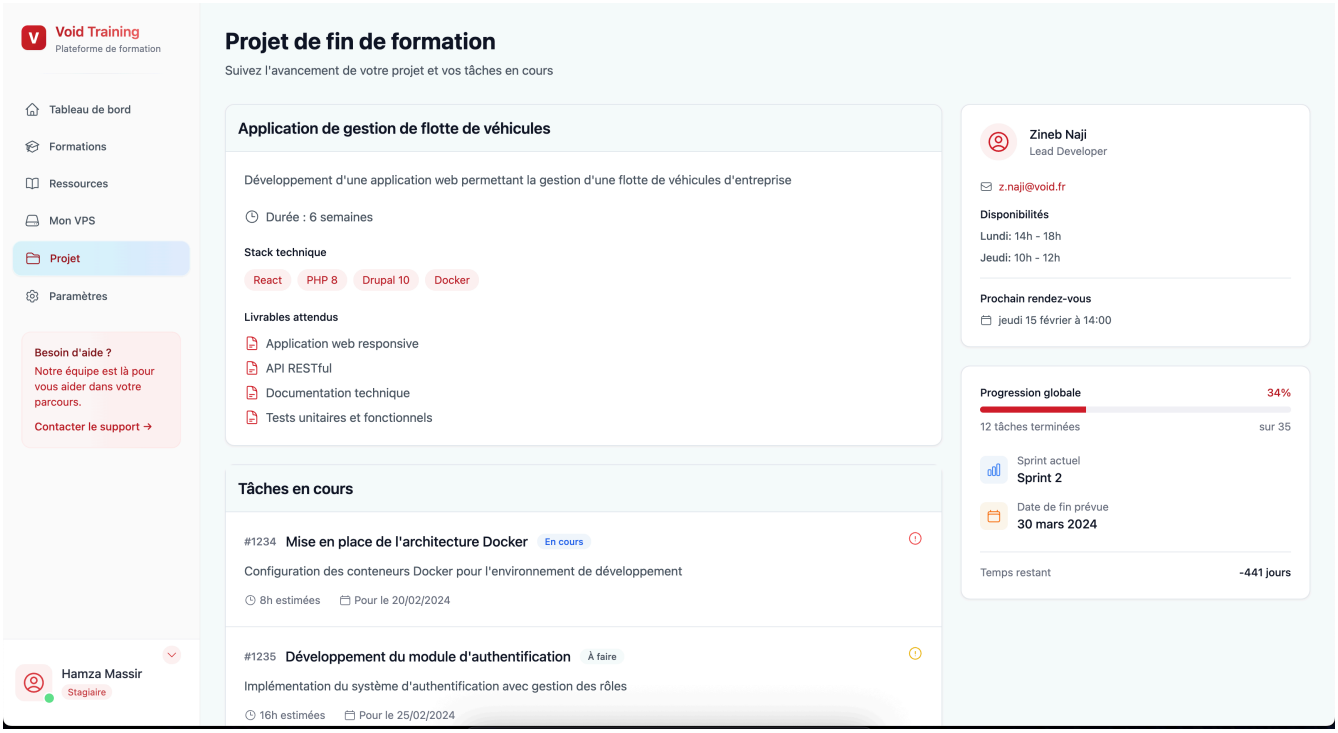

Figure 4.9: Project Management Page

The project page shows:

- Project requirements and goals
- Timeline and milestones
- Technical stack details
- Progress updates
- Supervisor comments

4.2.7 User Profile

Interns can manage their personal information:

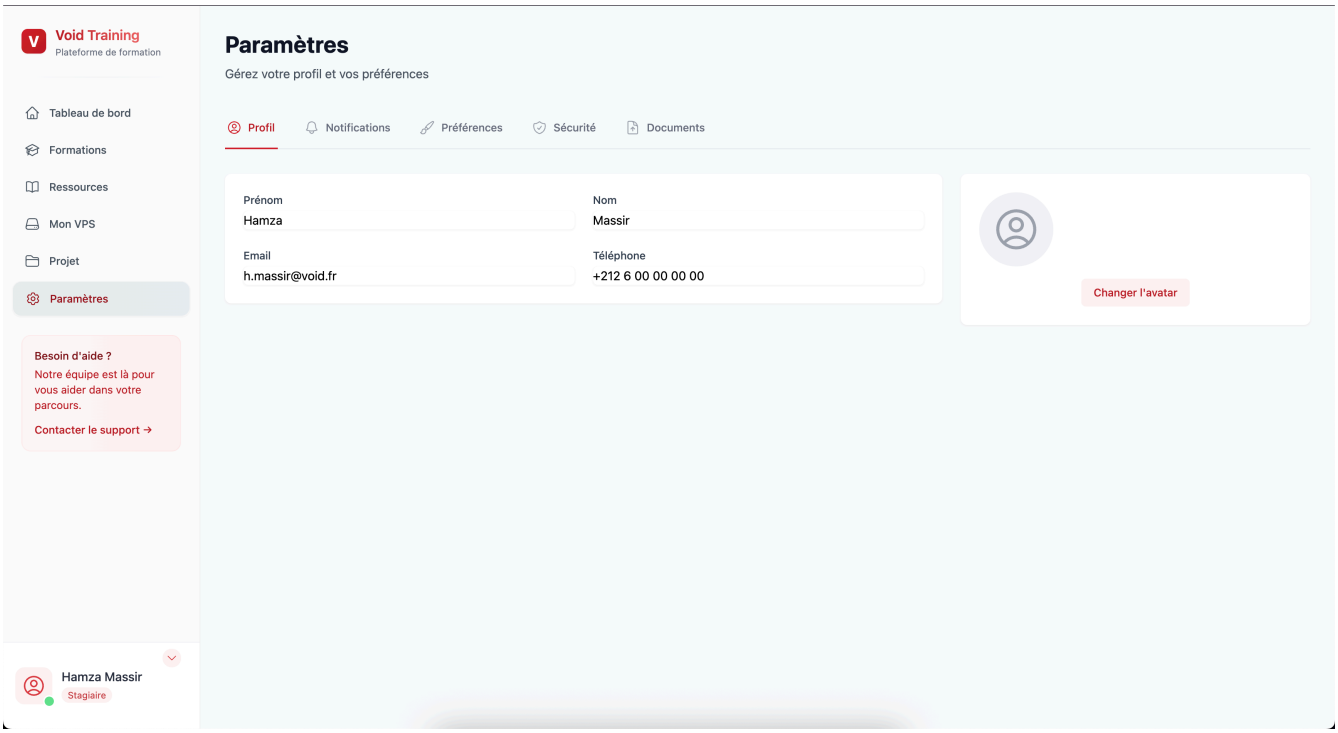

Figure 4.10: User Profile Page

The profile page allows:

- Update personal information
- Change profile picture
- View application history
- Access personal documents

4.2.8 Document Management

A secure space for personal documents:

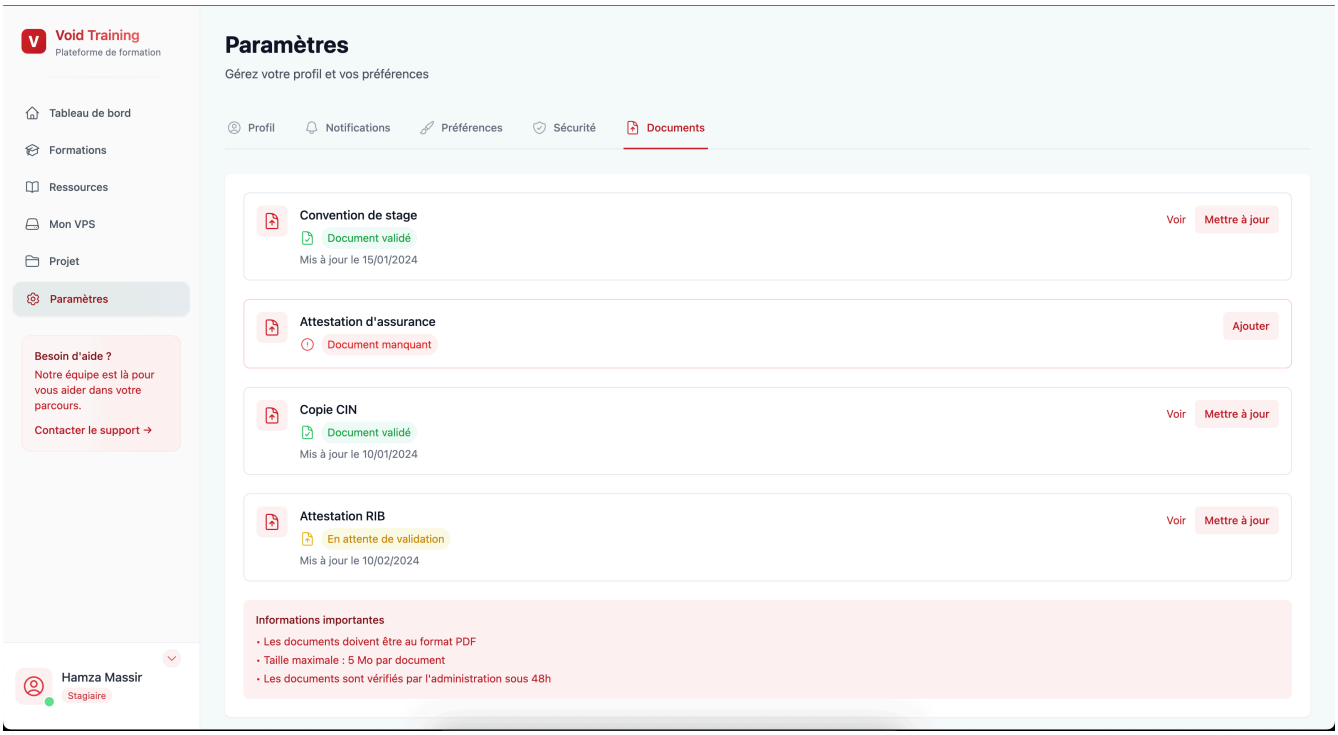

Figure 4.11: Document Management Page

The documents section provides:

- Upload and download files
- View document history
- Share documents with supervisors
- Track document status

4.2.9 Project Deployment

The deployment of the Candidate Platform was streamlined through an automated CI/CD pipeline integrated with Bitbucket. The deployment process is triggered automatically upon each tagged commit, ensuring consistent and reliable releases. The production environment leverages containerized services with Traefik handling SSL certificate management and load balancing, while Nginx serves as a reverse proxy for the frontend application.

Production Architecture Overview

The production environment implements a multi-layered architecture designed for scalability, security, and performance:

- **External User Layer**: Users access the platform through standard HTTP/HTTPS requests via web browsers to the public domain.
- **Edge Router (Traefik)**: Acts as the primary entry point, performing SSL termination using Let's Encrypt certificates and implementing container-based routing rules to direct traffic to appropriate services.
- **Reverse Proxy (Nginx)**: Handles request caching and forwards uncached requests to the Next.js frontend container, optimizing response times for static content.
- **Frontend Application (Next.js)**: Serves the user interface, rendering static pages and managing client-side interactions while communicating with the backend through JSON:API endpoints.
- **Backend API (Drupal)**: Processes business logic, handles authentication, and manages data operations through a headless CMS architecture.
- **Database Layer (MySQL)**: Provides persistent data storage and retrieval for all application data and user information.

Performance Optimization Strategy

To ensure optimal performance and minimal latency, the platform implements a multi-tier caching strategy:

- **Frontend Caching (Redis)**: The frontend container utilizes Redis as a high-performance key-value store for user sessions and temporary application state, reducing authentication overhead.
- **Backend Caching (Memcached)**: The backend container employs Memcached to cache database query results, significantly reducing database load and improving response times for frequently accessed data.
- **Caching Layer Separation**: Redis and Memcached operate independently, serving distinct optimization purposes - frontend user experience optimization versus backend data access optimization.

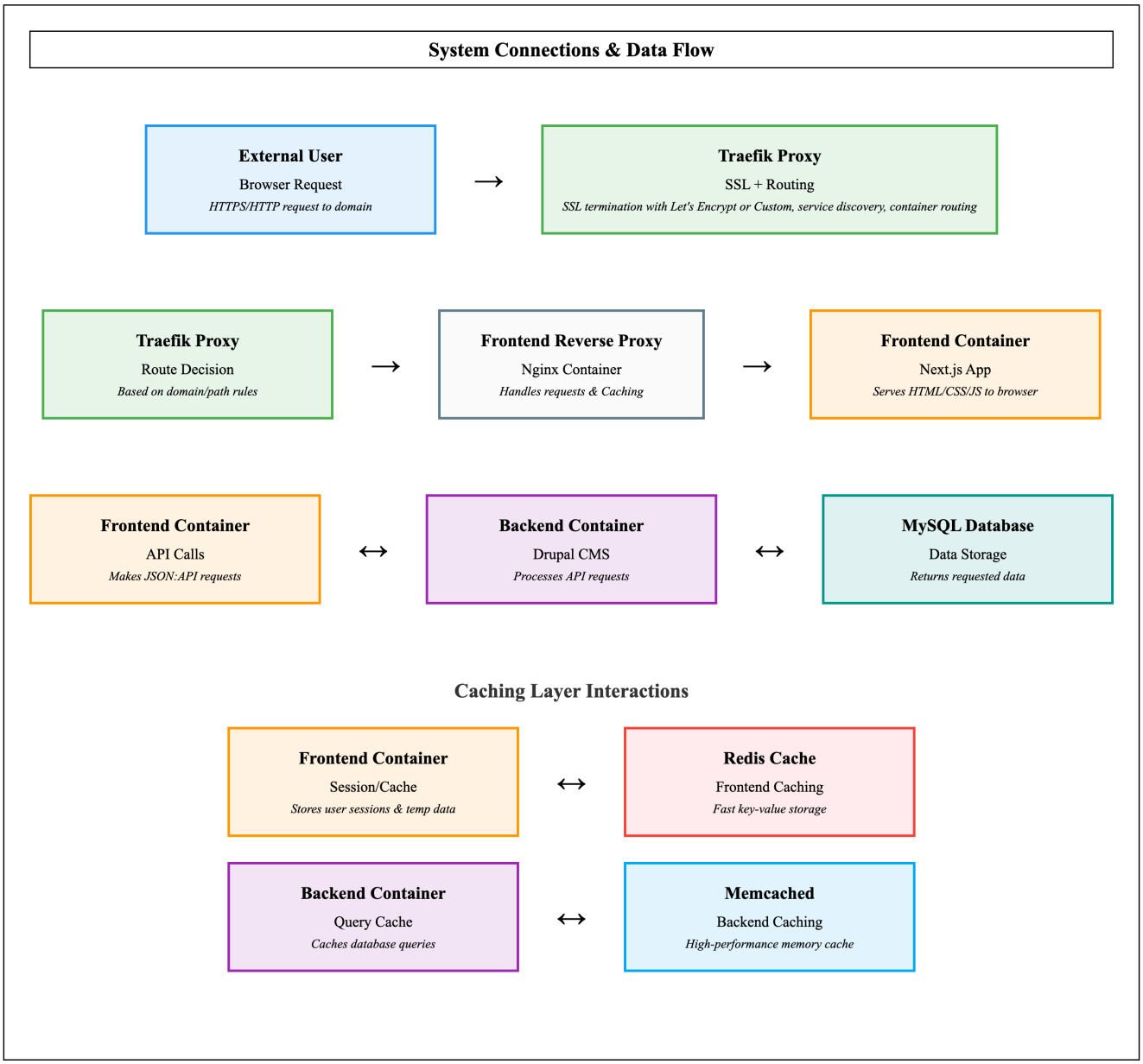

Figure 4.12: Production Environment Architecture

4.2.10 Conclusion

The Candidate Platform has successfully transformed VOID Digital Agency's intern recruitment process into a more efficient, transparent, and scalable system. Through the implementation of automated workflows, structured evaluation processes, and clear communication channels, the platform has significantly enhanced the overall recruitment experience for both candidates and recruiters. The robust production architecture ensures reliable performance, security, and maintainability, providing a solid foundation for future enhancements and scaling requirements.

5 Discussion and Future Improvements

5.1 Introduction

This chapter evaluates the Intern Management Platform developed for VOID Digital Agency. We examine the current results, identify limitations, and propose future improvements. The platform has transformed how VOID manages interns, but there are opportunities for enhancement.

5.2 Current Achievements

The Intern Management Platform has successfully improved VOID's intern management process. The system handles intern onboarding, project assignments, and performance tracking in a centralized manner. The transition from manual processes to an automated system has reduced administrative workload and improved efficiency.

The technical architecture using Drupal 10 backend with Next.js frontend has proven effective. The system handles multiple concurrent users and provides reliable performance. The component-driven development approach has made maintenance easier and ensured consistent user interface across the platform.

User experience has improved significantly. Interns can easily access their project information, track their progress, and communicate with supervisors. The recruitment team reports better organization and reduced time spent on routine administrative tasks.

5.3 Current Limitations

The platform has several areas that need improvement. The administrative dashboard is complex and requires technical knowledge to use effectively. Non-technical staff find it difficult to navigate and perform basic tasks. The interface lacks modern design elements that would improve usability.

The intern workspace has limitations in collaboration features. Interns cannot easily work together on shared tasks or projects. There is no real-time communication system for team collaboration during sprints. The platform lacks tools for group project management and peer-to-peer interaction.

Technical infrastructure needs enhancement. The current system does not provide virtual development environments for interns. This limits their ability to work on complex projects that require specific technical setups. The platform also lacks real-time assistance features that could help interns during their work.

5.4 Future Improvements

5.4.1 VPS Integration for Development Environments

We plan to integrate Virtual Private Server (VPS) technology to provide dedicated development environments for interns. This will allow each intern to have their own isolated workspace with the necessary tools and configurations for their projects. The VPS integration will support multiple programming languages and frameworks, enabling interns to work on diverse technical projects without environment conflicts.

The VPS system will include automated setup and teardown processes. When an intern joins a project, their development environment will be automatically configured with the required dependencies and access permissions. This will reduce setup time and ensure consistency across all intern workstations.

5.4.2 Enhanced Administrative Dashboard

We will develop a new, user-friendly administrative dashboard specifically designed for non-technical users. The new interface will feature simplified navigation with intuitive menu structures. Administrators will have role-based access control, ensuring they only see relevant information and functions.

The dashboard will include visual workflow management tools. Administrators will be able to drag and drop intern assignments and track project progress through visual representations. Customizable widgets will allow different administrative roles to personalize their dashboard according to their specific needs.

5.4.3 Real-Time Assistance in Sprints

We will implement real-time assistance features to support interns during sprint development cycles. The system will include instant messaging capabilities for quick communication between interns and supervisors. Real-time notifications will alert interns about task updates, deadline reminders, and important announcements.

The platform will feature integrated help systems and documentation access. Interns will have immediate access to project guidelines, coding standards, and troubleshooting resources. This will reduce the time spent searching for information and improve productivity during development sprints.

5.4.4 Collaborative Tasking for Group Projects

We will enhance the platform with collaborative tasking features to improve group collaboration between interns. The system will support shared task boards where interns can view, assign, and track group project tasks. Real-time updates will ensure all team members stay informed about project progress.

The collaborative features will include shared code repositories, document collaboration tools, and peer review systems. Interns will be able to work together on the same projects simultaneously, with version control and conflict resolution mechanisms. This will foster teamwork and improve the quality of group deliverables.

5.5 Impact and Benefits

These improvements will significantly enhance the intern experience at VOID Digital Agency. The VPS integration will provide interns with professional development environments, preparing them for real-world technical work. The enhanced dashboard will reduce administrative overhead and improve operational efficiency.

Real-time assistance features will increase intern productivity and reduce the learning curve for new team members. Collaborative tasking will improve teamwork skills and project outcomes. These enhancements will position VOID as an attractive destination for talented interns seeking comprehensive learning experiences.

5.6 Summary

The Intern Management Platform has successfully improved VOID's intern management processes. While the current system provides solid foundation, the proposed improvements will significantly enhance the platform's capabilities. The focus on VPS integration, improved dashboard design, real-time assistance, and collaborative features will create a more comprehensive and user-friendly intern management solution.

These enhancements will not only improve current operations but also position VOID for future growth. The platform's modular architecture allows for incremental implementation of these features, ensuring minimal disruption to ongoing operations while delivering continuous value to both interns and administrative staff.

General Conclusion and Perspectives

The Intern Management Platform developed during this internship at VOID Digital Agency represents a significant transformation in how the company handles its recruitment process. What began as a challenge to manage over 600 annual applications through scattered emails and spreadsheets has evolved into a comprehensive, automated system that streamlines the entire recruitment workflow.

The technical architecturecombining Drupal 10 as a headless CMS with Next.js 13+ frontendproved to be an excellent choice for this project. The decoupled approach provided the flexibility needed for complex content management while delivering the performance and user experience expected from modern web applications. The widget-based content management system emerged as a key innovation, allowing content editors to create dynamic layouts without technical intervention.

This internship experience provided far more than just technical skills developmentit offered a comprehensive introduction to professional software development practices. Working on a real project with actual business impact taught invaluable lessons about software architecture, user experience design, and collaborative development methodologies. The challenges faced throughout the projectfrom debugging complex API integrations to optimizing Docker configurationsbuilt confidence in tackling unfamiliar technologies and problem-solving under pressure.

Beyond the technical achievements, this experience demonstrated the critical importance of understanding user needs and designing solutions that genuinely improve workflows. The automated recruitment workflow reduced administrative overhead significantly while improving the candidate experience. The feedback sessions with candidates and staff provided valuable insights into user experience design principles that extend beyond technical implementation.

The platform's current success provides a strong foundation for future enhancements. The planned integration of VPS technology for development environments, enhanced administrative dashboards, and real-time collaboration features will further strengthen the platform's capabilities. From a personal perspective, this internship has clarified the type of developer I aspire to become: someone who writes code that solves real problems and makes people's work lives more efficient and enjoyable.

The technical skills acquired during this internshipranging from modern frontend frameworks to backend architecture and DevOps practicesprovide a competitive advantage in

the job market. The experience of working with production systems and real business requirements demonstrates the ability to handle professional development challenges.

This internship at VOID Digital Agency has been a transformative experience that exceeded initial expectations. The Intern Management Platform stands as a testament to the power of modern web technologies when applied to real-world problems. Most importantly, this experience has reinforced the belief that software development is fundamentally about creating solutions that make people's lives bettera principle that will continue to drive professional growth and development.

Appendix A: Glossary

Headless CMS: A content management system that provides content through APIs without a traditional frontend. The content is stored and managed in the backend but can be displayed on any device or platform through API calls.

JSON:API: A specification for building APIs in JSON format that defines how resources should be formatted, how relationships should be handled, and how data should be queried. It provides a standardized way to structure API responses.

Monorepo: A software development strategy where multiple related projects are stored in a single repository. This approach facilitates code sharing, dependency management, and coordinated development across projects.

Slug: A user-friendly URL identifier derived from a page title or content name.

UUID (Universally Unique Identifier): A 128-bit number used to uniquely identify information in computer systems. In Drupal, UUIDs are used to identify content entities across different environments.

Widget: A reusable component that encapsulates specific functionality and can be embedded in pages or other widgets. In this project, widgets bridge Drupal content management with Next.js component rendering.

Content Type: A predefined template in Drupal that defines the structure and fields for a specific type of content. Examples include "Job Offer," "Candidature," or "Quiz." Each content type can have custom fields and behaviors.

Entity: The fundamental unit of content in Drupal. Entities can be nodes (content), users, taxonomy terms, or custom entities. Each entity has fields that store data and can have relationships with other entities.

Hook: A PHP function that allows modules to interact with Drupal core and other modules. Hooks follow a naming convention (hook_event_name) and are called at specific points in Drupal's execution to allow custom functionality.

Paragraph: A Drupal entity type that allows content creators to build flexible, structured content by combining different paragraph types. Each paragraph type can have different fields and layouts.

Taxonomy: A system for classifying and organizing content using terms and vocabularies. For example, a "Specialization" vocabulary might contain terms like "Frontend Development," "Backend Development," and "DevOps."

Vocabulary: A collection of taxonomy terms used for categorizing content. Multiple vocabularies can exist for different classification systems (e.g., Tags, Categories, Specializations).

Webform: A Drupal module that provides form building capabilities, allowing administrators to create complex forms with various field types, validation rules, and submission handling logic.

AccessHandler Controller: A Drupal class responsible for determining whether a user has permission to perform specific operations (view, edit, delete) on entities. It implements access control logic based on user roles and permissions.

Webpack: A module bundler for JavaScript applications that processes and optimizes assets (JavaScript, CSS, images) for production deployment. It can also run custom plugins during the build process.

Bibliography

OWASP Foundation. Owasp top ten web application security risks, 2025. URL https://owasp.org/www-project-top-ten/.

VOID Digital Agency. Clients - Études de cas marketing digital, 2025. URL https://void.fr/fr/etude-de-cas-marketing-digital.

GeeksforGeeks. Software development life cycle (sdlc), 2025. URL https://www.geeksforgeeks.org/software-engineering/software-development-life-cycle-sdlc/.

Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015. ISBN 978-1491950357.

Dries Buytaert. Headless cms: Rest vs json:api vs graphql. https://dri.es/headless-cms-rest-vs-jsonapi-vs-graphql, 2019.

Traefik. Traefik docker provider, 2025. URL https://doc.traefik.io/traefik/providers/docker/.