

# 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 faculty 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 faculty 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 and automated testing, 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, Test-Driven Development (TDD), React, and Next.js, demonstrating the integration of modern web technologies with robust development practices. Secondary achievements included successful server migrations, custom PHP-FPM Docker image development, and the creation of an automated dependency management system.

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. The report concludes with recommendations for future enhancements, particularly in Kubernetes orchestration and automated maintenance systems, positioning the project for scalable growth and continued technological advancement.

**Keywords:** DevOps, CI/CD, Docker, Headless CMS, Drupal, React, Next.js, Test-Driven Development, Intern Management, Automation, Containerization, Web Development, PHP, Kubernetes

# Résumé

Ce rapport présente les travaux réalisés dans le cadre de mon stage de fin d'études au sein de VOID Digital Agency, une agence digitale indépendante leader de l'expérience utilisateur au Maroc.

L'objectif principal était de concevoir et de mettre en uvre un pipeline CI/CD robuste ainsi que de développer une plateforme de gestion des stagiaires basée sur un CMS headless Drupal, en utilisant PHP, Docker, le développement piloté par les tests (TDD), React et Next.js.

Les tâches secondaires comprenaient la migration de serveurs d'un environnement de production vers un environnement de test, la création d'images Docker personnalisées pour PHP-FPM, ainsi que le développement d'un outil automatisé de vérification et de mise à jour des dépendances.

Le pipeline CI/CD a permis de réduire de plus de 60% le temps de déploiement, tandis que la plateforme a centralisé la gestion des stagiaires, amélioré les processus d'intégration et permis un suivi en temps réel de leur progression.

Enfin, ce rapport traite des principales difficultés rencontrées telles que le réseau de conteneurs, les tests instables, et l'intégration d'un CMS headless et propose des pistes d'amélioration futures, notamment l'orchestration avec Kubernetes et l'automatisation étendue de la maintenance.

**Mots-clés:** DevOps, CI/CD, Docker, Headless CMS, Drupal, React, Next.js, Test-Driven Development, Intern Management, Automation, Containerization, Web Development, PHP, Kubernetes

# Contents

<b>Dedication</b>	<b>i</b>
<b>Acknowledgement</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Résumé</b>	<b>iv</b>
<b>List of Figures</b>	<b>xii</b>
<b>General Introduction</b>	<b>1</b>
<b>List of Acronyms</b>	<b>1</b>
<b>1 General Context of the Project</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Presentation of VOID . . . . .	2
1.2.1 Core Values . . . . .	3
1.2.2 Technical Expertise . . . . .	3
1.2.3 Notable Clients . . . . .	3
1.2.4 Recognition . . . . .	3
1.2.5 Company Solutions . . . . .	4
1.2.6 Company Services . . . . .	4
1.2.7 Clients and Projects . . . . .	5
1.2.8 Awards and Recognitions . . . . .	5
1.3 The Main Project . . . . .	6
1.3.1 Training Phase at VOID . . . . .	6
1.3.2 Problematic . . . . .	7

## **0. CONTENTS**

---

1.3.3	Objectives . . . . .	7
1.3.4	Proposed Solution . . . . .	8
1.4	Project Management Methodology . . . . .	9
1.4.1	Agile Methodology . . . . .	9
1.4.2	Organization and Communication Tools . . . . .	10
1.5	Conclusion . . . . .	11
<b>2</b>	<b>Project Analysis and Design</b>	<b>12</b>
2.1	Introduction . . . . .	12
2.2	Study of the Existing System . . . . .	12
2.2.1	Operational Inefficiencies . . . . .	12
2.2.2	Technical Limitations . . . . .	13
2.2.3	Impact on Stakeholders . . . . .	13
2.3	Specification Document . . . . .	13
2.3.1	Functional Requirements . . . . .	13
2.3.2	Non-Functional Requirements . . . . .	14
2.3.3	Success Criteria . . . . .	15
2.4	User Research and Methodology . . . . .	15
2.4.1	Research Methods . . . . .	15
2.4.2	User Journey Analysis . . . . .	16
2.5	UML Diagrams Overview . . . . .	16
2.5.1	Use Case Diagram . . . . .	16
2.5.2	Sequence Diagram . . . . .	17
2.5.3	Class Diagram . . . . .	20
2.5.4	Project Timeline Diagrams . . . . .	22
2.6	Mockups and Wireframes . . . . .	24
2.7	Conclusion . . . . .	25
<b>3</b>	<b>Main Project Implementation</b>	<b>26</b>
3.1	Introduction . . . . .	26
3.2	Programming Languages and Frameworks . . . . .	26
3.2.1	Programming Languages . . . . .	26
3.2.2	Frameworks and Libraries . . . . .	26

## 0. CONTENTS

---

3.2.3	Database and Caching . . . . .	27
3.3	Development Tools . . . . .	27
3.4	Development Environment . . . . .	28
3.4.1	Backend Containers . . . . .	28
3.4.2	Frontend Containers . . . . .	28
3.4.3	Secure API Proxying . . . . .	29
3.4.4	Caching Architecture . . . . .	29
3.4.5	Mail Testing Infrastructure . . . . .	30
3.4.6	Component Development with Storybook . . . . .	30
3.4.7	Routing Strategy . . . . .	30
3.5	Backend Implementation . . . . .	30
3.5.1	Introduction . . . . .	30
3.5.2	Drupal Architecture and API Strategy . . . . .	31
3.5.3	Performance Optimization with Memcached . . . . .	33
3.5.4	Email Debugging with MailHog . . . . .	34
3.5.5	Widget System Architecture and Discovery Process . . . . .	34
3.5.6	Main Website Backend Structure . . . . .	37
3.5.7	Training and Resource Management Custom Modules . . . . .	37
3.5.8	Security and Access Control . . . . .	40
3.6	Frontend Implementation . . . . .	40
3.6.1	Decoupled Architecture and Backend Integration . . . . .	40
3.6.2	Routing and Path Resolution . . . . .	42
3.6.3	Monorepo Architecture and Starter Kit Integration . . . . .	43
3.6.4	Design System and Storybook Integration . . . . .	44
3.6.5	Icon System and Asset Management . . . . .	45
3.6.6	Custom Widgets and Modules Implementation . . . . .	45
3.6.7	Context-Based Candidature Data . . . . .	47
3.6.8	React Query Integration . . . . .	47
3.6.9	Performance Optimization and Testing . . . . .	47
3.7	DevOps and CI/CD Pipeline Setup . . . . .	50
3.7.1	DevOps Stack Overview . . . . .	50
3.7.2	CI/CD Workflow Description . . . . .	50

## **0. CONTENTS**

---

3.7.3	Bitbucket CI/CD Pipeline Script (Simplified) . . . . .	51
3.7.4	Environment Configuration . . . . .	51
3.7.5	Automated Dependency Updates . . . . .	51
3.7.6	Benefits Observed . . . . .	52
3.8	Testing and Quality Assurance . . . . .	52
3.8.1	Testing Strategy Overview . . . . .	52
3.8.2	Frontend Testing Tools . . . . .	52
3.8.3	Backend Testing Tools . . . . .	53
3.8.4	Linting and Static Code Analysis . . . . .	53
3.8.5	Example Test Case (React + Jest) . . . . .	53
3.8.6	Continuous Testing Integration . . . . .	53
3.8.7	Quality Metrics . . . . .	54
<b>4</b>	<b>Project Realization</b>	<b>55</b>
4.1	Introduction . . . . .	55
4.2	Project Results and Showcase . . . . .	55
4.2.1	Main Page Components . . . . .	55
4.2.2	Intern Sprint Management . . . . .	56
4.2.3	Learning Resources . . . . .	58
4.2.4	Project Management . . . . .	59
4.2.5	User Profile . . . . .	60
4.2.6	Document Management . . . . .	61
<b>5</b>	<b>Discussion and Future Improvements</b>	<b>63</b>
5.1	Introduction . . . . .	63
5.2	Significance of the Achieved Results . . . . .	63
5.2.1	Process Transformation and Efficiency Gains . . . . .	63
5.2.2	Technical Architecture Success . . . . .	64
5.2.3	User Experience Enhancement . . . . .	64
5.3	Current Platform Limitations . . . . .	64
5.3.1	Candidate Space Limitations . . . . .	64
5.3.2	Administrative Interface Limitations . . . . .	65
5.4	Proposed Future Improvements . . . . .	66

5.4.1	Administrative Interface Enhancement . . . . .	66
5.4.2	Enhanced Security and Anti-Cheating Measures . . . . .	66
5.4.3	Candidate Experience Improvements . . . . .	66
5.5	Impact Assessment and Validation . . . . .	67
5.5.1	Quantitative Metrics . . . . .	67
5.5.2	Qualitative Benefits . . . . .	67
5.6	Summary . . . . .	68
<b>Conclusion</b>		<b>69</b>
<b>Appendix B: Glossary</b>		<b>70</b>

# List of Figures

1.1	Logo of VOID Digital Agency . . . . .	2
1.2	VOID Digital Agency Services . . . . .	4
1.3	VOID Digital Agency Clients . . . . .	5
1.4	Awards and recognitions received by VOID Digital Agency. . . . .	6
1.5	Software Development Life Cycle (SDLC) . . . . .	10
1.6	Redmine project management tool . . . . .	10
1.7	Slack . . . . .	11
1.8	Loom . . . . .	11
1.9	Google Meet . . . . .	11
1.10	Google Calendar . . . . .	11
1.11	Communication and collaboration tools used at VOID . . . . .	11
2.1	Use Case Diagram for the Intern Management Platform . . . . .	17
2.2	Sequence Diagram - Sprint Management Process Flow . . . . .	18
2.3	Sequence Diagram - Project Management Process Flow . . . . .	19
2.4	Sequence Diagram - Resource Management Process Flow . . . . .	19
2.5	Class Diagram - System Entity Relationships and Data Model . . . . .	21
2.6	Project Implementation Gantt Chart - VOID Intern Management Platform Sprint Timeline (6 Weeks) . . . . .	23
2.7	High-Fidelity Mockups Landing Website . . . . .	24
2.8	High-Fidelity Mockups Candidate Space . . . . .	24
2.9	High-Fidelity Mockups Formation Space . . . . .	25
3.1	Core Programming Languages: PHP, HTML5, and JavaScript . . . . .	26
3.2	Core Frameworks and Libraries: Next.js, Tailwind CSS, and Drupal . . . . .	27
3.3	Database and Caching Tools: MySQL, Redis, Memcached, and Mailhog . . . . .	27

3.4	Development Tools: VS Code, Docker, Orbstack, Git, Bitbucket, Docker Hub, EasyPanel, and Nginx . . . . .	28
3.5	Storybook UI . . . . .	30
3.6	JSON:API endpoint configuration with path prefix /api . . . . .	31
3.7	Exposed JSON:API resources in Drupal . . . . .	32
3.8	Visual placement of widgets (blocks) in the Drupal back office . . . . .	33
3.9	Memcached UI . . . . .	34
3.10	Widget Selection Interface with Category Tabs . . . . .	35
3.11	Available Widget Templates in Void Training Category . . . . .	36
3.12	Widget Configuration Interface for Recruitment Process . . . . .	36
3.13	Webpack Mapping Process . . . . .	42
3.14	Webpack Listing File . . . . .	42
3.15	Storybook Button Component . . . . .	44
3.16	Redis Commander UI . . . . .	48
3.17	Offline Mode Implementation with Redis Cache Schema . . . . .	49
3.18	CI/CD Pipeline for the Intern Management Platform . . . . .	51
3.19	Testing Pipeline Integration in CI/CD . . . . .	54
4.1	Main Page Cards Display . . . . .	55
4.2	Main Page Slider . . . . .	56
4.3	Intern Sprint Listing . . . . .	56
4.4	Sprint Details Page . . . . .	57
4.5	Learning Resources Page . . . . .	58
4.6	Learning Resources Page (Accounts) . . . . .	59
4.7	Project Management Page . . . . .	60
4.8	User Profile Page . . . . .	61
4.9	Document Management Page . . . . .	62

# List of Acronyms

<b>AI</b>	<i>Artificial Intelligence</i>
<b>API</b>	<i>Application Programming Interface</i>
<b>CI/CD</b>	<i>Continuous Integration/Continuous Deployment</i>
<b>CMS</b>	<i>Content Management System</i>
<b>CTA</b>	<i>Call To Action</i>
<b>CV</b>	<i>Curriculum Vitae</i>
<b>DevOps</b>	<i>Development and Operations</i>
<b>Drush</b>	<i>Drupal Shell</i>
<b>GitOps</b>	<i>Git Operations</i>
<b>OWASP</b>	<i>Open Web Application Security Project</i>
<b>PHP-FPM</b>	<i>PHP FastCGI Process Manager</i>
<b>PMO</b>	<i>Project Management Office</i>
<b>QA</b>	<i>Quality Assurance</i>
<b>SDLC</b>	<i>Software Development Life Cycle</i>
<b>SEO/SEA</b>	<i>Search Engine Optimization/Search Engine Advertising</i>
<b>TDD</b>	<i>Test-Driven Development</i>
<b>UX/UI</b>	<i>User Experience/User Interface</i>

# 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

## **1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT**

---

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 Notable Clients**

VOID serves leading organizations across various sectors. In finance, they work with Attijariwafa Bank, Bank of Africa, and CIH Bank. Their healthcare clients include Saham Assurance and CNOPS, while Orange Maroc represents their telecom portfolio. The agency also serves retail clients like Marjane and Label'Vie, and cultural institutions such as the Mohammed VI Museum and ONMT.

### **1.2.4 Recognition**

VOID has received numerous prestigious awards including:

- Janus du Service (Institut Français du Design)
- Grand Prix Stratégies

## **1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT**

---

- FWA (Favourite Website Awards)
- The Webby Awards
- Cannes Lions International Festival of Creativity

### **1.2.5 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 ?.

### **1.2.6 Company Services**

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

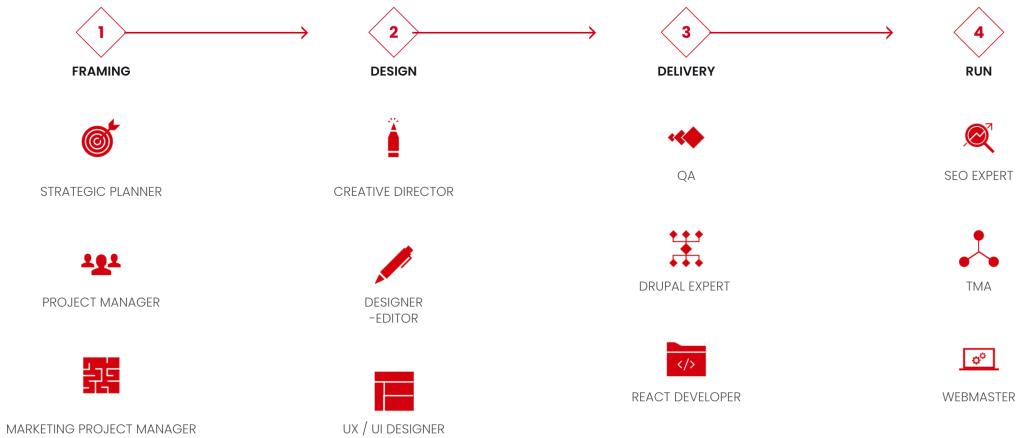


Figure 1.2: VOID Digital Agency Services

- **Transmedia Content** - Corporate videos, product demonstrations, and editorial production.
- **Social Media Activation** - Campaign design, content calendars, and performance analytics.

## 1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT

---

- **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.

### 1.2.7 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:



Figure 1.3: VOID Digital Agency Clients

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.

### 1.2.8 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

## **1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT**

---

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**, and **implementing Test-Driven Development (TDD)** strategies, 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.

## **1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT**

---

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.

## **1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT**

---

- **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.

## **1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT**

---

- 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 developed following **Test-Driven Development (TDD)** principles to guarantee robust and reliable features, and 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), and **TDD implementation**, 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.

# 1. CHAPTER 1. GENERAL CONTEXT OF THE PROJECT

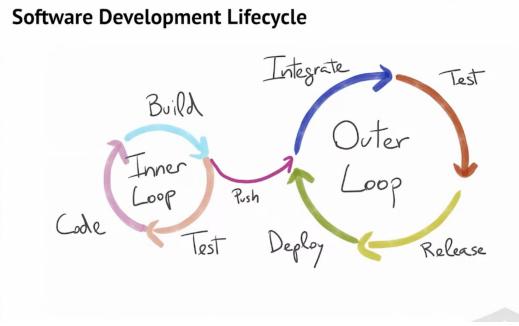


Figure 1.5: Software Development Life Cycle (SDLC)

## 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.

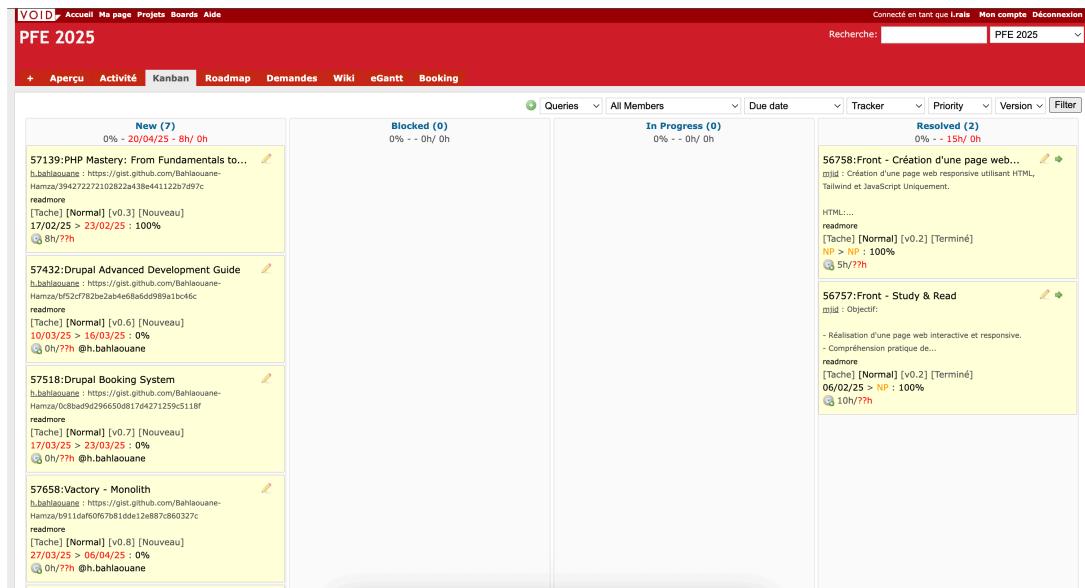


Figure 1.6: Redmine project management tool

- **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 platform 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.

## **2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN**

---

- **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.
- **Usability:** New users should be able to complete basic tasks with less than 30 minutes of training; usability testing should target at least 90% satisfaction.

### **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.

## **2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN**

---

- **User Journey Mapping:** The team mapped out the complete candidate journey from 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:

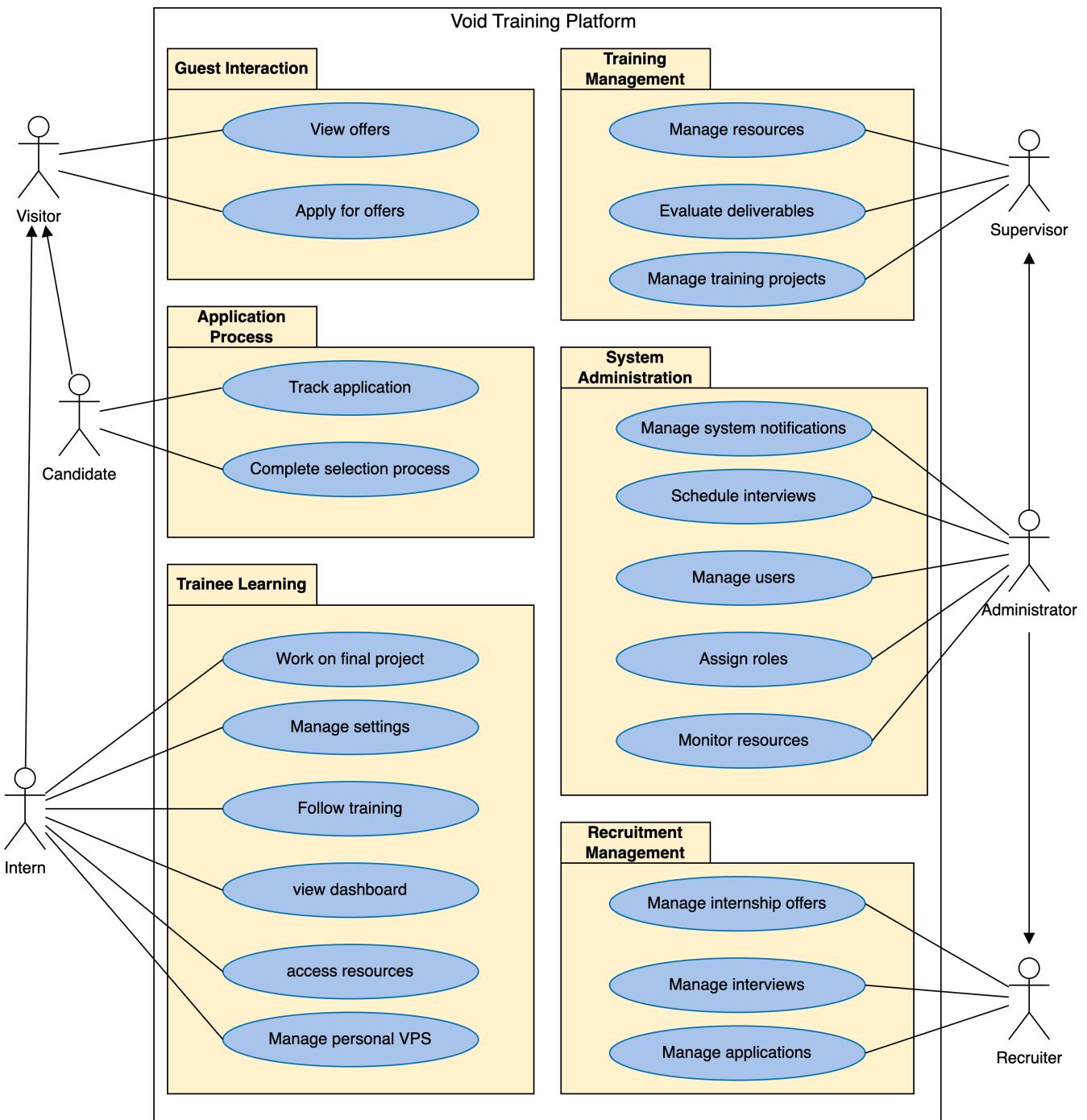


Figure 2.1: Use Case Diagram for the Intern Management Platform

### 2.5.2 Sequence Diagram

The sequence diagram (Figure ??) represents the detailed interaction flow of the candidate application process, showing the communication between different system components throughout the recruitment pipeline. This diagram illustrates the technical architecture and data flow from the candidate's initial application submission to the final decision notification.

## 2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN

---

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

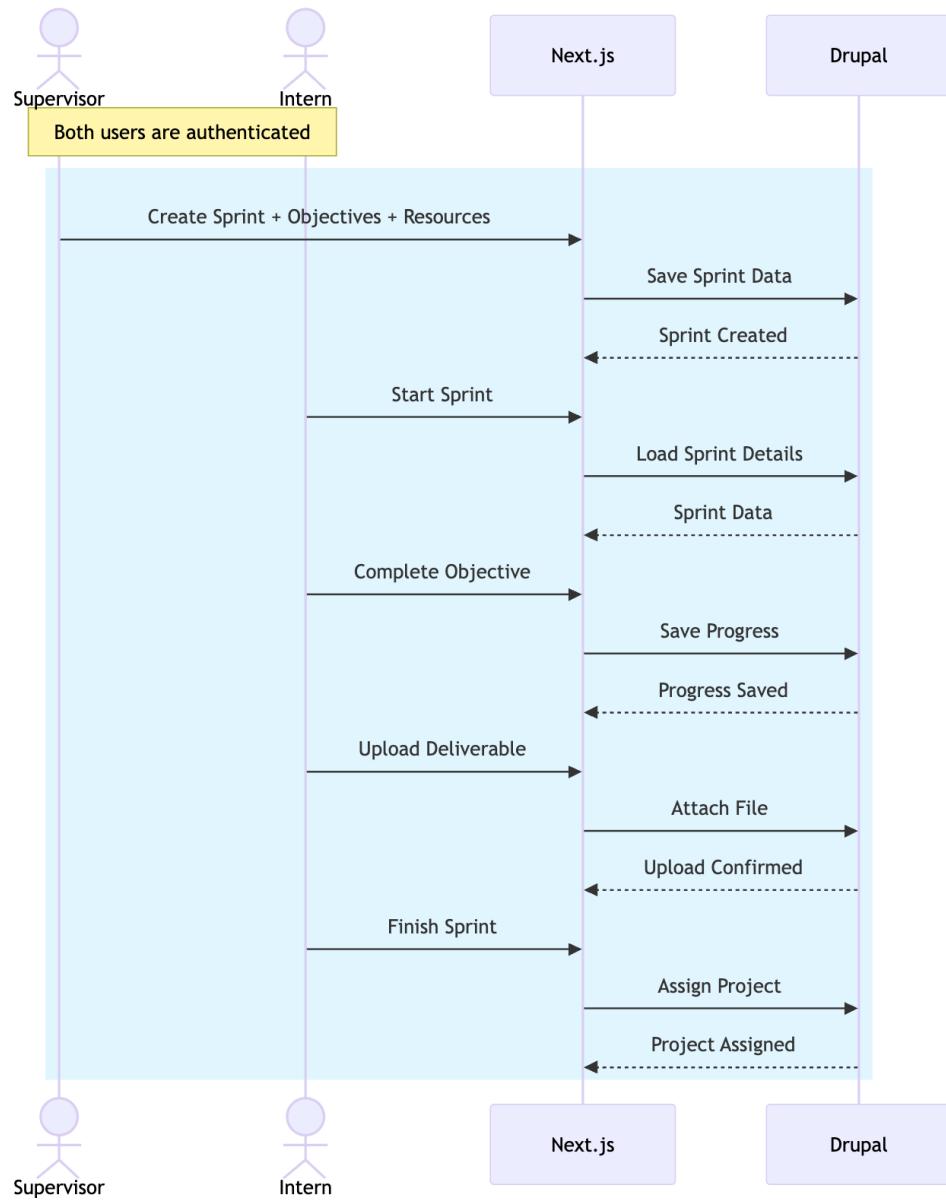


Figure 2.2: Sequence Diagram - Sprint Management Process Flow

## 2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN

---

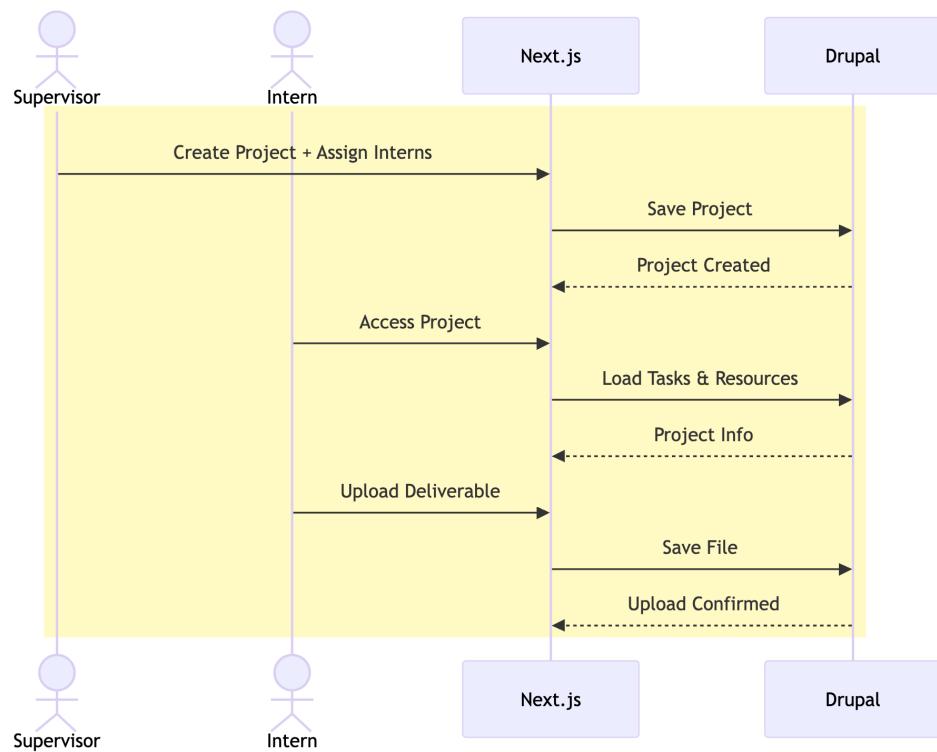


Figure 2.3: Sequence Diagram - Project Management Process Flow

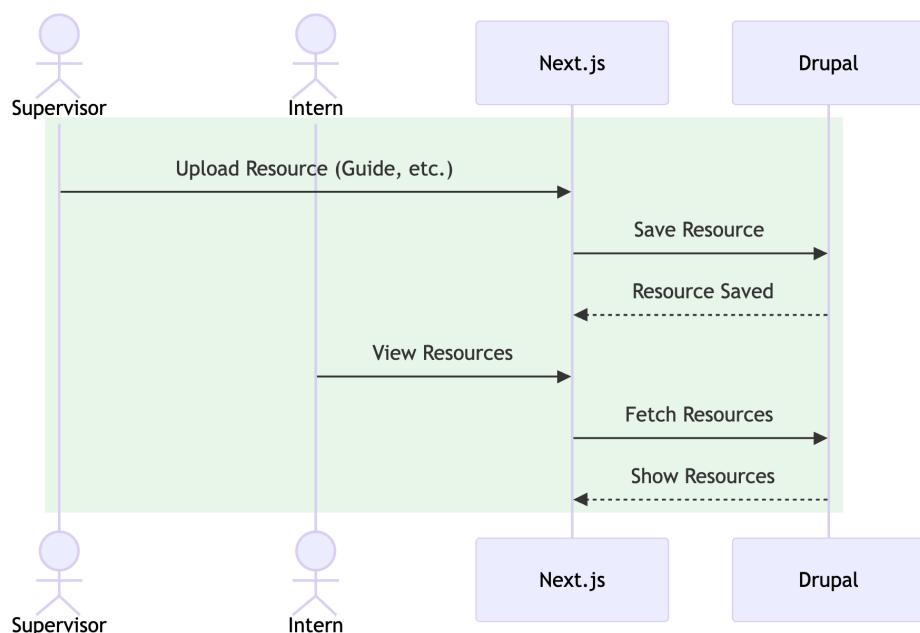


Figure 2.4: Sequence Diagram - Resource Management Process Flow

### **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.

## 2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN

---

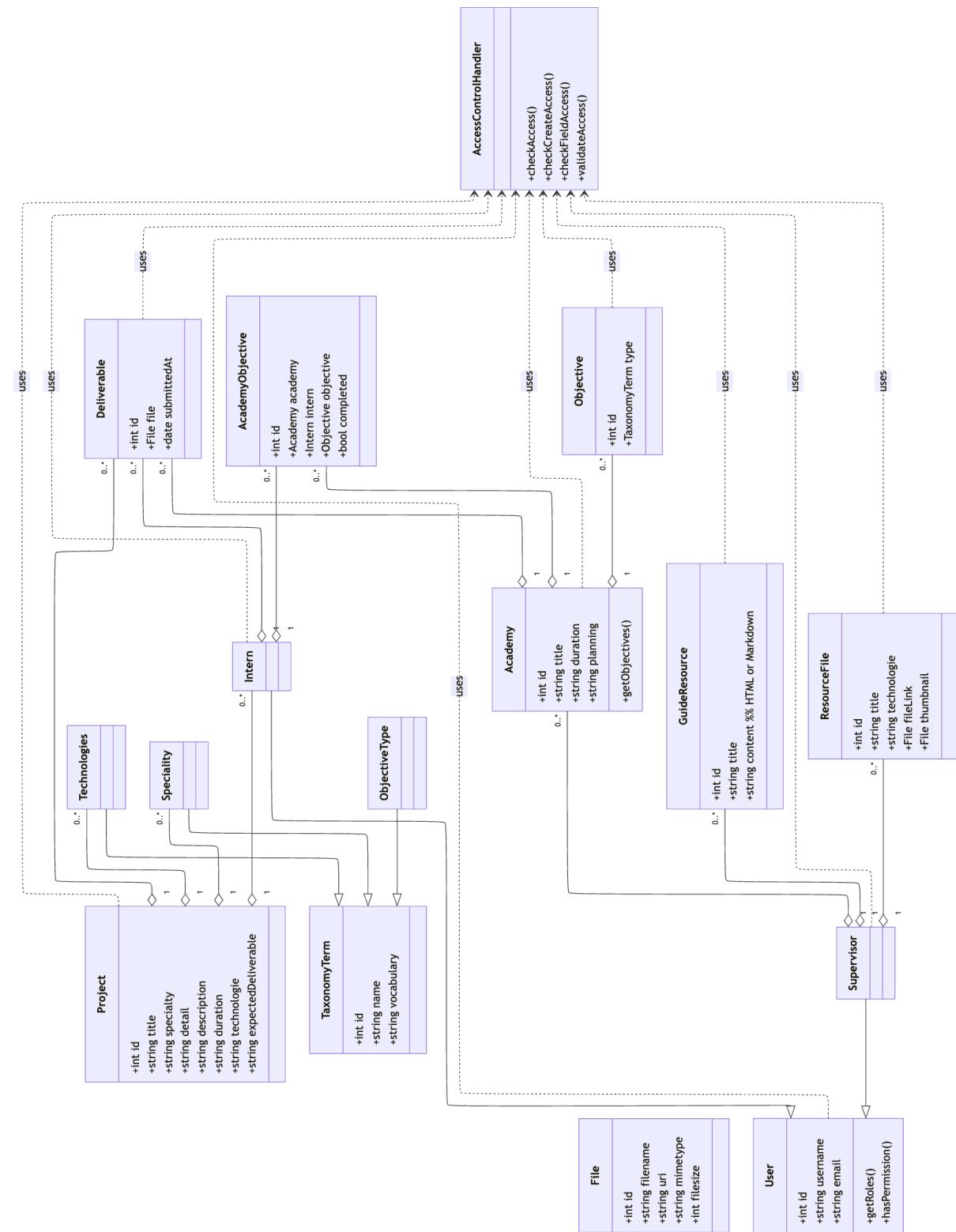


Figure 2.5: Class Diagram - System Entity Relationships and Data Model

### **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.

## 2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN

---

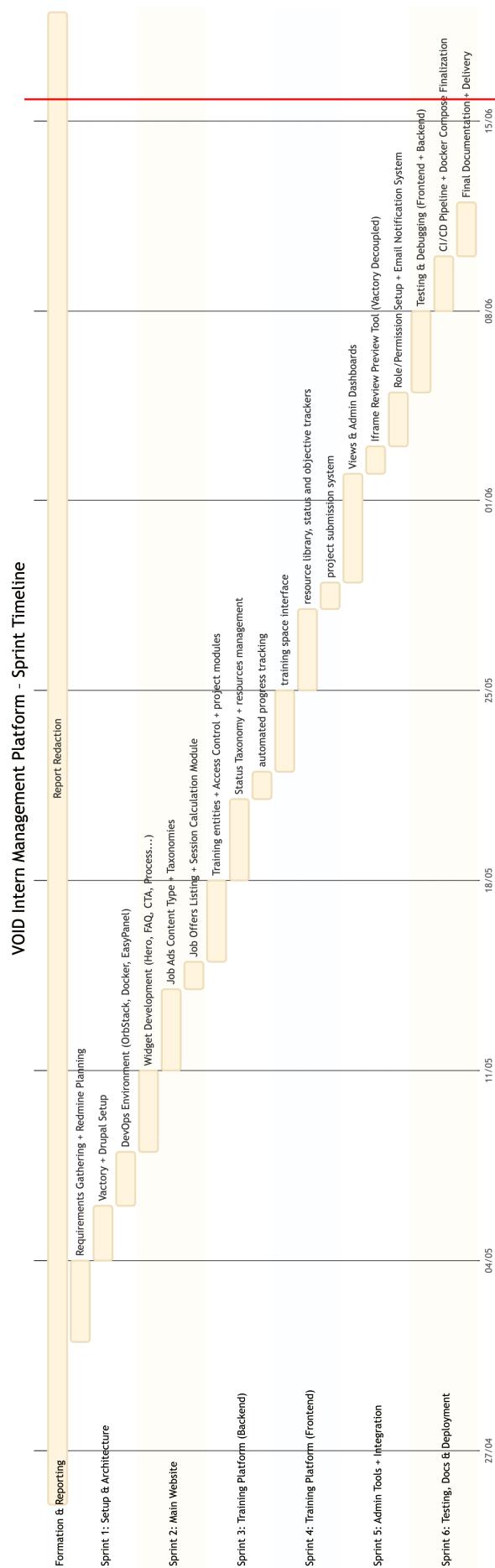


Figure 2.6: Project Implementation Gantt Chart - VOID Intern Management Platform<sup>23</sup>  
Sprint Timeline (6 Weeks)

### 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.

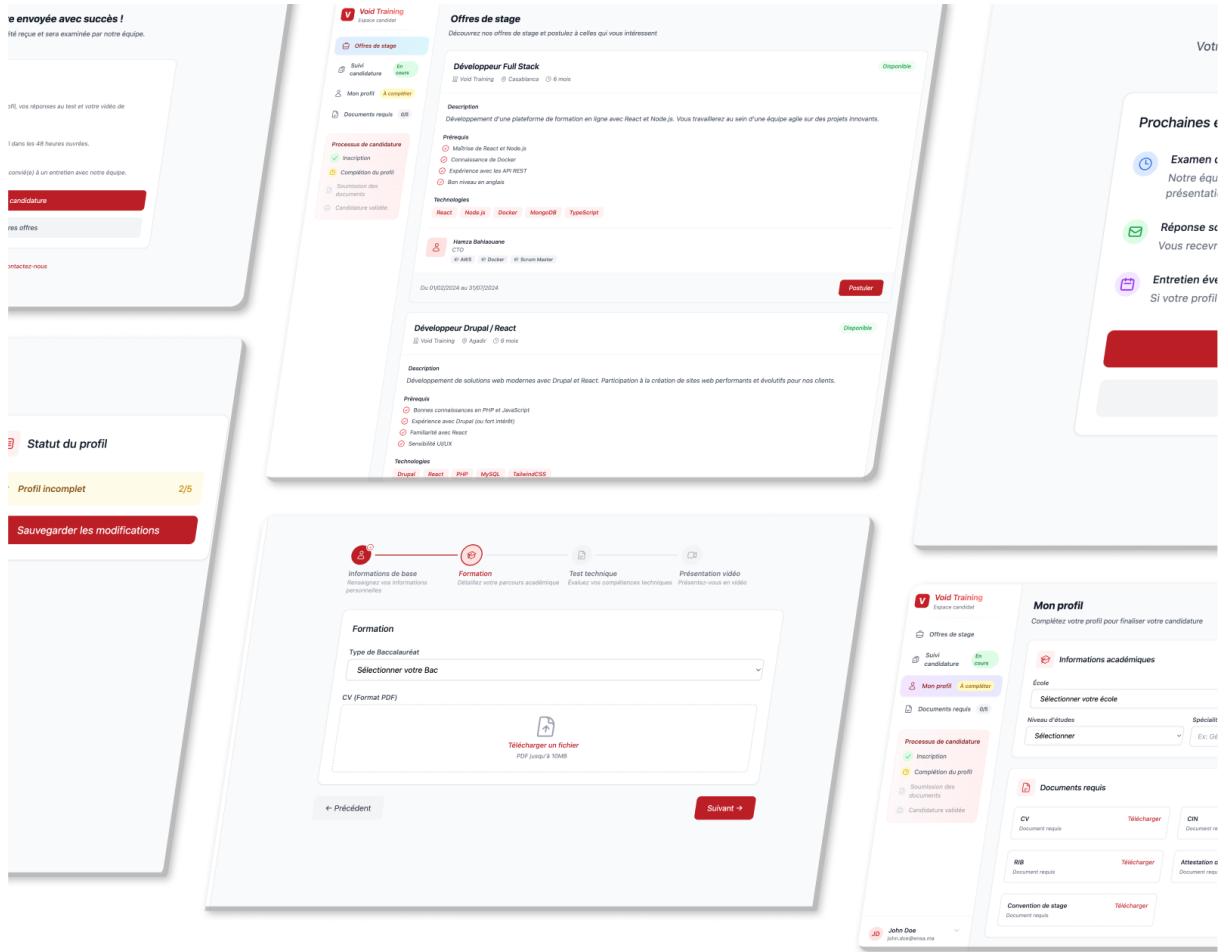


Figure 2.8: High-Fidelity Mockups Candidate Space

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

## 2. CHAPTER 2. PROJECT ANALYSIS AND DESIGN

---

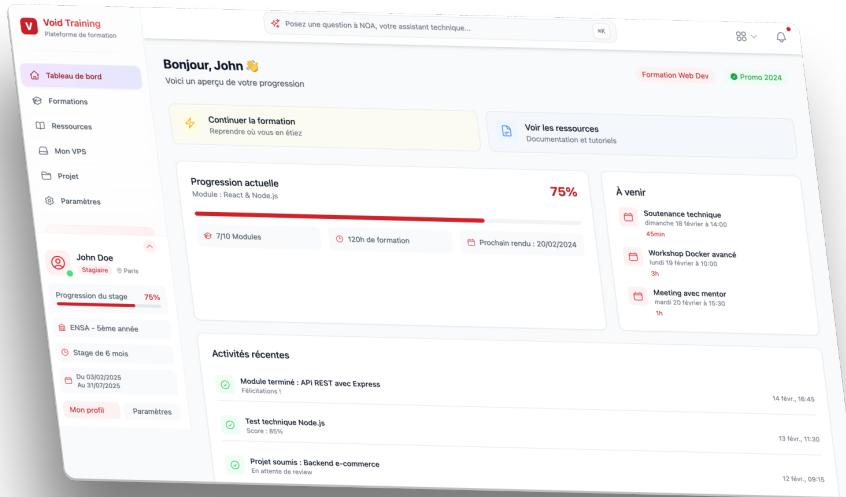


Figure 2.9: High-Fidelity Mockups Formation Space

- **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.

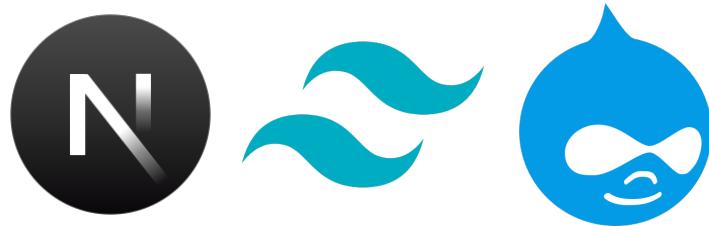


Figure 3.2: Core Frameworks and Libraries: Next.js, Tailwind CSS, and Drupal

#### 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.

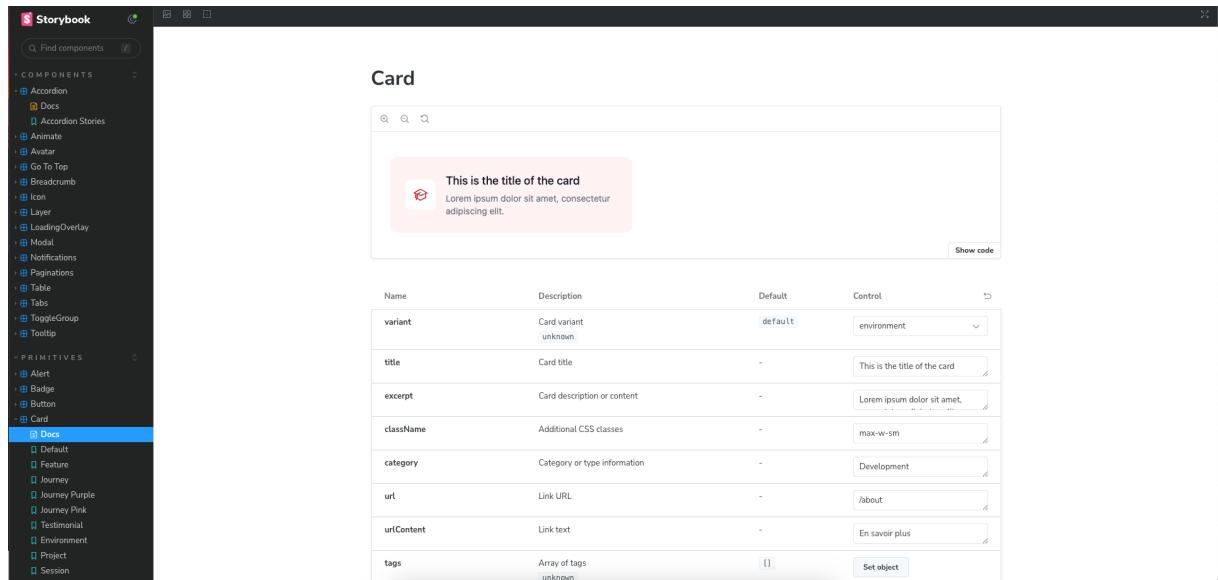


Figure 3.5: Storybook UI

#### 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.

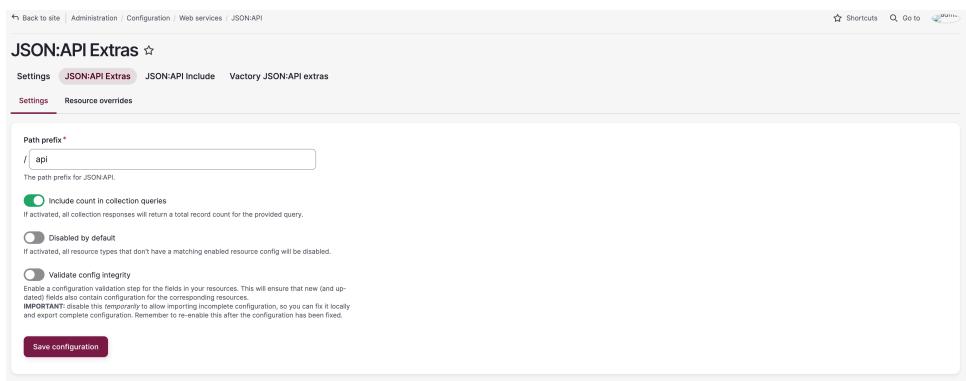
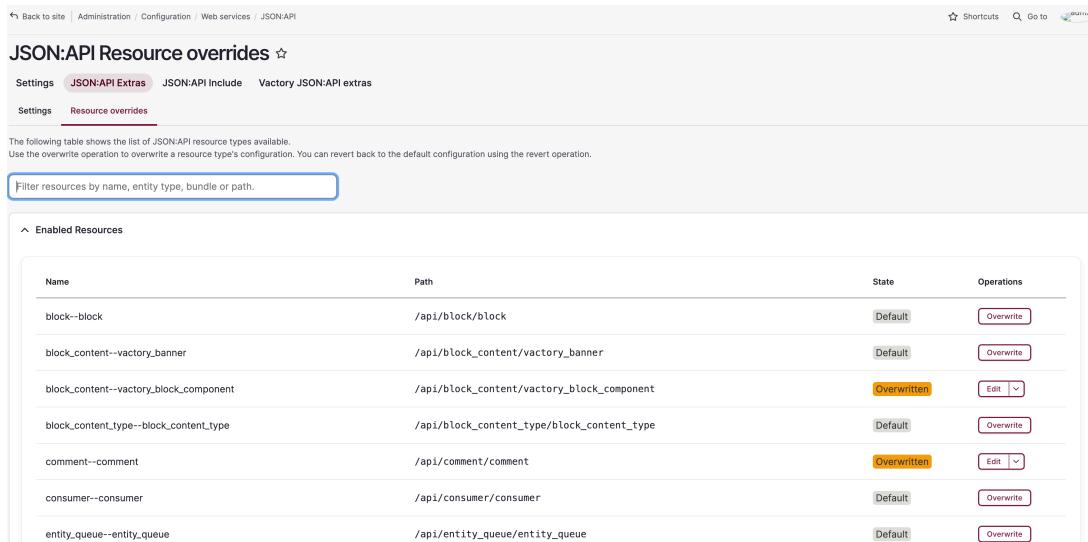


Figure 3.6: JSON:API endpoint configuration with path prefix /api

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION



The screenshot shows the 'JSON:API Resource overrides' page in the Drupal configuration interface. The top navigation bar includes links for 'Back to site', 'Administration', 'Configuration', 'Web services', and 'JSON:API'. On the right, there are 'Shortcuts', 'Go to', and a search bar. The main content area has tabs for 'Settings' and 'JSON:API Extras' (which is selected), 'JSON:API Include', and 'Vactory JSON:API extras'. Below these tabs is another set for 'Settings' and 'Resource overrides' (which is selected). A note states: 'The following table shows the list of JSON:API resource types available. Use the overwrite operation to overwrite a resource type's configuration. You can revert back to the default configuration using the revert operation.' A 'Filter resources by name, entity type, bundle or path.' input field is present. A table titled '^ Enabled Resources' lists various resource types with their paths and states:

Name	Path	State	Operations
block--block	/api/block/block	Default	<button>Overwrite</button>
block_content--vactory_banner	/api/block_content/vactory_banner	Default	<button>Overwrite</button>
block_content--vactory_block_component	/api/block_content/vactory_block_component	Overwritten	<button>Edit</button>
block_content_type--block_content_type	/api/block_content_type/block_content_type	Default	<button>Overwrite</button>
comment--comment	/api/comment/comment	Overwritten	<button>Edit</button>
consumer--consumer	/api/consumer/consumer	Default	<button>Overwrite</button>
entity_queue--entity_queue	/api/entity_queue/entity_queue	Default	<button>Overwrite</button>

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 components such as headers, footers, and card lists are 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.

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

---

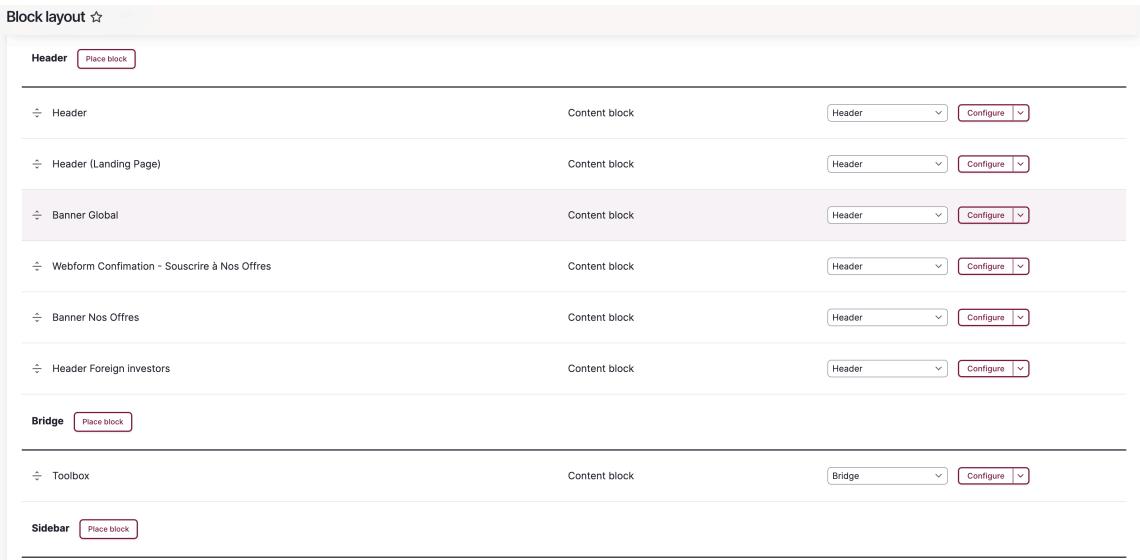


Figure 3.8: Visual placement of widgets (blocks) in the Drupal back office

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.

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

---

The screenshot shows the Memcached UI interface. At the top, there's a navigation bar with links like Back to site, Manage, Shortcuts, admin, Go to, and Devel. Below that is a secondary navigation bar with Home, Administration, Reports, and Memcache Statistics. The main content area is titled "Memcache Statistics". It displays various performance metrics for a Memcached instance at port 11211.

void-training_memcached:11211	
Uptime	v1.6.23 running 4 days 19 hours
Time	Mon, 06/09/2025 - 13:47
Connections	3 open of 13,695 total
 void-training_memcached:11211	
Sets	0.00/s; 486,091 sets (19.68%) of 2,469,486 commands
Gets	4.77/s; 1,983,395 gets (80.32%); 1,899,859 hits (95.79%) 83,536 misses (4.21%)
Counters	0 increments, 0 decrements
Transferred	643.25 MB:2.61 GB (24.03% to cache)
Per-connection average	204 bytes in 144.83 gets; 49 bytes in 35.49 sets
 void-training_memcached:11211	
Available memory	30.75 MB (48.04%) of 64 MB
Evictions	0

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

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

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

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

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.

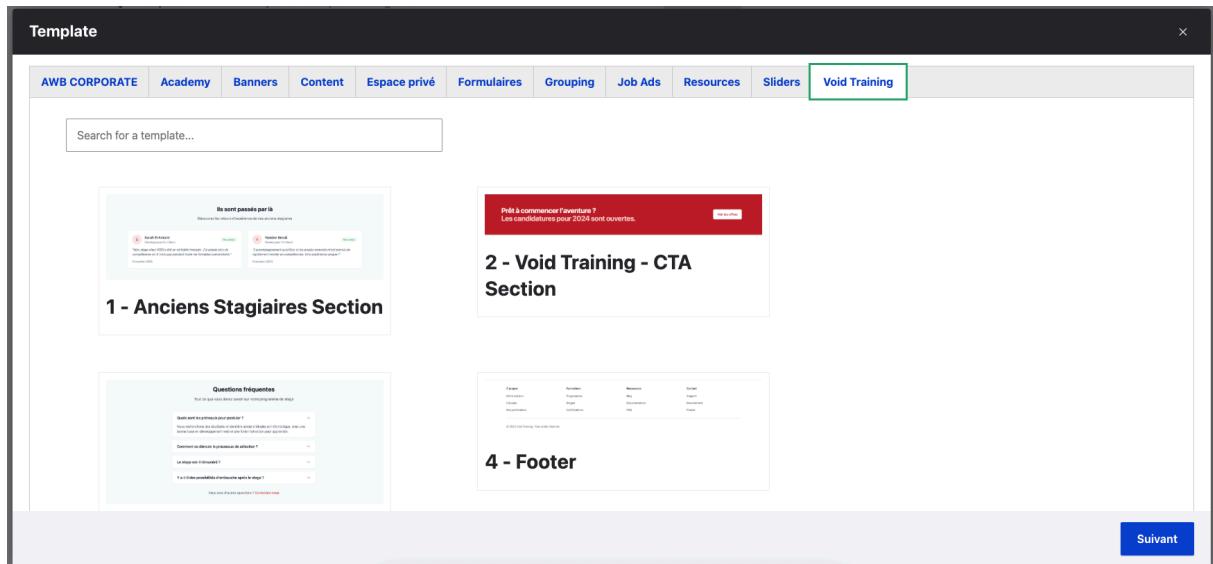


Figure 3.11: Available Widget Templates in Void Training Category

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.

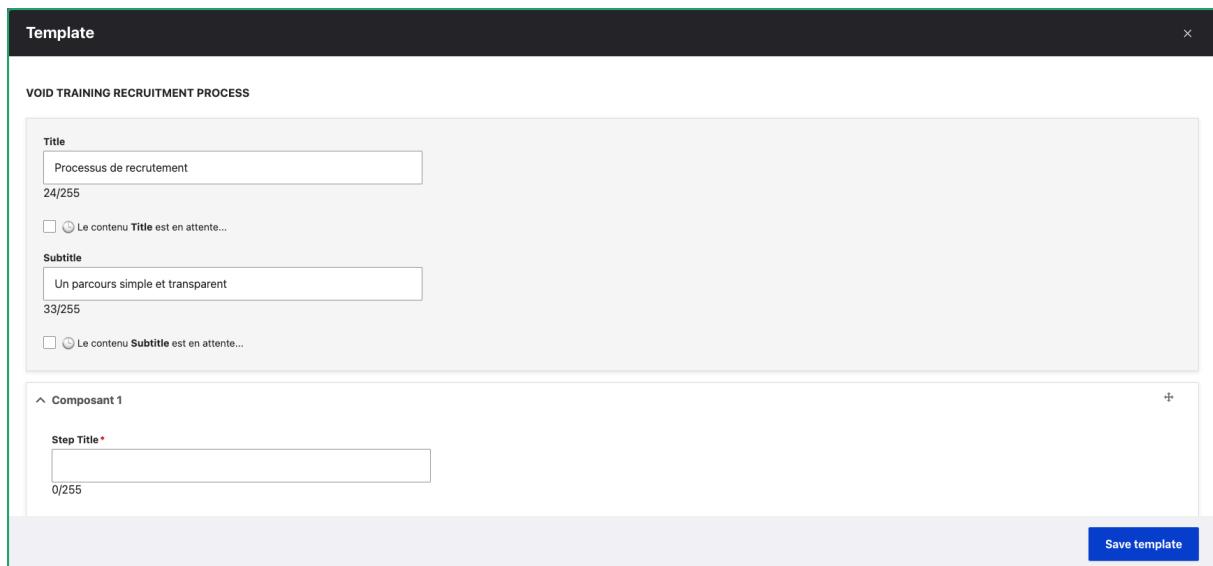


Figure 3.12: Widget Configuration Interface for Recruitment Process

#### **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

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

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:**

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

- 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**

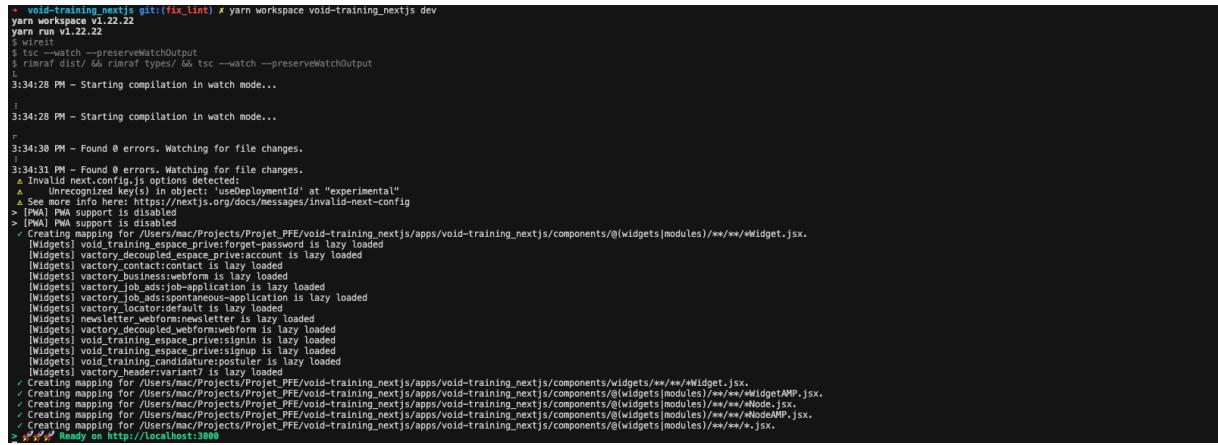
The mapping between backend widget configurations and frontend components is orchestrated by a custom Webpack plugin. This plugin automates the discovery and registration of widget components, ensuring seamless integration and maintainability. The core responsibilities of the `WidgetsPlugin` include:

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.

The complete implementation of the Webpack plugin is provided in Appendix ??.

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

---



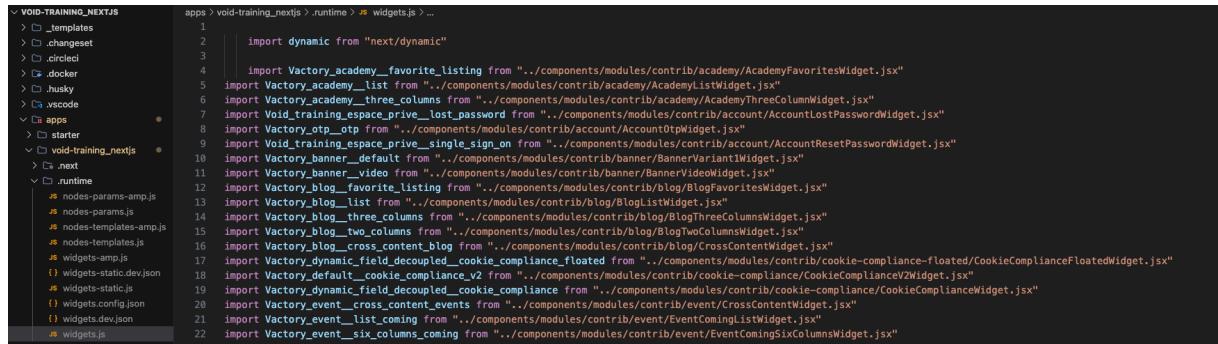
```

" void-training_nextjs git:(fix_lint) ✘ yarn workspace void-training_nextjs dev
yarn workspace v1.22.22
yarn run v1.22.22
$ wixrt
$ tsc --watch --preserveWatchOutput
$ rimraf dist/ && rimraf types/ && tsc --watch --preserveWatchOutput
$ 
3:34:28 PM - Starting compilation in watch mode...
:
3:34:28 PM - Starting compilation in watch mode...
:
3:34:39 PM - Found 0 errors. Watching for file changes.
3:34:31 PM - Found 0 errors. Watching for file changes.
  ▲ Invalid next.config.js options detected!
  ▲ Unknown option 'experimental': 'useDeploymentId' at "experimental"
  ▲ See more info here: https://nextjs.org/docs/messages/invalid-next-config
> [PWA] PWA support is disabled
> [PWA] PWA support is disabled
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/Widget.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetAMP.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/Widget.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetMP.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSR.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSRAMP.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSR.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSRMP.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSRSSR.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSRSSRAMP.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSRSSR.jsx.
  ✓ Creating mapping for /Users/mac/Projects/Project_PFE/void-training_nextjs/apps/void-training_nextjs/components/@(widgets|modules)/**/**/WidgetSSRSSRMP.jsx.

> ⚡ Ready on http://localhost:3000

```

Figure 3.13: Webpack Mapping Process



```

 VOID-TRAINING_NEXTJS
├── _templates
├── .changeset
├── .circleci
├── .docker
├── .husky
├── .vscode
└── __app__
    ├── .next
    └── runtime
        ├── apps
        │   ├── starter
        └── void-training_nextjs
            ├── .next
            └── runtime
                ├── nodes-params-amp.js
                ├── nodes-params.js
                ├── nodes-templates-amp.js
                ├── nodes-templates.js
                ├── widgets-amp.js
                ├── widgets-amp.js
                ├── widgets-static.dev.json
                ├── widgets-config.json
                └── widgets.json
                └── widgets.js
                    1 import dynamic from "next/dynamic"
                    2 import Vactory_academy_favorite_listing from "../components/modules/contrib/academy/AcademyFavoritesWidget.jsx"
                    3 import Vactory_academy_list from "../components/modules/contrib/academy/AcademyListWidget.jsx"
                    4 import Vactory_academy_three_columns from "../components/modules/contrib/academy/AcademyThreeColumnWidget.jsx"
                    5 import Vactory_void_training_espce_prive_lost_password from "../components/modules/contrib/account/AccountLostPasswordWidget.jsx"
                    6 import Vactory_otp_otp from "../components/modules/contrib/account/AccountotpWidget.jsx"
                    7 import Vactory_void_training_espce_prive_single_sign_on from "../components/modules/contrib/account/AccountResetPasswordWidget.jsx"
                    8 import Vactory_banner_default from "../components/modules/contrib/banner/BannerVariantWidget.jsx"
                    9 import Vactory_banner_video from "../components/modules/contrib/banner/BannerVideoWidget.jsx"
                    10 import Vactory_blog_favorite_listing from "../components/modules/contrib/blog/BlogFavoritesWidget.jsx"
                    11 import Vactory_blog_list from "../components/modules/contrib/blog/BlogListWidget.jsx"
                    12 import Vactory_blog_three_columns from "../components/modules/contrib/blog/BlogThreeColumnsWidget.jsx"
                    13 import Vactory_blog_two_columns from "../components/modules/contrib/blog/BlogTwoColumnsWidget.jsx"
                    14 import Vactory_blog_cross_content_blog from "../components/modules/contrib/blog/CrossContentWidget.jsx"
                    15 import Vactory_dynamic_field_decoupled_cookie_compliance_floated from "../components/modules/contrib/cookie-compliance-floated/CookieComplianceFloatedWidget.jsx"
                    16 import Vactory_dynamic_field_decoupled_cookie_compliance_v2 from "../components/modules/contrib/cookie-compliance/CookieComplianceV2Widget.jsx"
                    17 import Vactory_dynamic_field_decoupled_cookie_compliance_v2 from "../components/modules/contrib/cookie-compliance/CookieComplianceV2Widget.jsx"
                    18 import Vactory_event_cross_content_events from "../components/modules/contrib/event/CrossContentWidget.jsx"
                    19 import Vactory_event_list_coming from "../components/modules/contrib/event/EventComingListWidget.jsx"
                    20 import Vactory_event_six_columns_coming from "../components/modules/contrib/event/EventComingSixColumnsWidget.jsx"
                    21 import Vactory_event_six_columns_coming from "../components/modules/contrib/event/EventComingSixColumnsWidget.jsx"
                    22 import Vactory_event_six_columns_coming from "../components/modules/contrib/event/EventComingSixColumnsWidget.jsx"

```

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:

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

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

#### **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/
```

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

---

```
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.

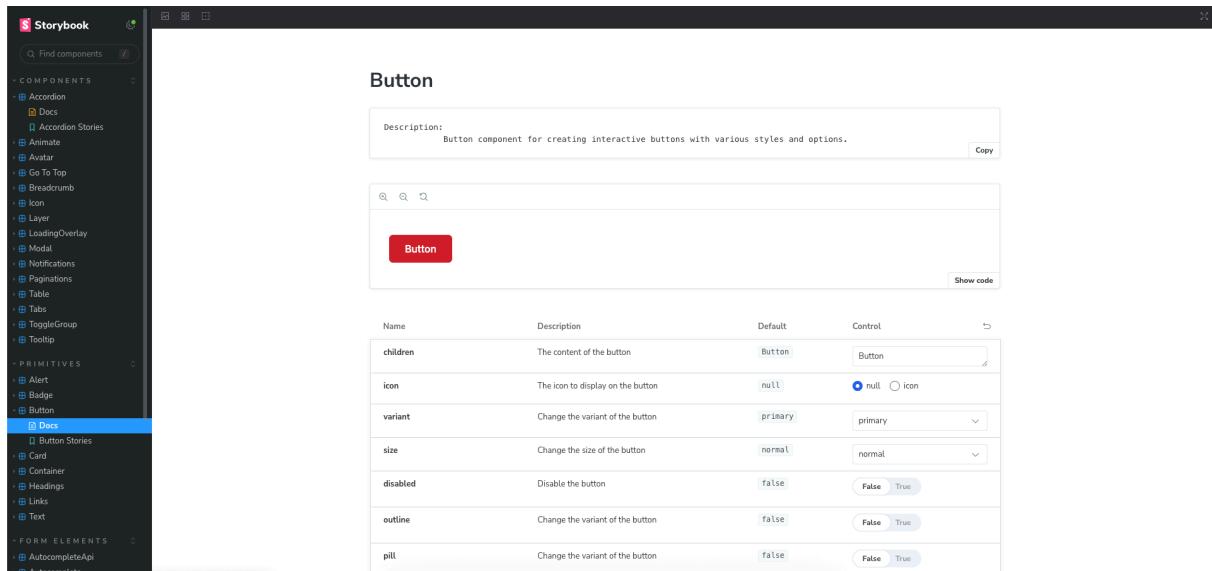


Figure 3.15: Storybook Button Component

#### **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.

```
1 {  
2   "scripts": {  
3     "icons:build": "npx --yes @vactorynext/svg-spreact-cli  
4       ./icons --optimize true --tidy true > ./public/icons.svg"  
5   }  
}
```

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

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

- 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

#### **Candidate Tracking Implementation**

The candidate tracking module is particularly noteworthy for its complexity and user-centric design. It provides a comprehensive overview of the application process, dynamically updating status and available actions:

```
1 export const config = {
2     id: "void_training_candidature:suivi",
3 };
4 const CandidatSuiviWidget = ({ data }) => {
5     const { profile } = useAccount();
```

### **3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION**

---

```
6  const currentStatus = profile?.user?.candidature?.status;
7  //render the data coming from drupal
8 };
```

Listing 3.2: Candidate Tracking Widget Structure

#### **3.6.7 Context-Based Candidature Data**

React Context manages global state for candidature data. This avoids prop drilling and keeps state accessible across deeply nested components.

#### **3.6.8 React Query Integration**

React Query handles server state, caching, and background updates. Custom hooks fetch candidature data, submit quizzes, and manage mutations. This setup keeps the UI responsive and data fresh, even with multiple users and frequent updates.

#### **3.6.9 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.

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

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.

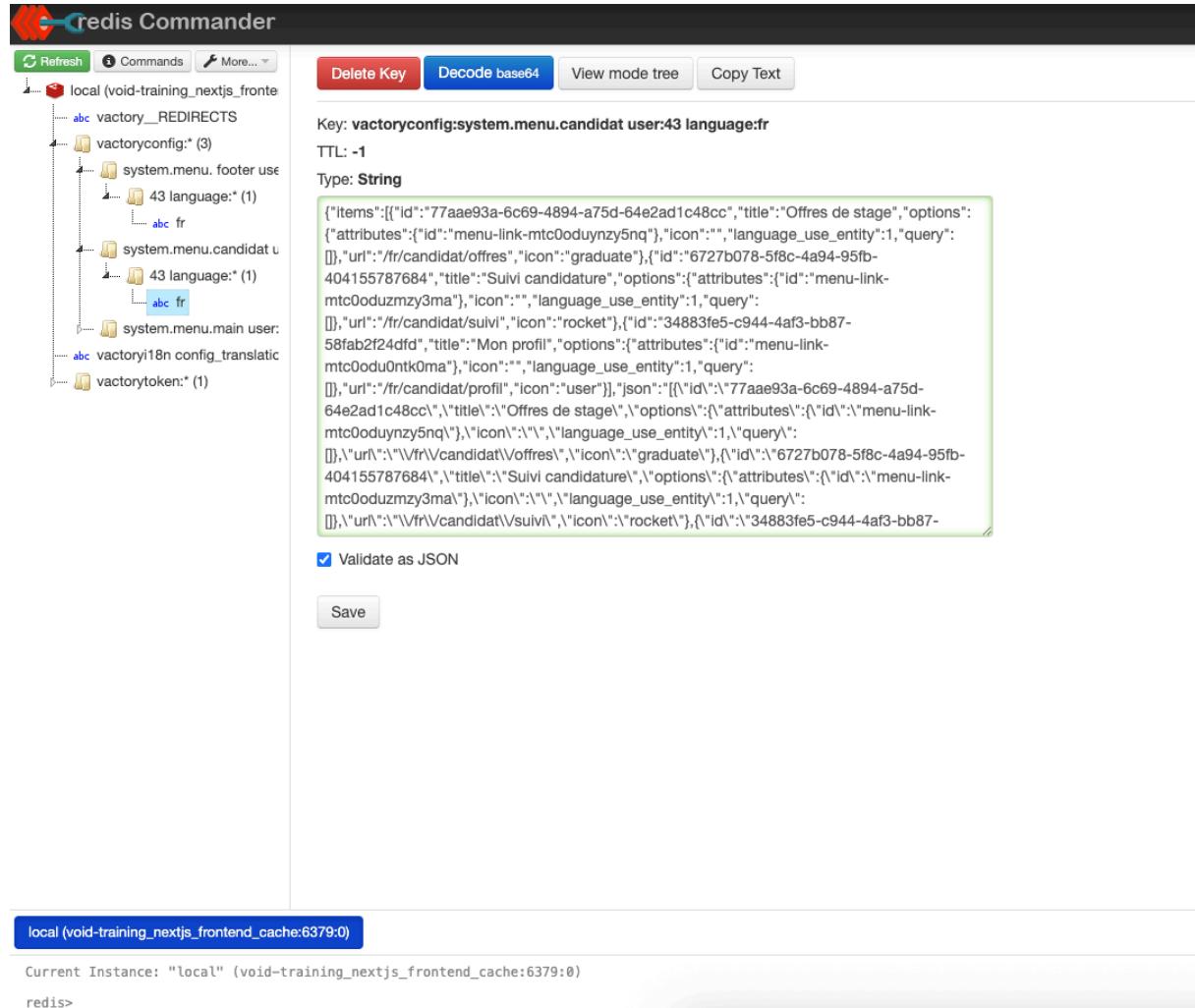


Figure 3.16: Redis Commander UI

#### 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:

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

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.

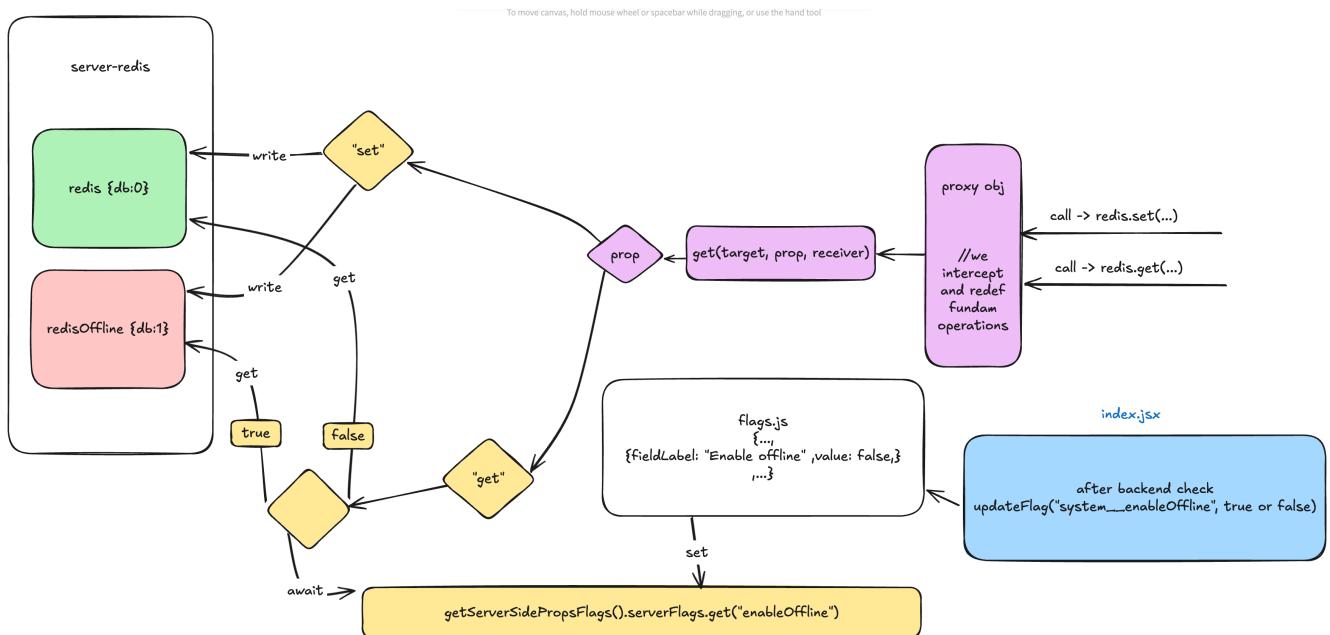


Figure 3.17: Offline Mode Implementation with Redis Cache Schema

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 complete offline mode implementation is detailed in Appendix ??.

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 CI/CD Pipeline Setup**

To ensure smooth development, testing, and deployment workflows, a complete CI/CD (Continuous Integration and Continuous Deployment) pipeline was implemented. The main goals were automation, repeatability, and stability in moving from local development to production with minimal manual intervention.

### **3.7.1 DevOps Stack Overview**

The stack implemented at **VOID** is centered around the following tools and services:

- **Bitbucket**: Source control system for Git repositories and CI pipeline triggers.
- **Docker & Docker Hub** Docker [2025]: Used to containerize services and store built images.
- **EasyPanel**: Lightweight UI-based orchestration platform to manage containers internally on the VOID Mac Studio server.
- **Drush**: CLI for managing Drupal operations (cache clearing, database import/export, config sync, etc.).
- **Traefik** Traefik [2025]: Used as a reverse proxy and load balancer for the application.

### **3.7.2 CI/CD Workflow Description**

The CI/CD pipeline was configured to perform the following steps automatically on every push to the `main` branch:

#### **1. Build Stage:**

- Build Docker images for the frontend (Next.js) and backend (Drupal).
- Run unit and integration tests inside the containers.

#### **2. Push Stage:**

- Push tagged Docker images to the VOID Docker Hub registry.

#### **3. Deploy Stage:**

- Pull the updated image into the EasyPanel internal server.
- Restart containers via EasyPanel to apply the new version.

#### 3.7.3 Bitbucket CI/CD Pipeline Script (Simplified)

```

pipelines:
  branches:
    main:
      - step:
          name: Build and Deploy
          caches:
            - docker
          script:
            - docker build -t void-candidate-frontend ./frontend
            - docker build -t void-drupal-backend ./backend
            - docker push void/void-candidate-frontend
            - docker push void/void-drupal-backend

```

#### 3.7.4 Environment Configuration

All environment-specific secrets and settings (API keys, backend URLs, mail server configuration) were injected securely via:

- Environment variables declared in EasyPanel.
- Bitbucket repository secrets.
- Drupal configuration sync files (`config/sync/*.yml`).

#### 3.7.5 Automated Dependency Updates

A Node.js script was introduced to automate dependency checking and alerting:

- Scan `package.json` and `composer.lock` for outdated dependencies.
- Compare with latest versions from NPM/Packagist.
- Notify the developers via Slack and optionally create a pull request.

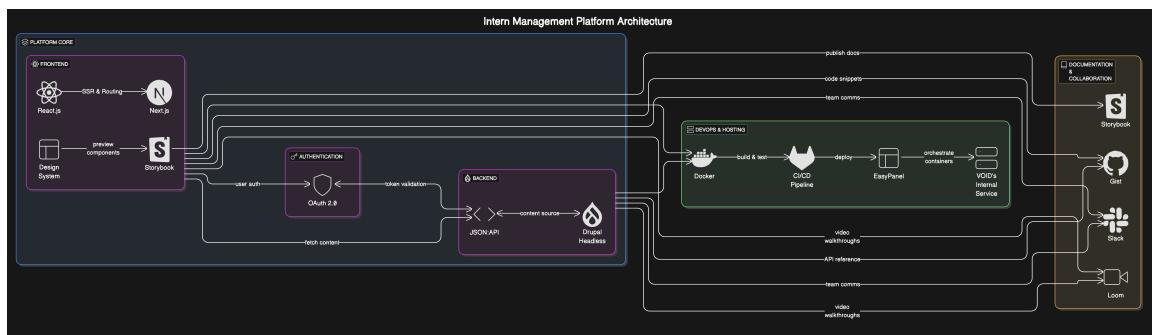


Figure 3.18: CI/CD Pipeline for the Intern Management Platform

#### **3.7.6 Benefits Observed**

The introduction of CI/CD led to the following benefits:

- **60% reduction** in deployment time.
- Elimination of manual production release errors.
- Faster testing feedback and release cycles.

*You should insert a Gantt chart after this section showing the implementation schedule. Let me know if you want one auto-generated in LaTeX or TikZ.*

### **3.8 Testing and Quality Assurance**

Testing played a pivotal role in ensuring the reliability, scalability, and maintainability of the Intern Management Platform. At **VOID**, a Test-Driven Development (TDD) approach was adopted, particularly for the frontend side, to detect regressions early and maintain a high standard of code quality.

#### **3.8.1 Testing Strategy Overview**

The testing efforts were organized across different layers of the application:

- **Unit Testing:** Targeted critical logic functions (e.g., authentication, form validation, state management) using Jest for React components and PHPUnit for backend Drupal hooks.
- **Integration Testing:** Validated end-to-end flows between components such as login, offer application, and training access using testing libraries like React Testing Library.
- **Manual Functional Testing:** Performed during sprints and after deployment on EasyPanel to validate flows such as uploading documents, passing tests, and generating reports.
- **Visual Testing and QA:** Used Storybook for developing and validating UI components in isolation before integrating them into production flows.

#### **3.8.2 Frontend Testing Tools**

- **Jest:** JavaScript testing framework used for snapshot and logic tests.

- **React Testing Library:** Ensures that UI behaves as users expect through DOM-level interaction testing.
- **Storybook:** Enables developers and designers to collaborate on components independently from the app logic.

#### **3.8.3 Backend Testing Tools**

- **PHPUnit:** Unit test framework used for validating custom Drupal modules and hook implementations.
- **Drush Scripts:** Used to automate repeated testing tasks such as database resets, cache rebuilds, and configuration syncs.

#### **3.8.4 Linting and Static Code Analysis**

- **eslint** and **prettier** were integrated into the CI pipeline to maintain consistent code formatting and avoid common pitfalls in React development.
- **PHPStan** was used for static analysis of backend code to detect unreachable code, wrong type usage, or missing return values.

#### **3.8.5 Example Test Case (React + Jest)**

```
import { render, screen } from '@testing-library/react';
import OfferCard from '../components/OfferCard';

test('renders offer title and description', () => {
  render(<OfferCard title="UX Intern" description="Join VOID UX team!" />);
  expect(screen.getByText(/UX Intern/i)).toBeInTheDocument();
  expect(screen.getByText(/VOID UX team/i)).toBeInTheDocument();
});
```

#### **3.8.6 Continuous Testing Integration**

Tests were executed automatically during each build in the CI/CD pipeline. If any unit or integration test failed, the build was aborted and the developer was notified via Slack. This automated feedback loop ensured high code quality and regression prevention.

### 3. CHAPTER 3. MAIN PROJECT IMPLEMENTATION

---

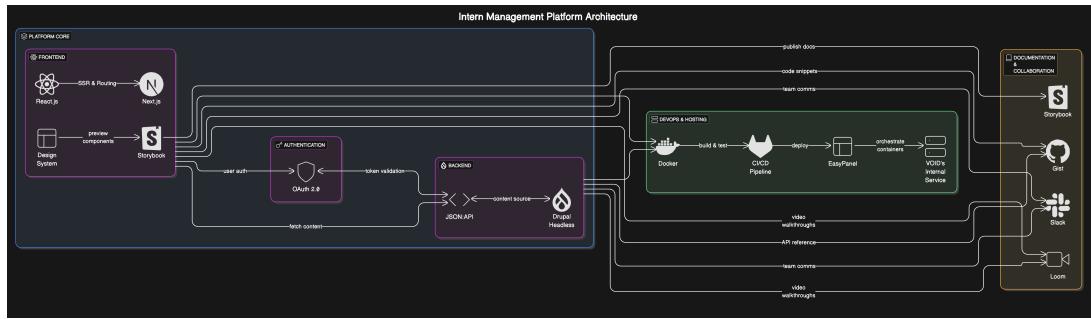


Figure 3.19: Testing Pipeline Integration in CI/CD

#### 3.8.7 Quality Metrics

- **Code coverage:** Maintained above 80% for frontend components.
- **Bug tracking:** Performed via Redmine tickets integrated with Slack.
- **Deployment approval:** Blocked automatically if test stage fails.

# 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:

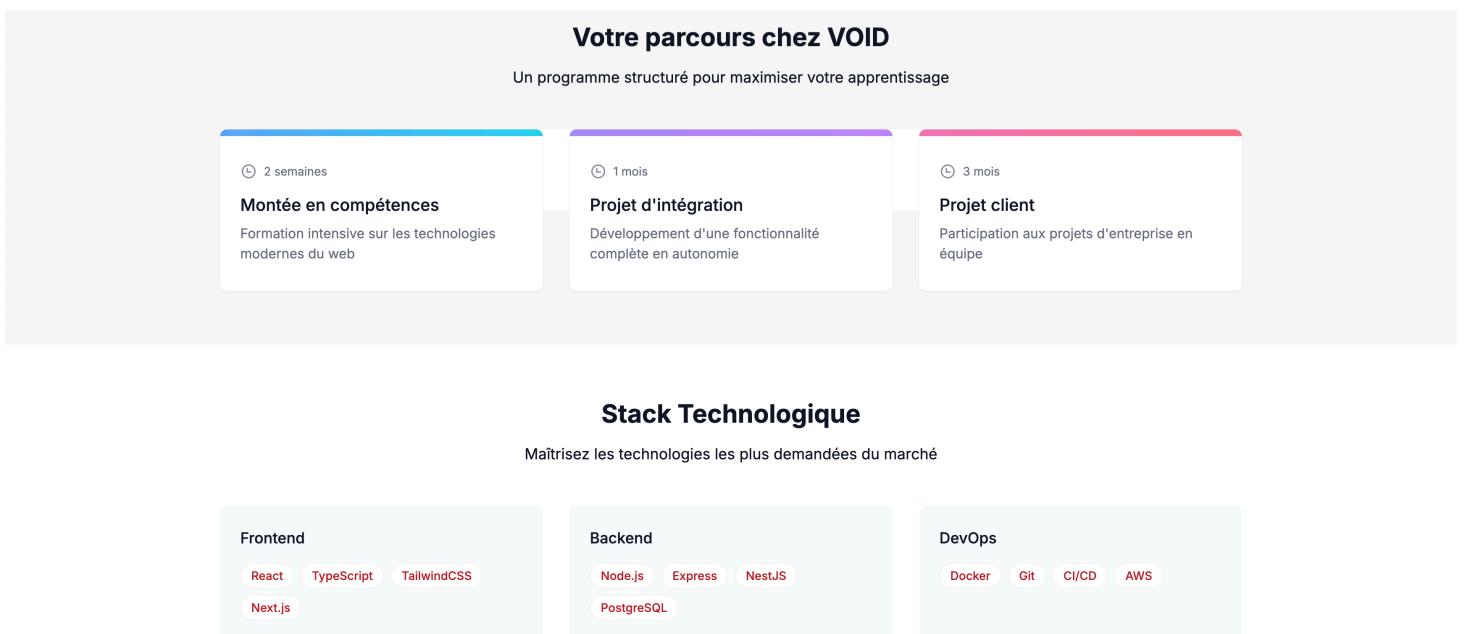


Figure 4.1: Main Page Cards Display

## 4. CHAPTER 4. PROJECT REALIZATION

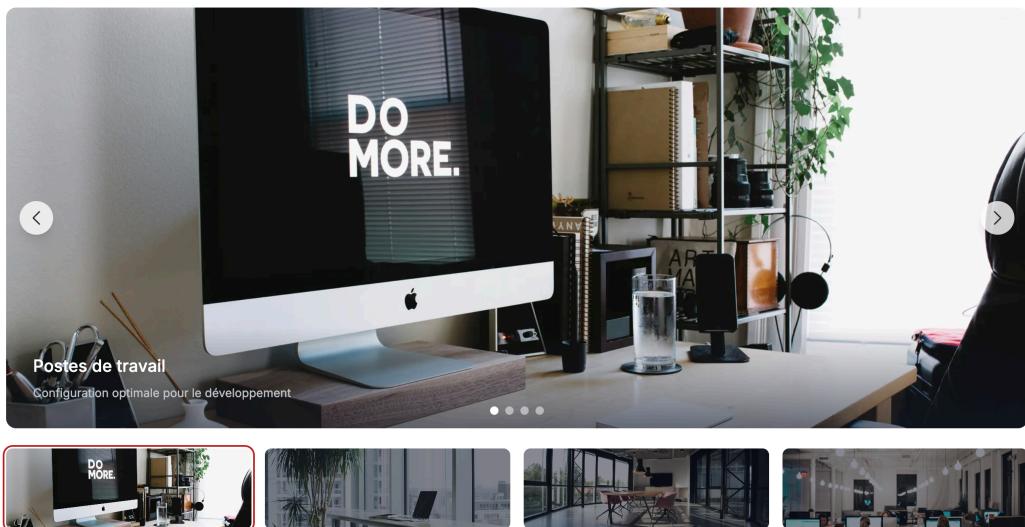


Figure 4.2: Main Page Slider

### 4.2.2 Intern Sprint Management

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

A screenshot of an application interface for intern sprint management. On the left, there's a sidebar with navigation links like "Tableau de bord", "Formations" (which is highlighted), "Ressources", "Mon VPS", "Projet", and "Paramètres". A red box highlights the "Formations" link. A pink box contains a message about needing help and a "Contacter le support" button. At the bottom, a user profile for "Hamza Massir" is shown. The main content area is titled "Parcours de formation" and describes a 10-week program from 03/02/2025 to 31/07/2025. It shows two sprints: "Sprint 1: Base Frontend" (Terminé, 100% completed) and "Sprint 2: Backend PHP" (En cours, 45% completed). Each sprint section includes a timeline bar, a list of topics covered, and a list of practical projects.

Figure 4.3: Intern Sprint Listing

The sprint listing shows:

## 4. CHAPTER 4. PROJECT REALIZATION

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

The screenshot shows the 'Sprint 1: Base Frontend' details page. At the top, there's a navigation bar with a back arrow labeled 'Retour aux sprints', the title 'Sprint 1: Base Frontend', a progress bar showing '1 semaine' and '50% complété', and a green 'Terminé' button. On the left, a sidebar includes links for 'Tableau de bord', 'Formations' (which is highlighted in pink), 'Ressources', 'Mon VPS', 'Projet', and 'Paramètres'. A 'Besoin d'aide?' section offers support via email or chat.

**Sprint — HTML5 & CSS3 Avancé**

**Introduction**

Ce sprint a pour but de renforcer vos compétences en développement front-end, en explorant les fonctionnalités avancées d'HTML5 et de CSS3. Vous mettrez en pratique des techniques modernes de mise en page, d'accessibilité et d'animation tout en respectant les standards du web.

**Partie 1 — HTML5 avancé**

**Balises séquentielles**

HTML5 introduit plusieurs balises qui donnent du sens au contenu :

- `<article>` , `<section>` , `<nav>` , `<header>` , `<footer>` , `<aside>`
- Objectif : structurer la page de façon logique pour les moteurs de recherche et les lecteurs d'écran.

**Formulaires avancés**

- Nouvelles balises : `<input type="date">` , `<input type="email">` , `<datalist>` , `<output>` , etc.
- Attributs utiles : `required` , `pattern` , `min` , `max` , `step` , `autocomplete`

**Accessibilité (a11y)**

- Utiliser des balises adaptées : `<label>` , `<fieldset>` , `<legend>` , etc.
- Ajouter des attributs ARIA si nécessaire ( `aria-label` , `aria-hidden` , etc.)

**Partie 2 — CSS3 avancé**

**Flexbox et Grid Layout**

- Flexbox : idéal pour les composants à une seule dimension (ligne ou colonne)

**Objectifs d'apprentissage**

- ✓ HTML5 & CSS3 avancé
- ✓ React.js fondamentaux
- ✓ Tailwind CSS
- ✓ TypeScript bases
- ✓ JavaScript ES6+
- ✓ Responsive Design
- ✓ Git & GitHub
- ✓ Tests unitaires Jest

**Progression du sprint** **50%**

4 objectifs complétés / 4 restants

**Livrable soumis**

Sprint terminé avec succès !  
<https://www.google.com>

**Planning de la semaine**

Lundi - Mardi	Mercredi - Jeudi	Théorie & Concepts	Projets pratiques	Vendredi	Évaluation & Review

Figure 4.4: 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. CHAPTER 4. PROJECT REALIZATION

### 4.2.3 Learning Resources

A dedicated resources page provides additional learning materials:

The screenshot shows the 'Ressources d'apprentissage' (Learning Resources) section of the Void Training platform. The sidebar on the left includes links for 'Tableau de bord', 'Formations', 'Ressources' (which is highlighted in pink), 'Mon VPS', 'Projet', and 'Paramètres'. A 'Besoin d'aide?' section offers support contact. The main content area features a search bar and filters for 'Tout', 'Frontend', 'Backend', 'DevOps', and 'Drupal'. Below this, a section titled 'Vidéos recommandées' (Recommended Videos) displays three video thumbnails: 'Docker pour les développeurs' (45:20), 'React.js - Les Hooks en détail' (32:15), and 'Architecture Drupal 10' (28:45). At the bottom, there's a GitHub logo and a stack of books with one titled 'STUNNING CSS'.

Figure 4.5: Learning Resources Page

## 4. CHAPTER 4. PROJECT REALIZATION

The screenshot shows a user interface for learning resources. On the left, there's a sidebar with a 'Void Training' logo, a 'Tableau de bord' link, 'Formations' (highlighted in pink), 'Ressources' (highlighted in pink), 'Mon VPS', 'Projet', and 'Paramètres'. A 'Besoin d'aide?' section offers support contact. Below it is a user profile for 'Hamza Massir' (Stagiaire). The main content area has several sections:

- Plateformes de formation**:
  - Udemy**:  
Plateforme de formation en ligne  
Identifiant: void.learning  
Mot de passe: .....  
Cours disponibles:
    - React - The Complete Guide
    - Drupal 10 Development
    - Advanced PHP Architecture
    - Docker & Kubernetes Masterclass
  - Pluralsight**:  
Formation développeur avancée  
Identifiant: void.tech  
Mot de passe: .....  
Cours disponibles:
    - Design Patterns in PHP
    - Docker & Kubernetes
    - Advanced React Patterns
    - API Design in Node.js
  - Frontend Masters**:  
Formations Frontend avancées  
Identifiant: void.frontend  
Mot de passe: .....  
Cours disponibles:
    - Advanced CSS Layouts
    - TypeScript in React
    - State Management
    - Web Performance
- Documentation interne**:  
Guide d'architecture VOID (PDF, 2.4 MB)
- Liens utiles**:  
PHP - The Right Way (Guide des bonnes pratiques PHP)

Figure 4.6: Learning Resources Page (Accounts)

The resources section offers:

- Technical documentation
- Tutorial videos
- Code examples
- Best practices guides

### 4.2.4 Project Management

Each intern has a dedicated project page:

## 4. CHAPTER 4. PROJECT REALIZATION

The screenshot shows the Void Training platform's project management interface. On the left, a sidebar includes links for Tableau de bord, Formations, Ressources, Mon VPS, Projet (which is selected and highlighted in blue), and Paramètres. A red callout box on the sidebar provides support information: "Besoin d'aide ? Notre équipe est là pour vous aider dans votre parcours. Contacter le support →". Below the sidebar, a user profile for Hamza Massir (Stage) is shown.

The main content area displays the "Projet de fin de formation" for the "Application de gestion de flotte de véhicules". Key details include:

- Description:** Développement d'une application web permettant la gestion d'une flotte de véhicules d'entreprise.
- Durée:** 6 semaines.
- Stack technique:** React, PHP 8, Drupal 10, Docker.
- Livrables attendus:** Application web responsive, API RESTful, Documentation technique, Tests unitaires et fonctionnels.

The "Tâches en cours" section lists two tasks:

- #1234 Mise en place de l'architecture Docker (En cours): Configuration des conteneurs Docker pour l'environnement de développement. Estimated 8h, due 20/02/2024.
- #1235 Développement du module d'authentification (À faire): Implémentation du système d'authentification avec gestion des rôles. Estimated 16h, due 25/02/2024.

On the right side, a sidebar for Zineb Naji (Lead Developer) shows contact info (z.naji@void.fr), availability (Lundi: 14h - 18h; Jeudi: 10h - 12h), and the next meeting (jeudi 15 février à 14:00). It also displays a progress bar (34% complete, 12 tasks out of 35), current sprint (Sprint 2), end date (30 mars 2024), and remaining time (-441 jours).

Figure 4.7: Project Management Page

The project page shows:

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

### 4.2.5 User Profile

Interns can manage their personal information:

## **4. CHAPTER 4. PROJECT REALIZATION**

---

The screenshot shows the 'Paramètres' (Settings) page of the Void Training platform. The left sidebar includes links for 'Tableau de bord', 'Formations', 'Ressources', 'Mon VPS', 'Projet', and 'Paramètres'. A red box highlights the 'Paramètres' link. A pink box on the left contains a 'Besoin d'aide?' section with a message about support and a 'Contacter le support' button. The main content area has tabs for 'Profil', 'Notifications', 'Préférences', 'Sécurité', and 'Documents', with 'Profil' selected. It displays personal information: Prénom (Hamza), Nom (Massir), Email (h.massir@void.fr), and Téléphone (+212 6 00 00 00 00). A placeholder for an 'Avatar' is shown with a 'Changer l'avatar' button. The bottom left shows a user card for 'Hamza Massir' (Stage). The URL in the address bar is https://void-training.com/parametres.

Figure 4.8: User Profile Page

The profile page allows:

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

### **4.2.6 Document Management**

A secure space for personal documents:

## 4. CHAPTER 4. PROJECT REALIZATION

The screenshot shows the 'Paramètres' (Parameters) section of the Void Training platform. On the left, there's a sidebar with links like 'Tableau de bord', 'Formations', 'Ressources', 'Mon VPS', 'Projet', and 'Paramètres'. A 'Besoin d'aide?' box offers support contact. On the right, the 'Documents' tab is selected, showing four document entries:

- Convention de stage**: Status 'Document validé', last updated 15/01/2024. Actions: 'Voir' and 'Mettre à jour'.
- Attestation d'assurance**: Status 'Document manquant'. Action: 'Ajouter'.
- Copie CIN**: Status 'Document validé', last updated 10/01/2024. Actions: 'Voir' and 'Mettre à jour'.
- Attestation RIB**: Status 'En attente de validation', last updated 10/02/2024. Actions: 'Voir' and 'Mettre à jour'.

A pink box at the bottom contains important information about document requirements:

- Les documents doivent être au format PDF
- Taille maximale : 5 Mo par document
- Les documents sont vérifiés par l'administration sous 48h

In the bottom-left corner of the sidebar, it says 'Hamza Massir Stagiaire'.

Figure 4.9: Document Management Page

The documents section provides:

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

# 5 Discussion and Future Improvements

## 5.1 Introduction

This chapter provides a comprehensive evaluation and analysis of the Intern Management Platform developed for VOID Digital Agency. The discussion focuses on the significance of the achieved results, the impact of the implemented solution, identified limitations, and potential areas for future enhancement. The analysis encompasses both technical and operational perspectives, examining how the platform addresses the initial challenges while identifying opportunities for continued improvement.

## 5.2 Significance of the Achieved Results

### 5.2.1 Process Transformation and Efficiency Gains

The implementation of the Intern Management Platform has fundamentally transformed VOID Digital Agency's internship management process. The transition from a manual, fragmented approach to a centralized, automated system represents a significant organizational advancement:

- **Application Processing:** The platform successfully handles a significant number of applications per recruitment cycle, compared to the previous manual processing that was prone to delays and human error.
- **Automated Assessment:** The integration of automatic technical test assignment and grading has significantly reduced evaluation time, allowing recruiters to focus on higher-value tasks while eliminating the cost of using external test platforms like CoderPad.
- **Standardized Workflow:** The implementation of a structured candidate journey ensures consistent evaluation criteria and fair assessment across all applicants.
- **Real-time Tracking:** Both candidates and administrators benefit from transparent status tracking, reducing inquiries and improving communication efficiency.

### **5.2.2 Technical Architecture Success**

The chosen technical architecture has proven its effectiveness in meeting both current and anticipated future requirements:

- **Scalability:** The headless Drupal 10 backend with Next.js frontend successfully handles concurrent user loads during peak application periods.
- **Maintainability:** The component-driven development approach using Storybook has facilitated consistent UI development and simplified maintenance procedures.
- **Integration Capabilities:** The JSON:API implementation enables seamless communication between frontend and backend, supporting future expansion and third-party integrations.
- **DevOps Excellence:** The containerized deployment using Docker and EasyPanel has demonstrated reliable performance with automated CI/CD pipelines reducing deployment risks.

### **5.2.3 User Experience Enhancement**

The platform has significantly improved the user experience for all stakeholder groups:

- **Candidate Satisfaction:** Initial feedback from candidates indicates high satisfaction with the streamlined application process, though formal metrics are still being collected.
- **Administrative Efficiency:** The recruitment team reports significant time savings in routine task management, enabling greater focus on candidate evaluation.
- **Transparency:** The improved communication system has noticeably reduced the volume of candidate inquiries.

## **5.3 Current Platform Limitations**

### **5.3.1 Candidate Space Limitations**

#### **User Interface and Experience Constraints**

Several areas within the candidate space require attention for optimal user experience:

- **Mobile Responsiveness:** While functional, the mobile experience could be enhanced for candidates primarily using smartphones for applications.
- **Multi-language Support:** The system is primarily designed for French-speaking candidates, limiting international applicant accessibility.

## **5. CHAPTER 5. DISCUSSION AND FUTURE IMPROVEMENTS**

---

### **Technical Assessment Limitations**

The technical testing component, while functional, presents several areas for improvement:

- **Cheat Detection Mechanisms:** The current system lacks robust anti-cheating measures such as:
  - Screen monitoring capabilities
  - Candidate webcam monitoring
  - Candidate Audio monitoring
- **Question Pool Limitations:** We currently have only 200 question in each specialization. The static question database may lead to answer sharing among candidates in subsequent recruitment cycles.
- **Challenge Pool Limitations:** We currently have only 3 challenges in each specialization. The static challenge database may lead to answer sharing among candidates in subsequent recruitment cycles.
- **Manual Challenge Review:** We currently have no automated challenge review system. The challenge review is done manually by the recruiters.
- **Manual Video Interview Review:** We currently have no automated video interview review system. The video interview review is done manually by the recruiters.

### **Communication and Feedback Systems**

Current communication mechanisms could be enhanced:

- **Detailed Feedback:** Limited constructive feedback provided to unsuccessful candidates for future improvement.
- **Interview Scheduling:** Manual coordination required for interview scheduling, lacking integration with calendar systems, and no automatic interview scheduling system.

#### **5.3.2 Administrative Interface Limitations**

The backoffice interface presents usability challenges for non-technical administrative staff:

- **Complex Navigation:** The current Drupal administrative interface requires technical knowledge and may overwhelm non-technical users.
- **Customization Limitations:** Limited ability to customize the interface layout and workflow to match specific administrative preferences.

## **5. CHAPTER 5. DISCUSSION AND FUTURE IMPROVEMENTS**

---

- **Learning Curve:** New administrators require extensive training to effectively utilize all platform features.
- **Visual Design:** The interface lacks modern, intuitive design elements that could improve user adoption and efficiency.
- **Data Export Limitations:** We only have the ability to export the data in CSV format.

### **5.4 Proposed Future Improvements**

#### **5.4.1 Administrative Interface Enhancement**

##### **Custom Administrative Dashboard**

Development of a user-friendly administrative interface specifically designed for non-technical users:

- **Simplified Navigation:** Intuitive menu structure with role-based access control
- **Visual Workflow Management:** Drag-and-drop interface for managing candidate pipeline stages
- **Customizable Widgets:** Personalized dashboard components for different administrative roles
- **Guided Workflows:** Step-by-step wizards for complex administrative tasks

#### **5.4.2 Enhanced Security and Anti-Cheating Measures**

##### **Advanced Proctoring System**

Implementation of comprehensive examination security features:

- **Browser Lockdown:** Prevent access to external resources during assessments
- **Webcam Monitoring:** Optional video proctoring for high-stakes assessments
- **Screen Recording:** Comprehensive session recording for review purposes

#### **5.4.3 Candidate Experience Improvements**

##### **Enhanced User Interface**

Comprehensive redesign of the candidate-facing interface:

## **5. CHAPTER 5. DISCUSSION AND FUTURE IMPROVEMENTS**

---

- **Progressive Web App (PWA):** Mobile-first design with offline capability
- **Multilingual Support:** Internationalization for French, English, and Arabic languages

### **Advanced Assessment Capabilities**

Implementation of more sophisticated evaluation methods:

- **Coding Sandbox:** Integrated development environment for practical programming assessments
- **Portfolio Review System:** Structured evaluation framework for candidate work samples
- **Video Interview Integration:** Built-in video conferencing for remote interviews

## **5.5 Impact Assessment and Validation**

### **5.5.1 Quantitative Metrics**

The platform's success can be measured through several key performance indicators:

- **Processing Efficiency:** Significant reduction in application processing time through automated workflows
- **System Reliability:** High uptime maintained during peak recruitment periods
- **User Adoption:** Strong administrative staff adoption following deployment
- **Cost Effectiveness:** Notable reduction in recruitment-related administrative costs

### **5.5.2 Qualitative Benefits**

Beyond measurable metrics, the platform has delivered significant qualitative improvements:

- **Professional Image:** Enhanced VOID Digital Agency's reputation as a technology-forward organization
- **Candidate Experience:** Improved perception of the recruitment process among applicants
- **Staff Satisfaction:** Reduced administrative burden and improved job satisfaction for recruitment team
- **Scalability Foundation:** Established infrastructure for future growth and expansion

## **5.6 Summary**

The Intern Management Platform represents a successful digital transformation initiative that has significantly improved VOID Digital Agency's recruitment processes. While the current implementation addresses the primary objectives and delivers substantial value, the identified limitations provide clear direction for future development efforts. The platform's modular architecture and solid technical foundation ensure that proposed enhancements can be implemented incrementally, allowing for continuous improvement and adaptation to evolving organizational needs.

The success of this project demonstrates the value of thoughtful system design, user-centered development approaches, and comprehensive testing methodologies. Future iterations should focus on enhancing user experience, implementing advanced security measures, and leveraging emerging technologies to maintain competitive advantage in the talent acquisition landscape.

# Conclusion

The Intern Management Platform transformed how VOID Digital Agency handles recruitment. Instead of managing 600+ applications through scattered emails and spreadsheets, they now have a streamlined system that works for everyone involved. This internship experience at VOID Digital Agency provided far more than just technical skills development. Working on a real project with actual business impact taught valuable lessons about software architecture, user experience design, and collaborative development practices. The challenges faced from debugging complex API integrations to optimizing Docker configurations built confidence in tackling unfamiliar technologies and problem-solving under pressure. Beyond the technical achievements, this experience demonstrated the importance of understanding user needs, communicating effectively with stakeholders, and writing maintainable code that others can work with. These skills, combined with hands-on experience in modern development practices like CI/CD, provide a solid foundation for entering the professional software development field. The internship proved that good software development is about more than writing code; it's about creating solutions that genuinely help people do their jobs better and more efficiently.

## Appendix B: 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

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.

Docker. Multi-stage builds, 2025. URL <https://docs.docker.com/build/building/multi-stage/>.

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