Conception Architecturale

ClearSprint AI

Présentation du Système d'Information

Ashref Ben Abdallah
MPGL 2

Année Universitaire 2025-2026
POLYTECH INTL
Enseignant : Mme A. TAYACHI

Table des matières

1 Partie 1 : Analyse

1.1 Présentation Générale

ClearSprint AI est une plateforme web de gestion de projets agiles utilisant l'IA pour automatiser la planification.

Objectifs : (1) Génération automatique de tickets via GPT-4, (2) Synchronisation bidirectionnelle avec Jira, (3) Hub centralisé de gestion, (4) Réduction de 70% du temps de planification.

Stack Technique : Next.js 16 + React 19, PostgreSQL (Neon), Minio S3, OpenAI GP T-4, Better-Auth.

1.2 Exigences Fonctionnelles

Acteurs : Utilisateur (PM/Dev), Système Jira.

Module 1 - Authentication : (1) Sign Up/Sign In via Better-Auth

Module 2 - Project Management : (2) Create Project, (3) View Dashboard, (4) Upload PRD

Module 3 - Ticket Management : (5) Generate Tickets (AI), (6) Edit Tickets, (7) Manage Status

Module 4 - Jira Integration : (8) Connect Jira (OAuth 2.0), (9) Sync Tickets, (10) Pull Projects

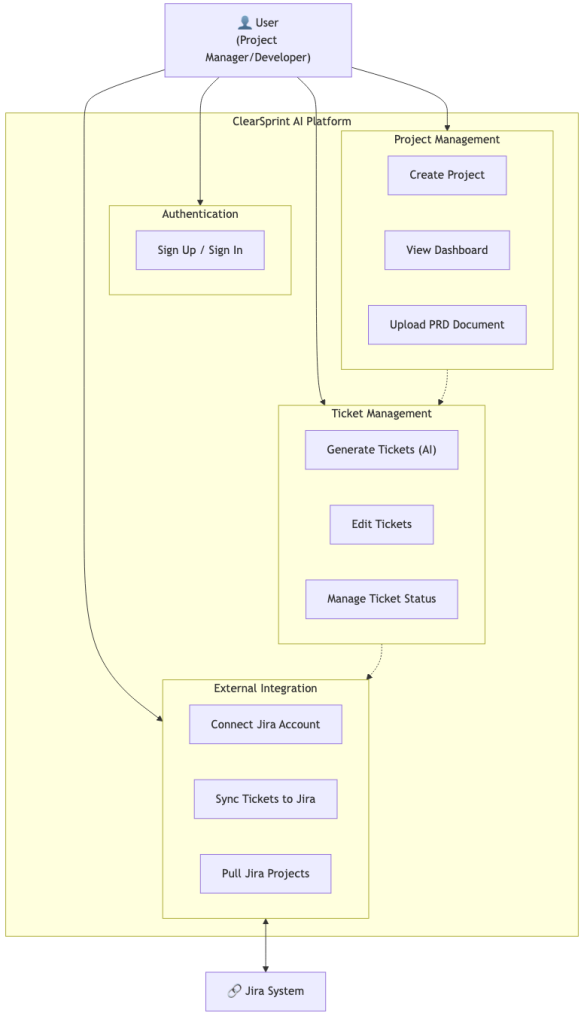


FIGURE 1 – Cas d'Utilisation - ClearSprint AI

2 Partie 2 : Conception Architecturale

L'architecture suit le modèle **4+1 Vues** de Philippe Kruchten.

2.1 Vue Logique - Architecture en Couches

Style : Architecture en Couches avec Modules Feature-Driven

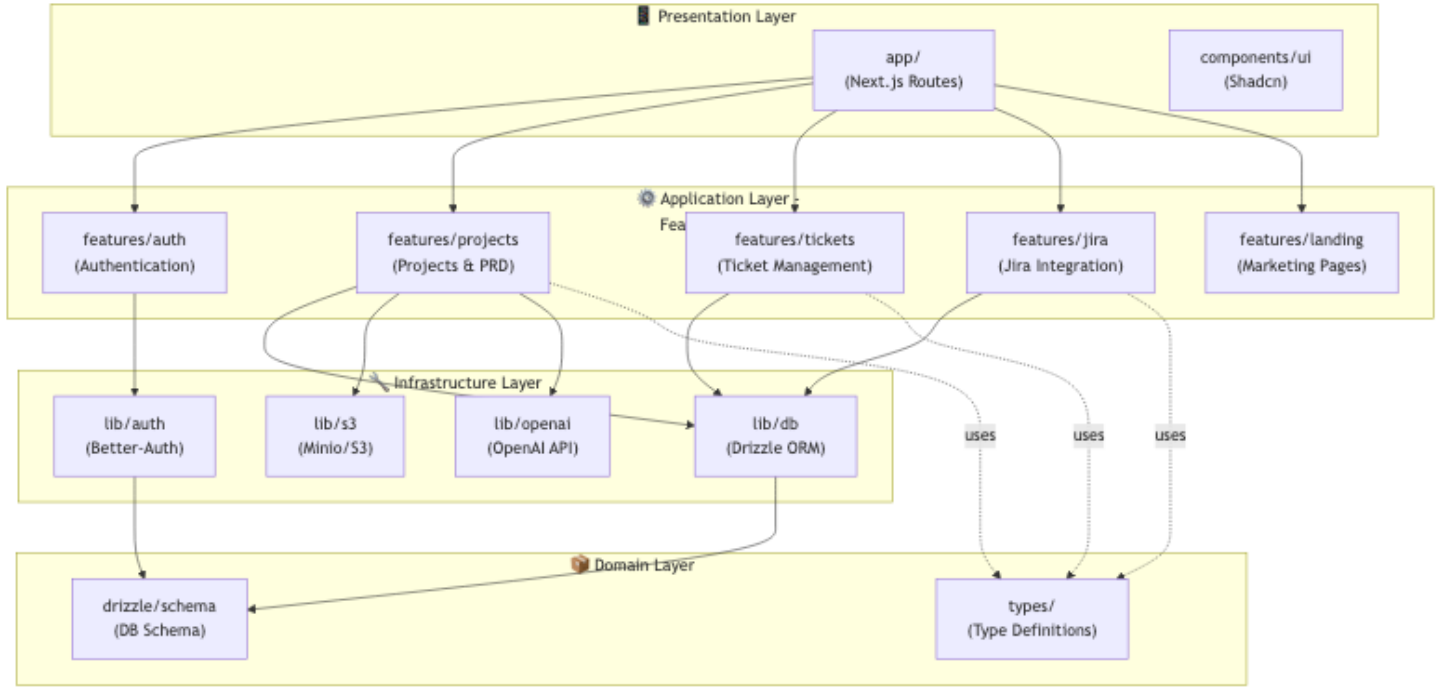


FIGURE 2 – Vue Logique - Couches

Couche Présentation : `app/` (Next.js App Router), `components/ui/` (Shadcn/ui)

Couche Application : `features/auth` (Auth), `features/projects` (Projects + PRD), `features/tickets` (Tickets + AI), `features/jira` (Jira Integration)

Couche Domaine : `types/` (TypeScript), `drizzle/schema` (DB Schema)

Couche Infrastructure : `lib/db` (Drizzle ORM), `lib/auth` (Better-Auth), `lib/s3` (Minio), `lib/openai` (GPT-4)

Flux : Présentation → Application → Infrastructure → Domaine

2.2 Vue Implémentation - Monolithe Moderne

Style : Monolithe Moderne Next.js 16
Client Tier : Web Browser (React UI)
App Server : Frontend (React Server Components) + Backend (Server Actions + API Routes)
Data & Services : PostgreSQL (Neon), Minio S3, OpenAI API, Jira REST API

2.3 Vue Déploiement - 3-Tiers Cloud

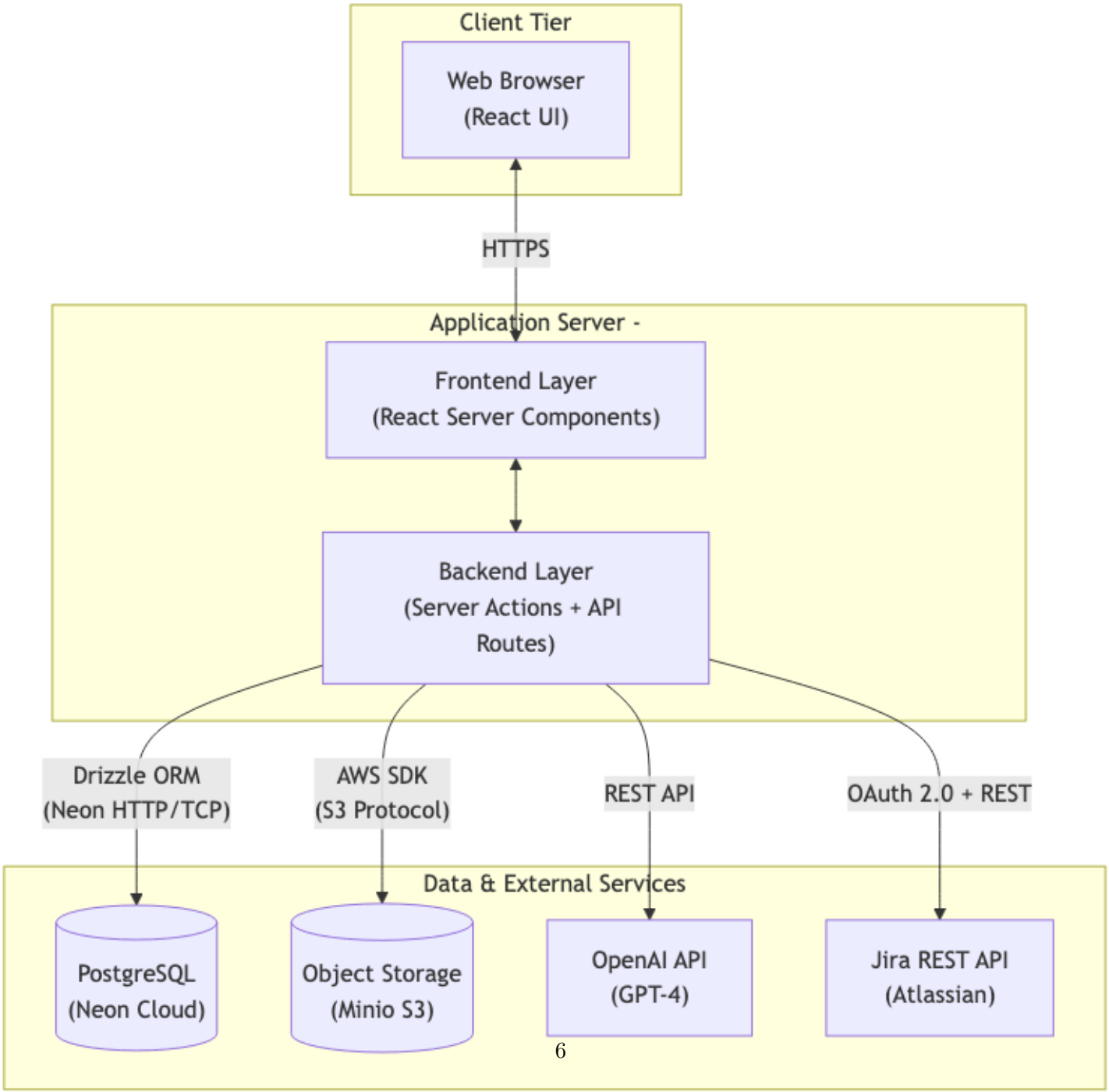
Style : Architecture 3-Tiers (Vercel)
Tier 1 - Client : Web Browser (HTTPS/443 + TLS 1.3)
Tier 2 - Application : Next.js Server (Serverless) + Vercel CDN (Auto-scaling)
Tier 3 - Data : Neon PostgreSQL (Serverless) + Minio S3
External : Jira Cloud + OpenAI Platform

2.4 Évaluation de l'Architecture

Critère	Layered	Mono	3-Tier	Score
Development	Élevé	Élevé	Bon	8.3/10
Deployment	Moyen	Moyen	Élevé	7.0/10
Testability	Élevé	Bon	Bon	8.0/10
Performance	Bon	Élevé	Élevé	8.7/10
Scalability	Moyen	Limité	Élevé	6.7/10

Table 1 – Évaluation Comparative

Architecture en Couches : Development (9/10), Deployment (7/10), Testability (9/10), Performance (8/10), Scalability (6/10)
Monolithique : Development (9/10), Deployment (6/10), Testability (8/10), Performance (9/10), Scalability (5/10)
3-Tiers : Development (7/10), Deployment (8/10), Testability (7/10), Performance (9/10), Scalability (9/10)
Conclusion : Compromis optimal MVP avec évolution vers microservices si nécessaire.


Client Tier
Web Browser
(React UI)
HTTPS
Application Server -
Frontend Layer
(React Server Components)
Backend Layer
(Server Actions + API
Routes)
Drizzle ORM
(Neon HTTP/TCP)
AWS SDK
(S3 Protocol)
REST API
OAuth 2.0 + REST
Data & External Services
PostgreSQL
(Neon Cloud)
Object Storage
(Minio S3)
OpenAI API
(GPT-4)
Jira REST API
(Atlassian)

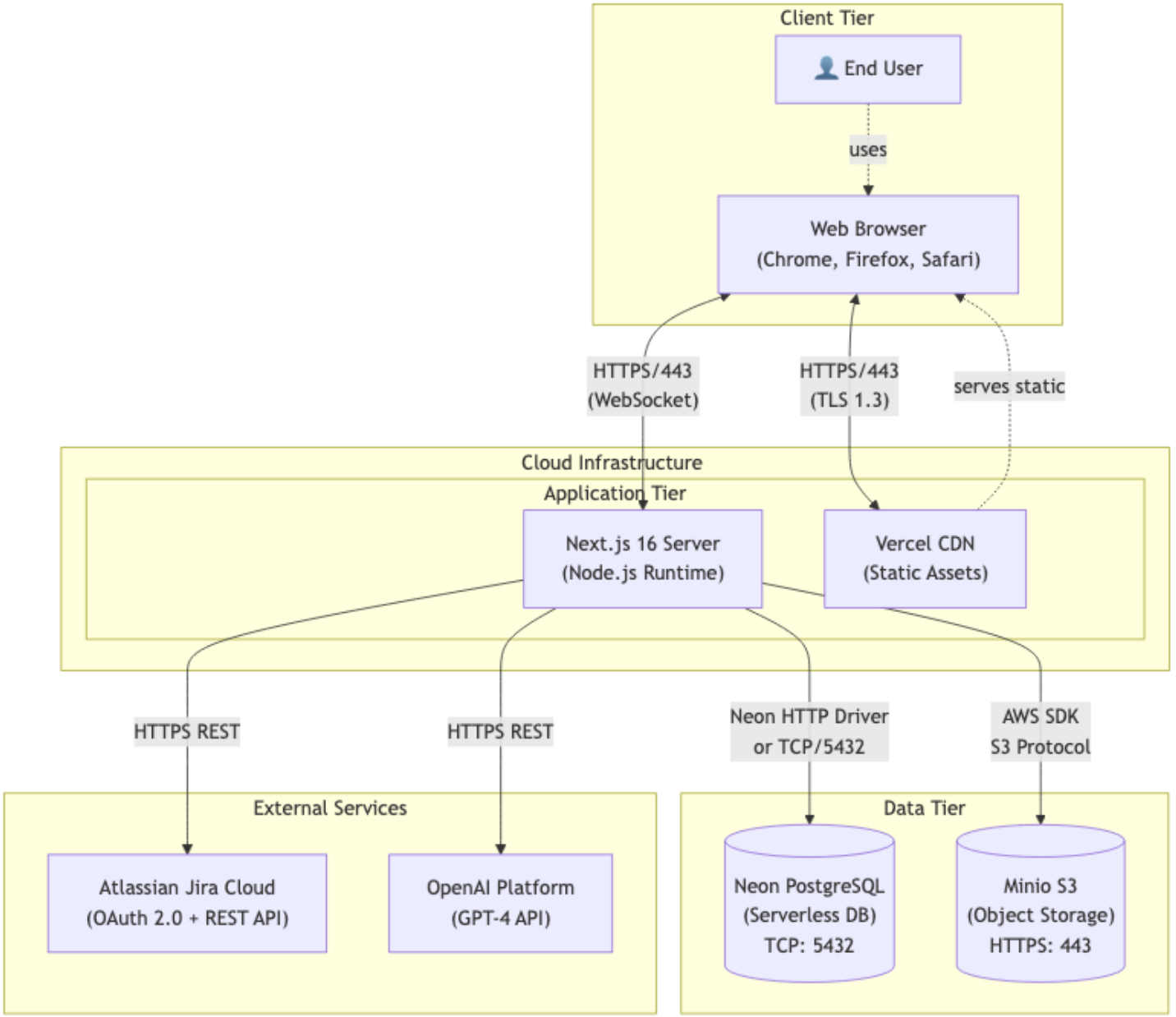


FIGURE 4 – Vue Déploiement

3 Partie 3 : Réalisation

3.1 Stack Technologique

Langages : TypeScript 5.9 (strict), SQL

Frontend : Next.js 16, React 19.2, Tailwind CSS v4, Shadcn/ui, Framer Motion

Backend : Server Actions, API Routes, Better-Auth 1.3

Database : PostgreSQL (Neon), Drizzle ORM 0.44

Storage & AI : Minio 8.0, OpenAI SDK 6.9, PDF Parse

DevOps : Bun, Biome 2.3, Git, Vercel

3.2 Mapping Code ↔ Architecture

Logical View : `app/` → `app/(dashboard)/`, `features/projects` → `actions/`, `lib/db` → `lib/db.ts`, `drizzle/schema` → `drizzle/schema.ts`

Deployment View : Dev : localhost :3000 + Neon + Minio local — Prod : Vercel + Neon Cloud + Minio Cloud

3.3 Démonstration

Flux Principal : (1) Auth via `/auth/signin`, (2) Create Project (Server Action), (3) Upload PRD (Minio S3 + text extraction), (4) Generate Tickets (GPT-4 → JSON → DB batch insert), (5) Sync Jira (OAuth 2.0 + REST API + jiraId tracking)

Fonctionnalités Opérationnelles : Auth multi-providers, CRUD projets/tickets, Upload PRD, Génération IA, Sync Jira, Dashboard responsive, Performance Lighthouse 95+

4 Conclusion

Synthèse : Architecture moderne (Layered + Feature-Driven), Type-safety (TS strict), Performance (RSC + CDN), IA intégrée (GPT-4), Interopérabilité (Jira OAuth)

Évolutions : Court terme (Notifications, Templates, Export), Moyen terme (GitHub/Linear, Analytics, Mobile), Long terme (Microservices, Multi-tenancy, Fine-tuning IA)

Remerciements : Mme A. TAYACHI et l'équipe POLYTECH INTL.

Ashref Ben Abdallah
MPGL 2 - Architecture Logicielle - 2025-2026