

**Faculté des Sciences et Techniques de Tanger (F.S.T.T)**

**Département : Génie Informatique**

**LST – IDAI : Développement Web**

# **RAPPORT DE PROJET**

**Plateforme de Gestion et de Réservation des Ressources d'un Data Center**

**“DataHub”**



- **Encadré par:**

**Pr. Yasyn Elyusufi**

**Pr. M'hamed AIT KBIR**

- **Réalisé par:**

Dany Homam

El Hajioui Houssam

El Bourmaki Salim

Farssi Fatima Zahraa

**Année Universitaire : 2025 / 2026**

# **SOMMAIRE**

## **REMERCIEMENTS**

- 1. INTRODUCTION GENERALE & PROBLEMATIQUE**
- 2. ARCHITECTURE TECHNIQUE ET ORGANISATION DU CODE**
- 3. ÉTAT DE L'ART & CHOIX TECHNOLOGIQUES**
- 4. OBJECTIFS ET PERIMETRE DU PROJET**
- 5. ANALYSE DES BESOINS & SPECIFICATIONS (PAR PROFILS)**
- 6. CONCEPTION & MODELISATION (UML + MCD/MLD)**
- 7. REALISATION DES MODULES (ECRANS & LOGIQUE METIER)**
- 8. SECURITE, CONTROLE D'ACCES & INTEGRITE**
- 9. DEPLOIEMENT & GUIDE D'INSTALLATION**
- 10. CONCLUSION**
- 11. BIBLIOGRAPHIE**

## REMERCIEMENTS

Nous tenons à exprimer notre gratitude à nos professeurs, **Pr. Yasyn Elyusufi** et **Pr. M'hamed AIT KBIR**.

Leur rigueur académique et leurs exigences techniques ont été un moteur constant pour nous. C'est grâce aux connaissances acquises lors de leurs cours que nous avons pu mener à bien ce projet ambitieux et approfondir nos compétences en développement backend et en architecture logicielle.

## 1. INTRODUCTION GENERALE & PROBLEMATIQUE

### 1.1. Contexte

La gestion d'un Data Center universitaire ou privé est une tâche d'une complexité rare, nécessitant une synchronisation parfaite entre le matériel physique, les couches de virtualisation et les besoins des utilisateurs. Une mauvaise gestion entraîne non seulement une sous-utilisation des ressources (coûts élevés), mais aussi des risques de pannes critiques et de conflits de planification.

À l'ère de la transformation numérique, la gestion des infrastructures informatiques est devenue un enjeu stratégique majeur. Les Data Centers, centres névralgiques de stockage et de calcul, nécessitent une organisation millimétrée pour garantir la continuité de service. Dans un cadre universitaire ou professionnel, l'allocation de ressources telles que les serveurs physiques, les machines virtuelles (VM) ou les baies de stockage est souvent source de conflits d'usage et de sous-optimisation matérielle.

### 1.2. Problématique

Comment centraliser, sécuriser et automatiser la gestion des ressources (inventaire + réservation + incidents + suivi), tout en offrant :

- Une vision temps réel (disponible/occupé/maintenance),
- Une traçabilité complète (qui a réservé quoi, quand, pourquoi),
- Un workflow de validation robuste,
- Un pilotage via Dashboard (KPIs / graphiques) ?

### 1.3. Objectif du projet

DataHub est une plateforme DCIM visant à remplacer les méthodes de gestion disparates par un système unifié de gouvernance des ressources.

Développée sous Laravel, cette application web full-stack dépasse le simple CRUD en intégrant un moteur de réservation intelligent, une gestion fine des droits (RBAC) et des tableaux de bord décisionnels. L'enjeu principal est de centraliser et d'automatiser l'administration de l'infrastructure pour garantir un suivi fluide, proactif et sécurisé.

## 2. ARCHITECTURE TECHNIQUE ET ORGANISATION DU CODE :

Contrairement aux structures génériques, DataHub suit une organisation modulaire stricte pour séparer les responsabilités. Voici l'architecture réelle de l'application telle qu'elle a été implémentée.

### 2.1. Arborescence détaillée du projet

Le Projet est structuré comme suite :

```
DataHub/
├── app/                                # CŒUR DE L'APPLICATION (Logique métier Laravel)
│   ├── Http/
│   │   ├── Controllers/               # Contrôleurs (orchestration des cas d'usage)
│   │   │   ├── Auth/                 # Espace Admin (RBAC, gestion globale)
│   │   │   │   ├── AuthenticatedSessionController.php
│   │   │   │   ├── NewPasswordController.php
│   │   │   │   ├── PasswordController.php
│   │   │   │   ├── PasswordResetLinkController.php
│   │   │   │   └── RegisteredUserController.php
│   │   │   ├── AdminController.php    # Pilote global : statistiques et
│   │   │   │                               gestion utilisateurs
│   │   │   ├── ChatbotController.php  # Cerveau de l'IA (NLP & réponses
│   │   │   │                               automatiques)
│   │   │   ├── Controller.php         # Classe mère dont héritent tous les
│   │   │   │                               contrôleurs
│   │   │   ├── DashboardController.php # KPIs généraux (Taux d'occupation,
│   │   │   │                               activité)
│   │   │   ├── EngineerController.php # Vue technique (Rack Map, KPIs infra)
│   │   │   ├── IncidentController.php # Ticketing (Signalement et résolution
│   │   │   │                               de pannes)
│   │   │   ├── NotificationController.php # Flux d'alertes en temps réel
│   │   │   ├── ProfileController.php   # Gestion du compte utilisateur (Bio,
│   │   │   │                               MDP)
│   │   │   ├── ReportController.php    # Générateur de rapports (Audit,
│   │   │   │                               Inventaire)
│   │   │   ├── ReservationController.php # Orchestrateur des emprunts (Workflows)
│   │   │   └── ResourceController.php  # Inventaire technique (CRUD serveurs,
│   │   │   │                               QR Codes)
│   │   └── Middleware/               # LES FILTRES (Sécurité et vérifications avant
│   │   │   │                               l'action)
│   │   │   ├── Authenticate.php       # Vérifie si l'utilisateur est connecté (auth)
│   │   │   ├── CheckAccountStatus.php # Bloque l'accès si le compte est désactivé
│   │   │   ├── CheckRole.php          # Filtre les accès selon le rôle
│   │   │   │                               (Admin/Responsable)
│   │   │   └── EncryptCookies.php     # Chiffre les cookies pour la confidentialité
```

```

├── PreventRequestsDuringMaintenance.php # Active le "Mode Maintenance"
│                                       du site
├── RedirectIfAuthenticated.php # Redirige vers le dashboard si déjà
│                               connecté
├── RoleMiddleware.php # Gestion fine des permissions RBAC
├── TrimStrings.php # Nettoie les espaces inutiles dans les
│                  formulaires
├── TrustHosts.php # Protection contre les attaques de type Host
│                 Header
├── TrustProxies.php # Configuration pour les serveurs derrière un
│                   Proxy
├── VerifyCsrfToken.php # Protection contre les failles CSRF (Sécurité
│                       Formulaire)
├── Requests/ # Validation serveur (Form Requests)
│   ├── Auth/
│   │   └── LoginRequest.php
│   └── ProfileUpdateRequest.php
├── Kernel.php
├── Models/ # Couche Données (Eloquent ORM : les objets DB)
│   ├── User.php # Profils + Rôles (Admin/Responsable/User)
│   ├── Resource.php # Spécifications serveurs (CPU/RAM/Position
│   │                 Rack)
│   ├── Reservation.php # Lien temporel entre un User et une Resource
│   ├── Log.php # Historique des actions critiques (Auditeur)
│   └── Incident.php # Traces des pannes par ressource
├── routes/ # Définition des routes (web, auth, admin...)
│   └── web.php # Routes principales + middlewares
├── database/ # Schéma & données initiales
│   ├── migrations/ # Création tables: users/ressources/...
│   └── seeders/ # Données de démo (admin, ressources)
├── resources/ # Interface (Vue)
│   ├── views/ # Blade (pages, dashboards, forms)
│   ├── js/ # Vanilla JS + Vite (interactions)
│   └── css/ # Design System (dark mode, animations)
├── public/ # Racine web (Seul dossier accessible via URL)
├── config/ # Paramètres (Services, Mails, Base de données)
├── storage/ # Persistance locale (PDF générés, logs PHP)
└── tests/ # Scénarios de validation automatisés

```

```

├─ bootstrap/                # Initialisation de l'application (Démarrage)
├─ artisan                  # Outil magique en ligne de commande
                             (Migrations, Seeders)
├─ composer.json            # Registre des bibliothèques PHP (Backend)
├─ package.json             # Registre des bibliothèques JS/CSS (Frontend)
└─ RAPPORT_TECHNIQUE.md     # Guide complet pour l'examen (Diagrammes,
                             Logique)

```

## 2.2. Analyse des fichiers sources clés

- **web.php** : Ce fichier est le point d'entrée sécurisé. Il implémente des groupes de routes protégés par les middlewares auth et admin, garantissant que personne ne peut accéder aux fonctions de gestion sans les droits appropriés.
- **AdminController.php** : Il contient les méthodes critiques de validation des utilisateurs (approveUser) et de résolution d'incidents, illustrant une gestion rigoureuse du workflow administratif.
- **Les Vues Blade** : L'utilisation de @extends('layouts.app') dans la quasi-totalité des vues prouve l'utilisation d'une architecture DRY (Don't Repeat Yourself), facilitant la maintenance visuelle.

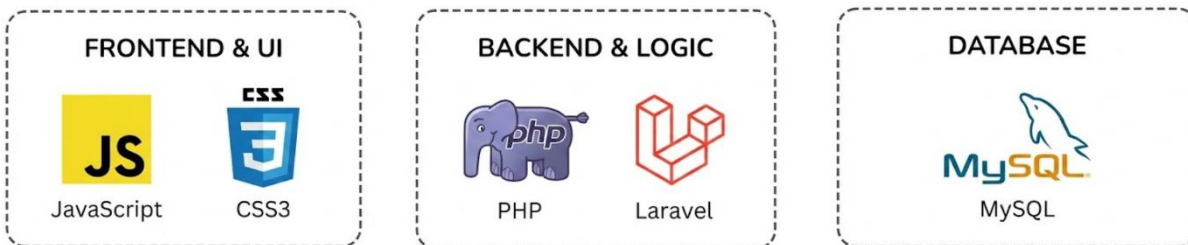
## 3. ÉTAT DE L'ART ET CHOIX DES TECHNOLOGIES :

### 3.1. La pile technologique : Laravel 11 & MySQL 8 (Stack)

Le choix de cette pile technologique ne relève pas d'une simple préférence, mais répond impérativement aux exigences de performance et de sécurité industrielle du projet. L'adoption du framework **Laravel** s'est imposée comme une évidence architecturale, justifiée par trois piliers fondamentaux :

- **La puissance de l'ORM Eloquent** : Cet outil ne sert pas uniquement à transformer les tables SQL en objets manipulables pour protéger l'application contre les injections SQL (abstraction sécurisée) ; il a également été déterminant pour sa capacité à gérer nativement des relations de données complexes (One-to-Many, Many-to-Many).
- **Le système de Migrations** : Il permet un versioning précis et historique de la structure de la base de données, une fonctionnalité indispensable pour garantir l'évolutivité du projet.
- **Le moteur de template Blade** : Il assure une séparation nette entre le code logique (Backend) et le rendu HTML (Front-end), tout en permettant une injection de données fluide et sécurisée.

En complément, le SGBD **MySQL 8** a été privilégié pour la couche de persistance. Ce choix assure la robustesse nécessaire au traitement des requêtes concurrentes, un enjeu critique pour la gestion fiable de la table des réservations.



### 3.2. Philosophie du "No-Framework" Frontend (Vanilla)

Il s'agit du choix le plus audacieux de ce projet : contrairement à l'utilisation systématique de frameworks comme Bootstrap ou Tailwind qui imposent souvent des styles génériques, j'ai opté pour une approche "Vanilla" (code pur) afin de garantir une maîtrise totale et une performance optimale. Cette stratégie repose sur deux axes :

- **CSS Pur & Identité Visuelle** : J'ai développé un système de grille et de thèmes néons basés sur les variables CSS. Ce choix démontre ma capacité à concevoir des mises en page complexes (Grid, Flexbox) à partir de zéro, assurant une identité graphique unique et une compatibilité multi-navigateurs parfaite.
- **JS Pur & Performance** : L'interactivité (modales, prévisualisations d'images) est gérée nativement, notamment via l'API FileReader. Cette méthode assure une rapidité d'exécution inégalée et une réduction drastique du poids des pages en éliminant les bibliothèques inutiles de plusieurs Mo.

#### Résumer :

Technologie	Catégorie	Justification
PHP 8.2+	Langage Serveur	Langage mature, performant pour le traitement de logique métier complexe.
Laravel 11	Framework Backend	Fournit une structure MVC rigoureuse et des outils de sécurité (Eloquent, Blade, Middlewares).
MySQL 8	Base de Données	Standard industriel pour les données relationnelles avec support des transactions.

<b>Vanilla CSS</b>	Design Interface	Maîtrise totale du rendu visuel sans dépendances tierces, garantissant légèreté et rapidité.
<b>Vanilla JS</b>	Dynamisme	Manipulation du DOM haute performance pour les interactions asynchrones (Modales, Filtres).

## 4. OBJECTIFS ET PERIMETRE DU PROJET :

### 4.1 Objectifs stratégiques

- **Inventaire intelligent** : gestion centralisée des ressources + localisation + statut.
- **Réservations orchestrées** : demandes, validation, prévention des conflits.
- **Maintenance & incidents** : signalement et suivi.
- **Dashboard** : indicateurs, occupation, alertes.
- **Expérience utilisateur** : interface fluide (mode sombre, feedback UI).
- **Rapports** : génération PDF et visualisation via graphiques.

### 4.2 Périmètre fonctionnel

- Gestion utilisateurs & rôles
- Gestion des ressources (serveur/VM/switch/storage...)
- Réservations (workflow + calendrier)
- Incidents (tickets)
- Statistiques + reporting

## 5. ANALYSE DES BESOINS ET SPÉCIFICATIONS :

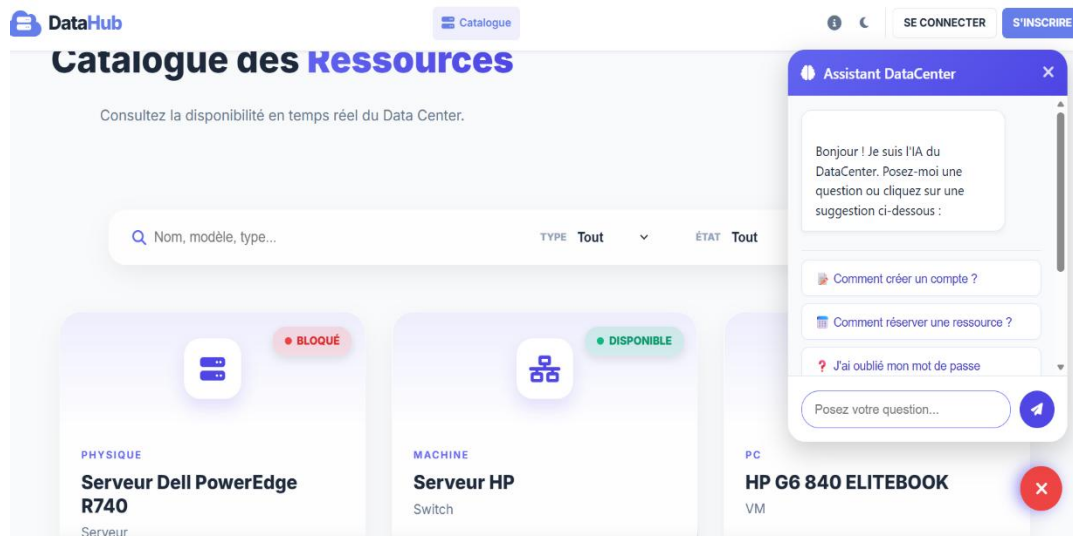
L'application DataHub repose sur une hiérarchie stricte de droits d'accès, répondant aux spécifications fournies.

### 5.1. Profils (RBAC)

#### A) Invité / Candidat

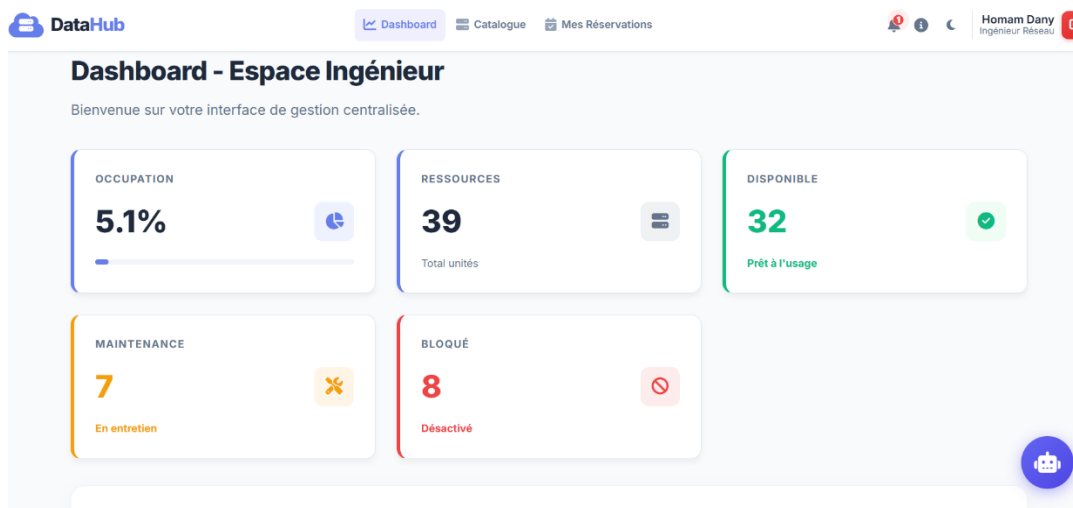
- Consulter infos générales
- Demander un compte / accès





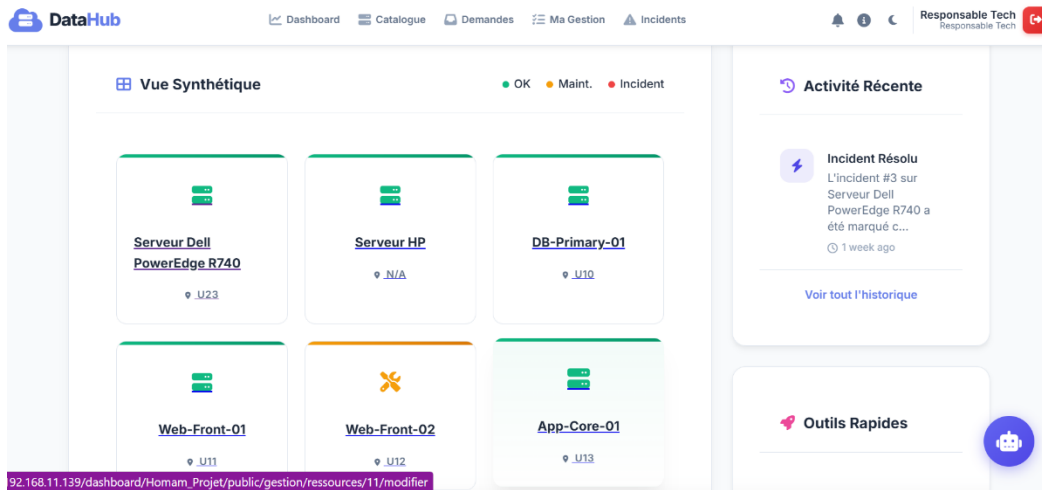
## B) Utilisateur

- Rechercher une ressource
- Voir disponibilité
- Demander une réservation (période + justification)
- Suivre le statut
- Déclarer un incident



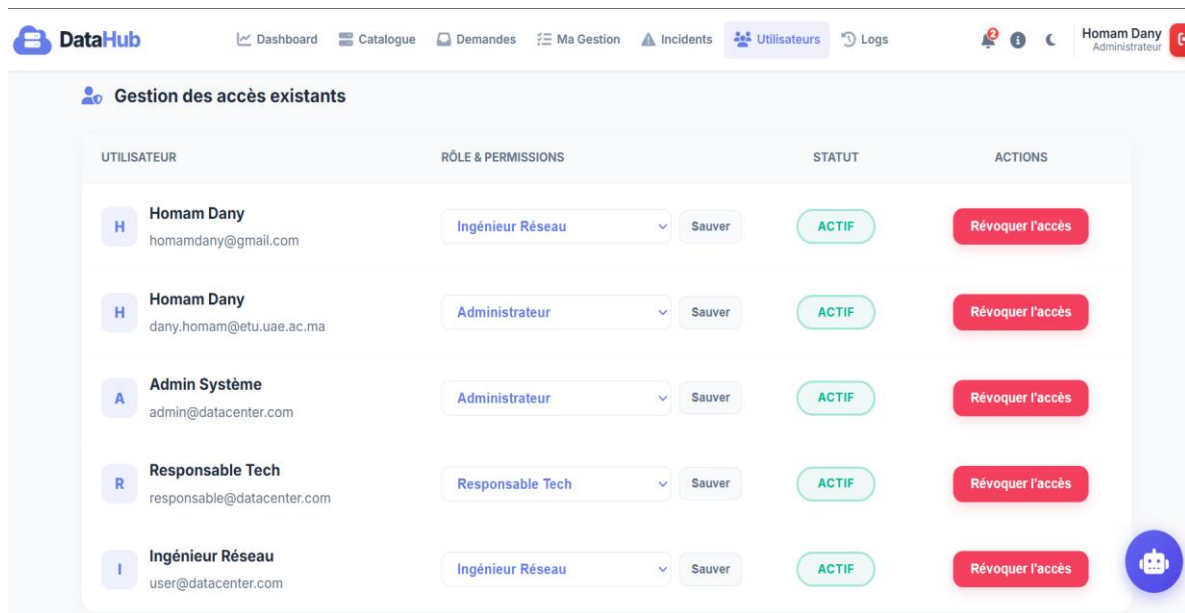
## C) Responsable (manager)

- Gérer un périmètre de ressources
- Valider / rejeter des réservations
- Suivre incidents, prioriser interventions d'administrateur
- Supervision globale
- Gestion utilisateurs & rôles
- Dashboard global + reporting



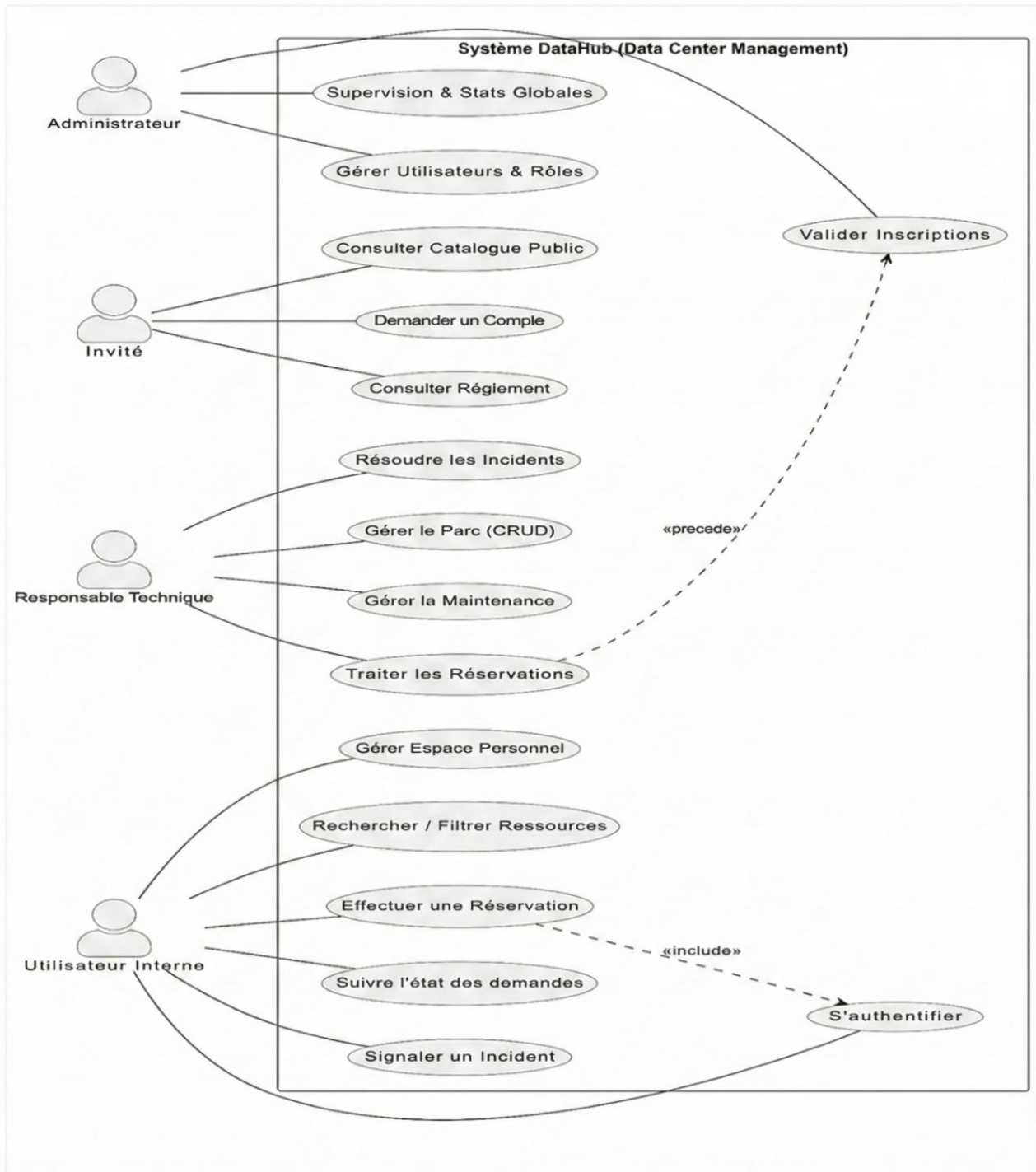
## D) Administrateur

- Supervision globale
- Gestion utilisateurs & rôles
- Dashboard global + reporting

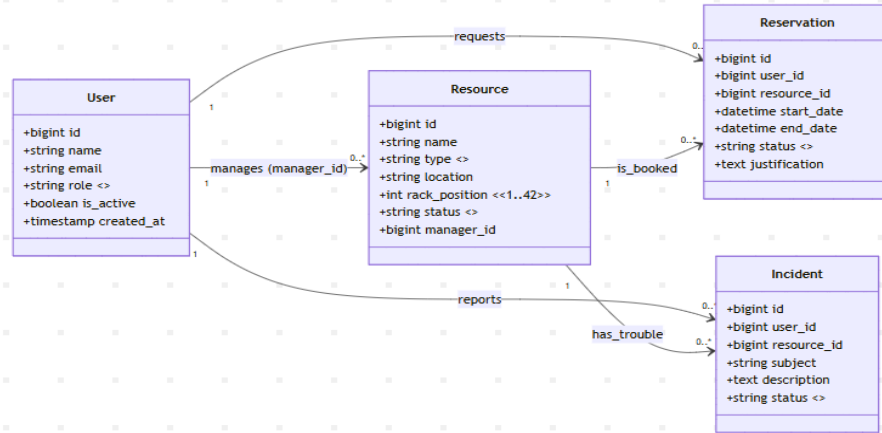


## 6. CONCEPTION & MODELISATION (UML + DONNEES) :

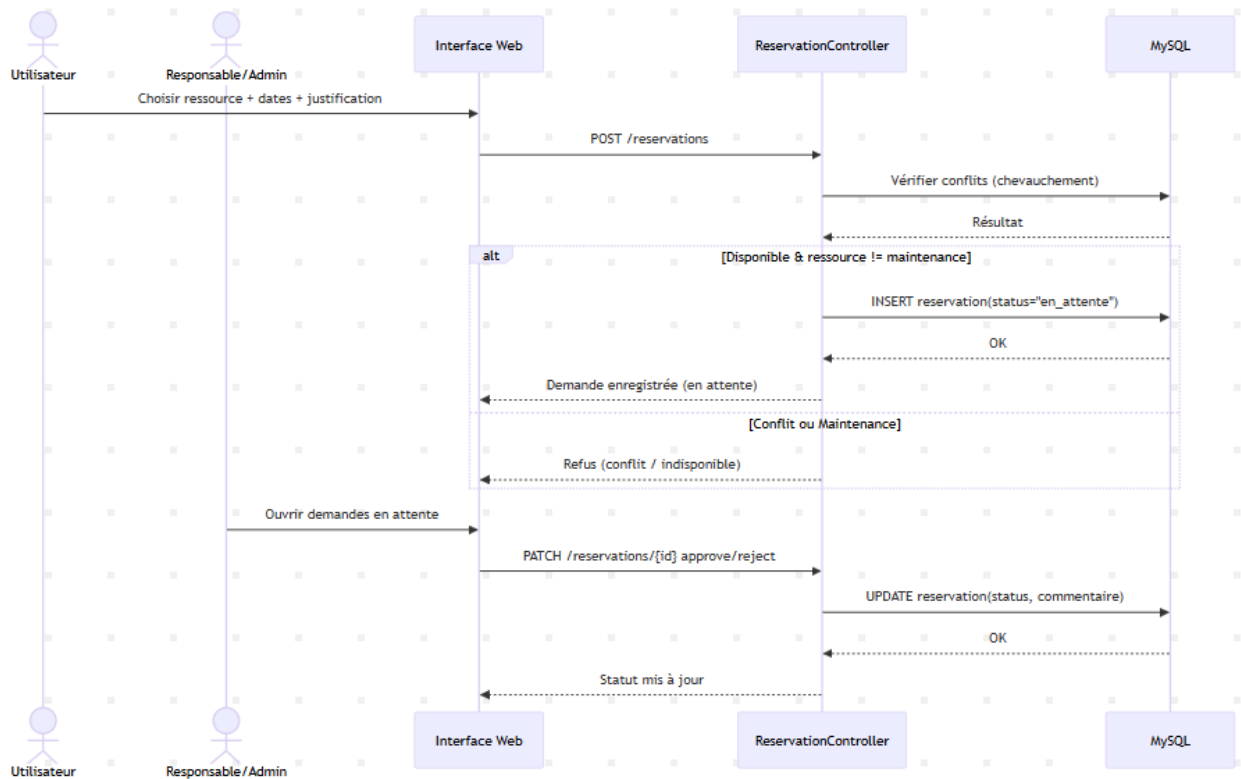
### Diagramme de cas d'utilisation (UML) :



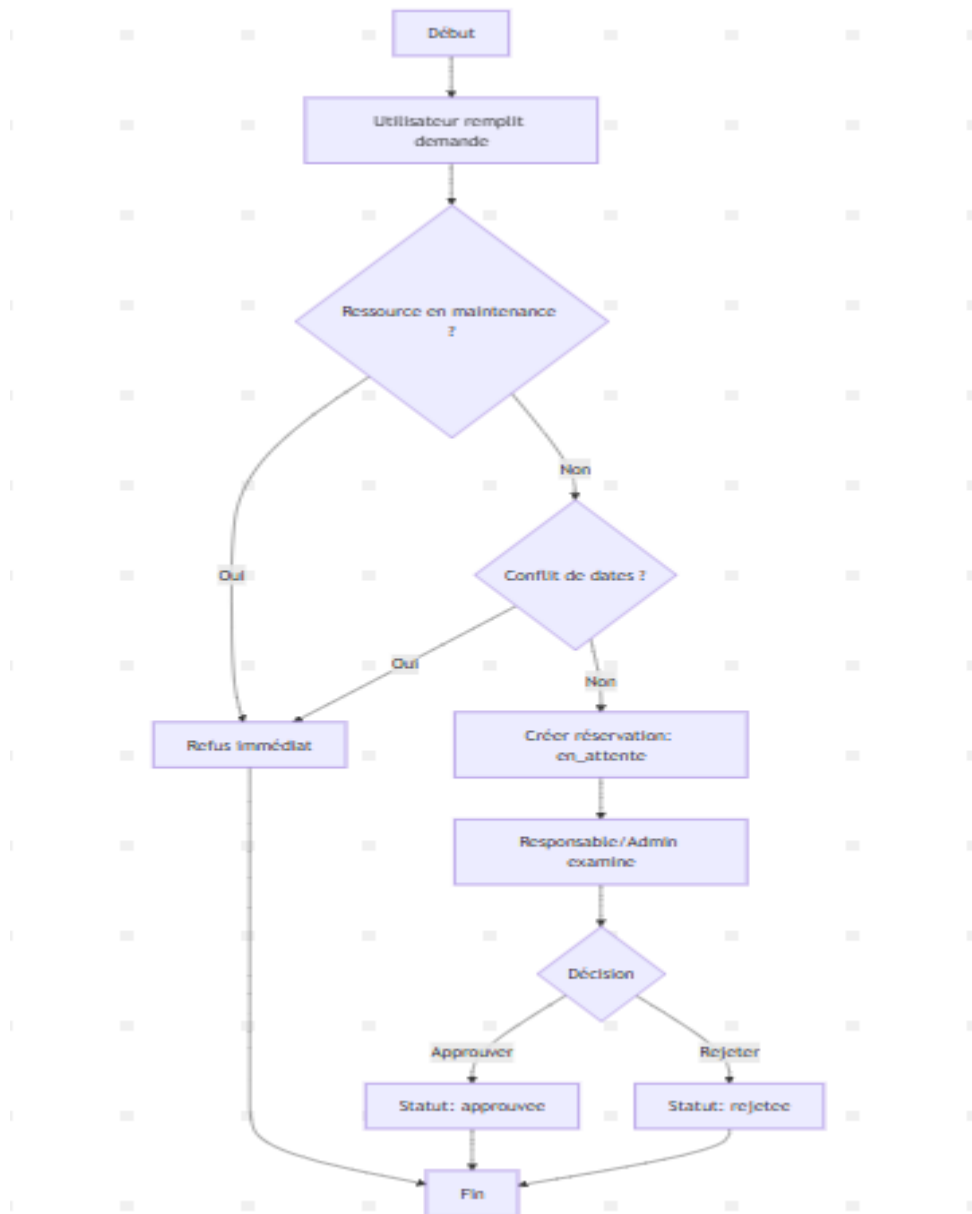
## Diagramme de classes UML :



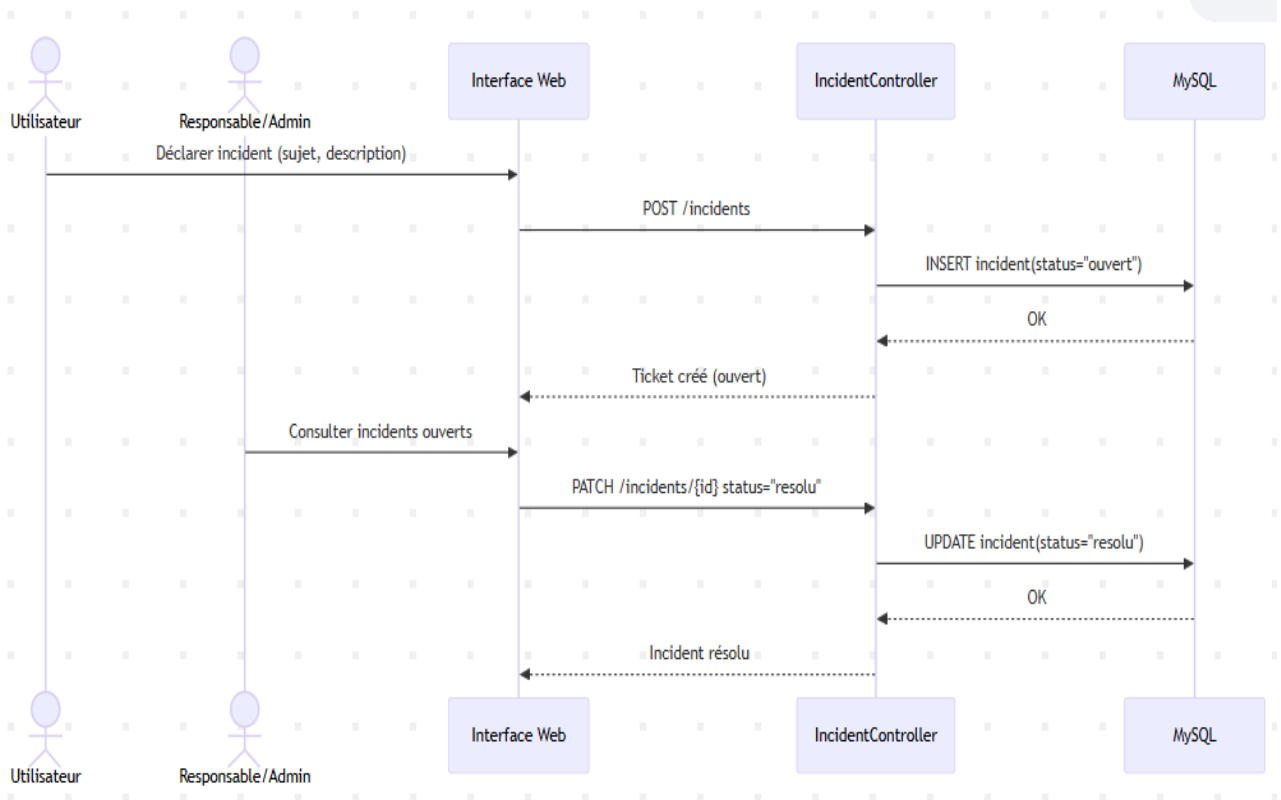
## Séquence (Réservation + détection conflit + validation) :



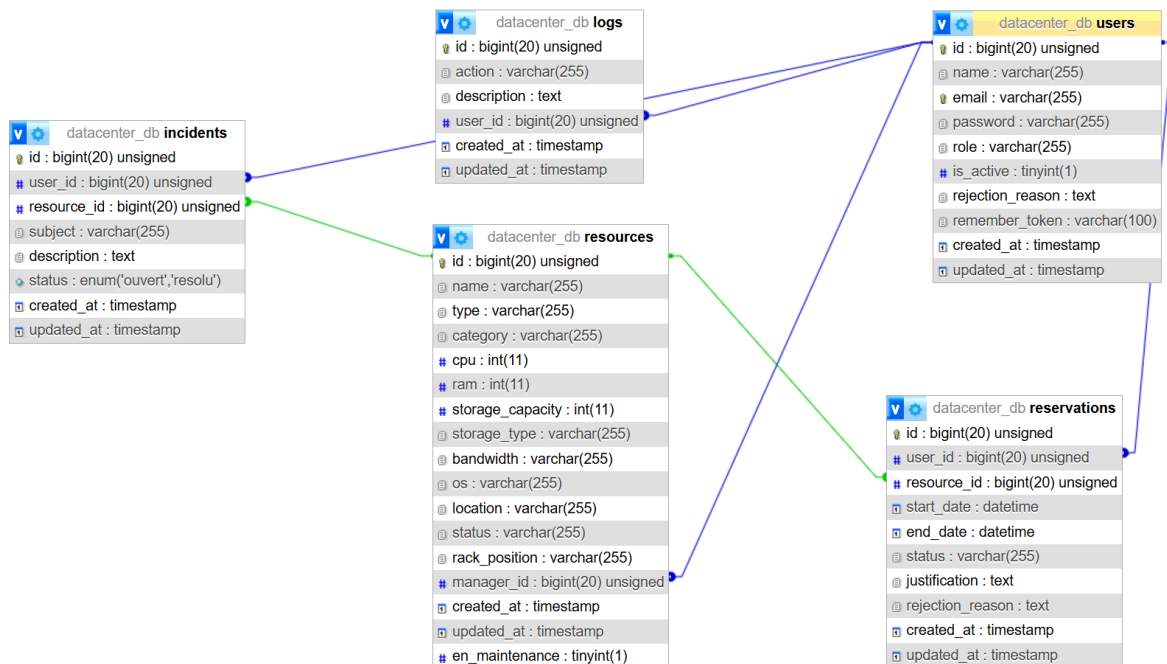
## Diagramme d'activité (Cycle de vie réservation) :



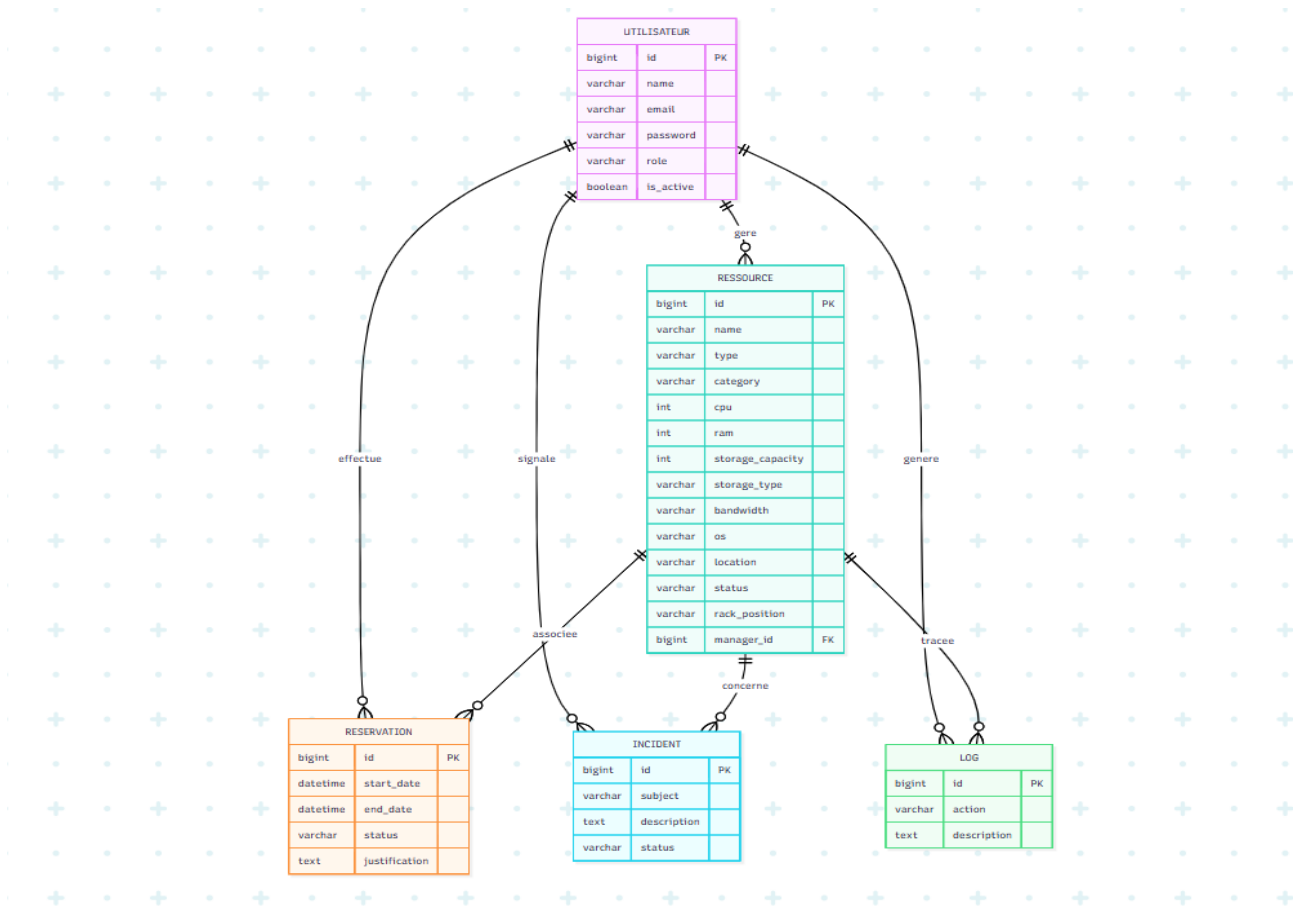
## Séquence (Déclaration incident + traitement) :



## MLD :



## MCD avec cardinalités :



## 7. REALISATION DES MODULES :

### 7.1. Inventaire / Catalogue :

- Recherche multicritères (type/statut/localisation)

**DataHub** | Dashboard | Catalogue | Demandes | Ma Gestion | Incidents | Utilisateurs | Logs | Homam Dany Administrateur

Recherche: Nom, modèle, type... | TYPE: Tout | ÉTAT: Tout | 47 ressources

**PHYSIQUE** **Serveur Dell PowerEdge R740** **BLOQUÉ**

45 CORES | 140 GO RAM

**MACHINE** **Serveur HP** **DISPONIBLE**

10 CORES | 5 GO RAM

**PC** **HP G6 840 ELITEBOOK** **DISPONIBLE**

5 CORES | 4 GO RAM

- Fiches ressources (caractéristiques + statut + gestion)



## 7.2. Rack Map 42U

- Visualisation physique : position rack, unités occupées/libres, alertes maintenance.

**DataHub** Dashboard Catalogue Demandes Ma Gestion Incidents Utilisateurs Logs

Homam Dany Administrateur

42 ● Load-Balancer-01  
41 ● Firewall-Main-01  
40 ● Switch-Core-01  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30 ● AI-GPU-Node-01  
29  
28  
27  
26  
25 ● Monitor-Sys-01  
24  
23 ● Serveur Dell PowerEdge R740  
22 ● Archive-Store-01  
21 ● NAS-Storage-02  
20 ● NAS-Storage-01  
19 ● Serveur HP 2026  
18

### Légende

● Actif / Disponible ● Maintenance ■ Emplacement Vide

**Note :** Pour placer un serveur dans la baie, éditez la ressource et assignez-lui une position (ex: "U10"). Les serveurs multi-U ne sont pas encore visualisés en pleine hauteur.

### Serveurs non-rackés


NOM	TYPE	ACTION
Liste disponible dans la gestion des ressources...		





### 7.3. Réservations (workflow) :

- Création demande :
- Vérification dispo + conflits
- Statut : en attente → approuvée / rejetée
- Historique & traçabilité

**CHOISIR L'ÉQUIPEMENT**

Serveur HP (Switch) 

**DISPONIBILITÉ & SÉLECTION**

 **Février 2026** 

Di	Lu	Ma	Me	Je	Ve	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20 •	21 •
22	23	24	25	26 •	27 •	28 •

**DATE DE DÉBUT**  
---

**DATE DE FIN**  
---

☐ Indisponible


☐ Libre

☒ Réservé

☒ Sélection

**JUSTIFICATION DU BESOIN**

Expliquez pourquoi vous avez besoin de cette ressource...

 Confirmer la réservation

Annuler

### 7.4. Incidents (ticketing) :

- Déclaration (sujet + description)
- Statuts : ouvert / résolu
- Suivi par responsable / admin

## ⚠ Signaler un problème



### SUJET DE L'INCIDENT

Ex: Panne réseau, Surchauffe...

### DESCRIPTION DÉTAILLÉE

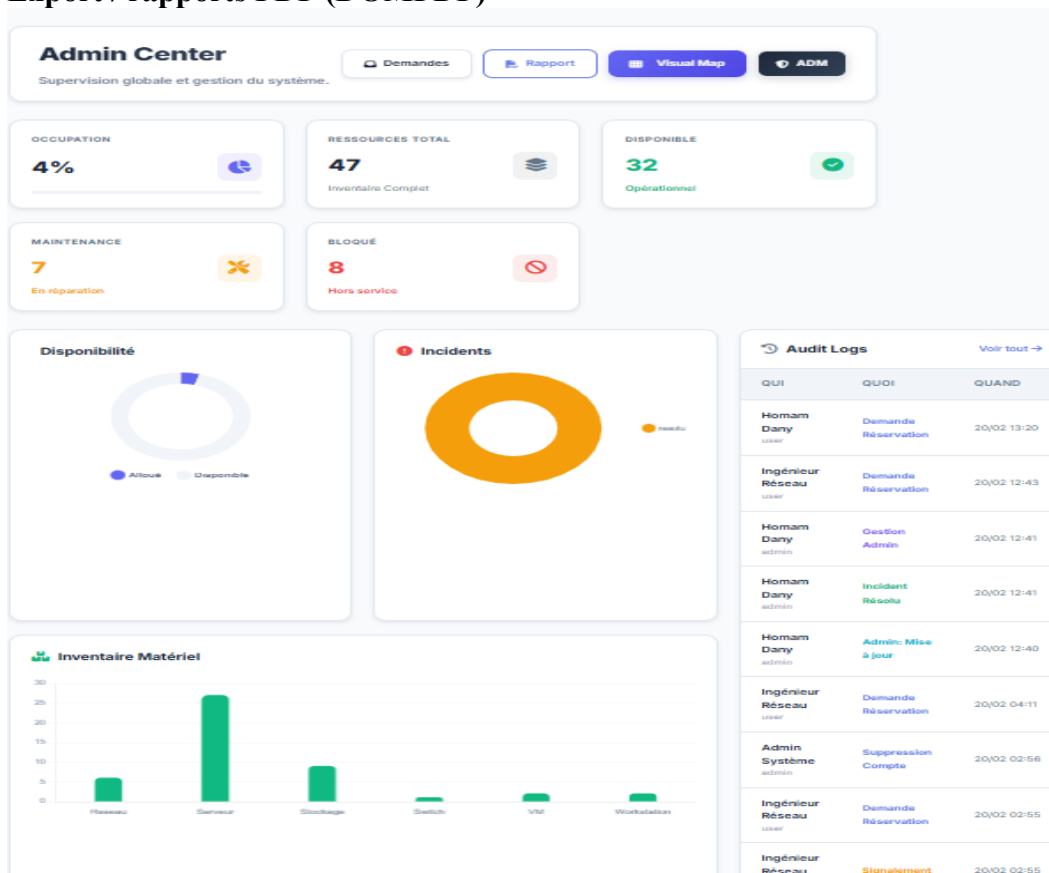
Décrivez le problème rencontré sur cette ressource...

Annuler

➤ Envoyer le signalement

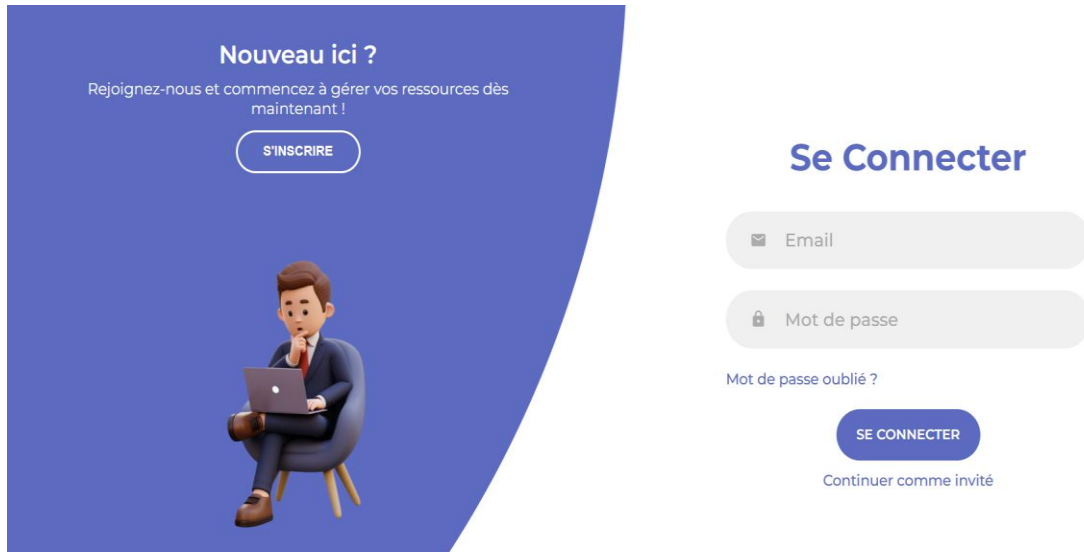
## 7.5. Dashboard & KPI:

- Taux d'occupation, répartition statuts
- Graphiques (Chart.js)
- Export / rapports PDF (DOMPDF)



## 8. SECURITE, CONTROLE D'ACCES & INTEGRITE:

- **Authentification sécurisée**
- **Middlewares RBAC**
- **Validation serveur (Request validation)**
- **Protection CSRF**
- **ORM Eloquent (anti injection SQL)**
- **Hash password (bcrypt)**



## 9. DEPLOIEMENT & GUIDE D'INSTALLATION :

Pour permettre au jury de tester l'application, un fichier README.md détaillé a été rédigé à la racine du projet GitHub. Il contient les étapes suivantes :

- **Clonage du dépôt** : git clone [https://github.com/Homam-Dany/Application\\_Web\\_DataCenter](https://github.com/Homam-Dany/Application_Web_DataCenter)
- **Installation des dépendances** : composer install
- **Configuration de l'environnement** : Création du fichier .env et configuration de la base de données MySQL.
- **Migrations et Seeders** : php artisan migrate --seed (pour générer les tables et les comptes de test).
- **Lancement du serveur** : php artisan serve

Note : Ce dépôt contient l'intégralité du code source, des ressources CSS/JS et de la documentation technique.

## 10.CONCLUSION :

Au terme de ce travail, nous pouvons affirmer que le projet **DataHub** marque une étape clé dans notre formation commune. Il concrétise notre capacité à collaborer pour concevoir une architecture logicielle robuste, modéliser des données complexes et développer une interface utilisateur aux standards professionnels.

Cette expérience a été particulièrement formatrice. En partant d'une problématique réelle de gestion d'infrastructure, nous avons réussi le pari de construire une solution "Full Stack" complète, alliant la puissance de Laravel à la performance d'un frontend développé sur mesure.

Aujourd'hui, DataHub n'est plus un simple concept académique, mais une plateforme fonctionnelle, prête à être déployée pour optimiser la gestion d'un véritable Data Center universitaire. Ce projet nous a prouvé qu'en alliant rigueur technique et attention portée à l'expérience utilisateur, il est possible de transformer des processus administratifs complexes en une solution fluide, sécurisée et transparente.

## 11.BIBLIOGRAPHIE ET WEBOGRAPHIE:

- **Documentation Laravel 10 Framework Guide** : Base de l'architecture MVC, du middleware de sécurité et de l'ORM Eloquent. <https://laravel.com/docs/10.x>
- **MDN Web Docs (Mozilla Developer Network)** : Standard de référence pour l'implémentation de la logique JavaScript ES6, des APIs Fetch et des manipulations directes du DOM. <https://developer.mozilla.org/>
- **Chart.js Documentation** : Bibliothèque open-source utilisée pour la data-visualisation interactive du tableau de bord d'administration. <https://www.chartjs.org/>
- **Vite.js Documentation** : Outil de build nouvelle génération pour l'optimisation des assets web. <https://vitejs.dev/>
- **DomPDF** : Convertisseur HTML vers PDF en PHP, utilisé pour la génération des rapports mensuels et KPIs.
- **FontAwesome & Lucide Icons** : Bibliothèques d'icônes vectorielles intégrées pour l'expérience utilisateur.
- **Normes IDAI (Ingénierie de Développement d'Applications Informatiques)** : Standards académiques de la FST Tanger régissant l'architecture logicielle, la modélisation UML et la rédaction de rapports techniques.