

Rapport du projet de fin d'études

*Mémoire de fin d'études pour l'obtention du diplôme de licence en
Développement informatique.*

Développement d'une application de gestion des incidents

Soutenu le : 09/07/2025

Réalisé par : othmane amahal

Encadré par : M.marah

Année universitaire: 2024 – 2025

Dédicace

Je dédie ce travail à :

Mes parents, pour leur amour inestimable, leurs sacrifices et leur soutien indéfectible tout au long de mon parcours académique.

Monsieur **Marah**, mon encadrant, pour son accompagnement, sa disponibilité, ses conseils pertinents et son soutien constant durant la réalisation de ce projet.

L'ensemble du corps professoral de l'école **SUPMIR**, pour la qualité de l'enseignement dispensé et les compétences transmises tout au long de ma formation.

Mes camarades de promotion, pour l'esprit de collaboration et les échanges enrichissants que nous avons partagés.

Toutes les personnes ayant, de près ou de loin, contribué à l'aboutissement de ce projet.

Je leur exprime ma profonde gratitude et mes remerciements les plus sincères.

Remerciements

Ce travail marque l'aboutissement de trois années d'études au sein de l'école **SUPMIR**, une étape importante de mon parcours académique. À cette occasion, je tiens à exprimer ma reconnaissance à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce projet de fin d'études.

Je remercie tout particulièrement **M. Marah**, mon encadrant pédagogique, pour son accompagnement tout au long de ce projet. Sa disponibilité, ses conseils avisés, sa patience et son encadrement rigoureux ont été essentiels à l'aboutissement de ce travail.

Je souhaite également exprimer ma gratitude à **l'ensemble du corps professoral de SUPMIR**, pour la qualité de l'enseignement reçu, leur implication, et le partage de leurs connaissances tout au long de ma formation.

Un merci tout spécial à **mes parents**, pour leur amour inconditionnel, leur soutien moral et leurs encouragements constants, qui m'ont permis d'avancer avec confiance et motivation dans mon parcours.

Enfin, je n'oublie pas **mes camarades et amis**, avec qui j'ai partagé cette belle aventure académique. Leur présence, leur entraide et nos échanges constructifs ont grandement enrichi cette expérience.

SOMMAIRE

DEDICACE.....	1
REMERCIEMENTS.....	2
SOMMAIRE	3
RESUME.....	4
INTRODUCTION.....	5
CHAPITRE 1 : PRESENTATION DU PROJET.....	6
1.1- Définition de la gestion des incidents	7
1.2- Cycle de vie d'un incident.....	8
1.3- Outils et méthodes existants.....	9
1.4- Importance d'un bon système de gestion	10
CHAPITRE 2 : cahier des charges	11
2.1- Problématique	12
2.2- Objectifs du projet.....	13
2.3- Spécification fonctionnelles et techniques	14
CHAPITRE 3 : conception du système	15
3.1- Architecture générale du système	16
3.2- Modèle de donnée (Mongo DB)	17
3.3- Diagramme UML (cas d'utilisation, séquence, classes)	18
CHAPITRE 4 : Réalisation et Développement	19
4.1- Choix technologiques (MERN Stack).....	20
4.2- Interfaces principales(React).....	21
4.3- Structure de l'application	22
4.4- API REST (Node.js & Express.js)	23
4.4- Sécurité et authentification.....	24
CHAPITRE 5 : Résultats et tests	25
5.1- Scénarios de test.....	26
5.2- Résultats obtenus.....	27
5.3- Problème rencontrés et solutions	28
Conclusion et perspectives	29
Bibliographie.....	30

Annexes (captures d’écran, scripts...)..... 31

RESUME

Ce projet consiste en la conception et le développement d'un système de gestion des incidents destiné à faciliter le suivi, la classification, et la résolution rapide des incidents au sein d'une organisation.

Le système permet l'enregistrement des incidents, l'affectation aux techniciens, ainsi que la génération de rapports statistiques.

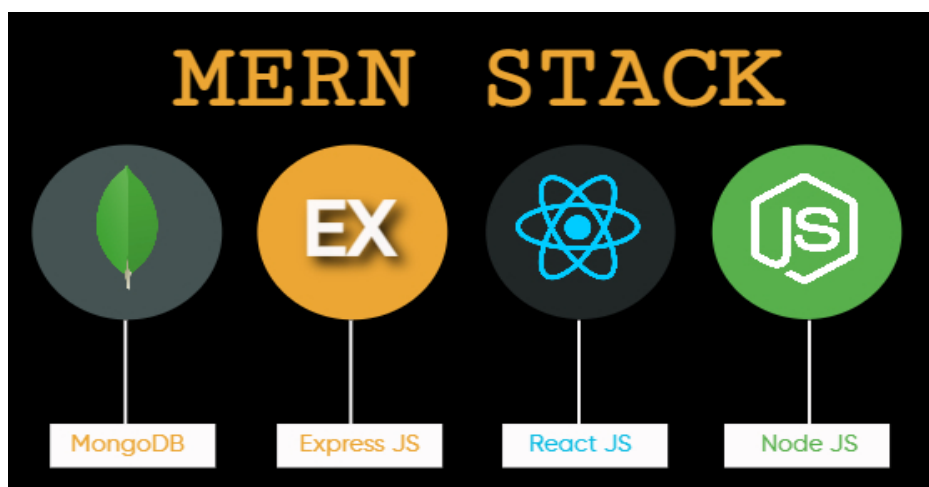
Pour réaliser ce projet, nous avons utilisé les technologies: (**Mongo DB**, **React**, **Node.js**, **Express.js**)

Les résultats montrent que ce système améliore considérablement la réactivité et la gestion des incidents, contribuant ainsi à une meilleure qualité de service.

Mon travail s'inscrit dans le cadre du développement d'une application informatique des incidents dédiée à la gestion des techniques.

Afin de mener à bien ce projet, j'ai suivi une démarche méthodique structurée en plusieurs étapes clés :

- **Rédaction du cahier des charges**
- **Définition des modules et des spécifications fonctionnelles**
- **Conception des interfaces utilisateur**
- **Développement des interfaces**
- **Connexion avec la base de données**
- **Phase de tests techniques**



Introduction

La gestion des incidents est un processus essentiel dans le domaine de la gestion des services informatiques (ITSM-**Information Technologie Service Management**). Elle vise à restaurer le fonctionnement normal d'un service aussi rapidement que possible après un dysfonctionnement, tout en minimisant l'impact négatif sur les opérations métiers.

Un incident est défini comme un événement qui n'est pas prévu et qui cause ou pourrait causer une interruption ou une réduction de la qualité d'un service informatique.

La gestion efficace des incidents permet d'assurer une meilleure disponibilité des services, de réduire les temps d'arrêt, et d'améliorer la satisfaction des utilisateurs finaux.

CHAPITRE 1 : PRESENTATION DU PROJET

Cycle de vie d'un incident

Le traitement d'un incident suit généralement un cycle structuré, composé des étapes suivantes :

1. Détection et enregistrement :

L'incident est détecté par l'utilisateur ou le système, puis enregistré dans une base de données centralisée à l'aide d'un formulaire ou d'un outil de ticketing.

2. Classification et priorisation :

L'incident est catégorisé selon son type (logiciel, matériel, réseau, etc.) et priorisé en fonction de son impact et de son urgence.

3. Affectation :

L'incident est assigné à un technicien ou à une équipe en fonction de la compétence requise et de la disponibilité.

4. Diagnostic initial :

Le technicien effectue une première analyse pour identifier la cause probable de l'incident.

5. Résolution et restauration :

Une solution est appliquée pour rétablir le service. Cette solution peut être temporaire (contournement) ou définitive.

6. Clôture de l'incident :

Une fois l'incident résolu, il est clôturé dans le système, accompagné d'un rapport ou d'un retour utilisateur.

7. Analyse post-incident (*facultative*) :

Pour les incidents majeurs, une revue est effectuée afin d'identifier les causes racines et d'éviter la récurrence.

Outils et méthodes existants

Il existe plusieurs outils de gestion des incidents qui permettent d'automatiser et de centraliser ce processus.

Parmi les plus utilisés:

- **ServiceNow** : plateforme cloud très complète pour la gestion des services IT.
- **Jira Service Management** : outil de gestion des tickets avec intégration Agile.
- **GLPI** : solution open-source très répandue dans les environnements francophones.
- **Freshdesk** : outil simple pour les petites et moyennes entreprises.

Ces plateformes offrent des fonctionnalités comme :

- L'ouverture de tickets automatisée.
- Le suivi des SLA (Service Level Agreement).
- La gestion des files d'attente.
- Les notifications en temps réel.
- Les tableaux de bord pour le suivi des performances.

Importance d'un bon système de gestion des incidents

Mettre en place un système performant de gestion des incidents présente de nombreux avantages :

- **Réduction des temps d'arrêt** : grâce à une détection rapide et à une résolution structurée.
- **Amélioration de la satisfaction des utilisateurs** : via une communication claire et transparente.
- **Centralisation des données** : permettant une meilleure traçabilité.
- **Analyse des incidents récurrents** : facilitant l'amélioration continue et la prévention.
- **Suivi de performance des équipes de support** : grâce aux statistiques et indicateurs clés (KPI).

CHAPITRE 2 : cahier des charges

Problématique

Dans la majorité des organisations, la gestion des incidents se fait encore de manière manuelle ou avec des outils non adaptés. Cette approche peut entraîner:

- **Une perte de temps dans le traitement des incidents ;**
- **Une mauvaise répartition des tâches entre les techniciens ;**
- **Un manque de traçabilité et de suivi ;**
- **Une difficulté à mesurer la performance du service technique.**

Ces limites impactent directement la qualité des services fournis aux utilisateurs et freinent la réactivité des équipes informatiques. D'où le besoin d'un système automatisé, centralisé et performant permettant de gérer efficacement tous les incidents signalés au sein d'une organisation.

Objectifs du projet

Objectif general:

Développer une application web de gestion des incidents, simple d'utilisation, performante, et adaptée aux besoins des utilisateurs et des techniciens.

Objectifs spécifiques:

- Permettre l'enregistrement des incidents via une interface conviviale.
- Classifier et prioriser automatiquement les incidents.
- Assigner les incidents à des techniciens selon leur disponibilité.
- Suivre l'état d'avancement des incidents (nouveau, en cours, résolu, fermé).
- Générer des rapports statistiques pour analyser les performances.
- Gérer les utilisateurs selon des rôles différents (admin, technicien, utilisateur).

Specifications fonctionnelles

Le système devra permettre :

- La création de comptes pour différents types d'utilisateurs.
- L'authentification sécurisée avec gestion de sessions.
- La déclaration d'un incident avec formulaire (titre, description, priorité, type).
- L'affichage des incidents dans une liste avec filtres.
- Le changement d'état d'un incident (workflow).
- L'affichage des détails d'un incident (suivi).
- L'export ou affichage de statistiques (graphes, tableaux).

Specifications techniques

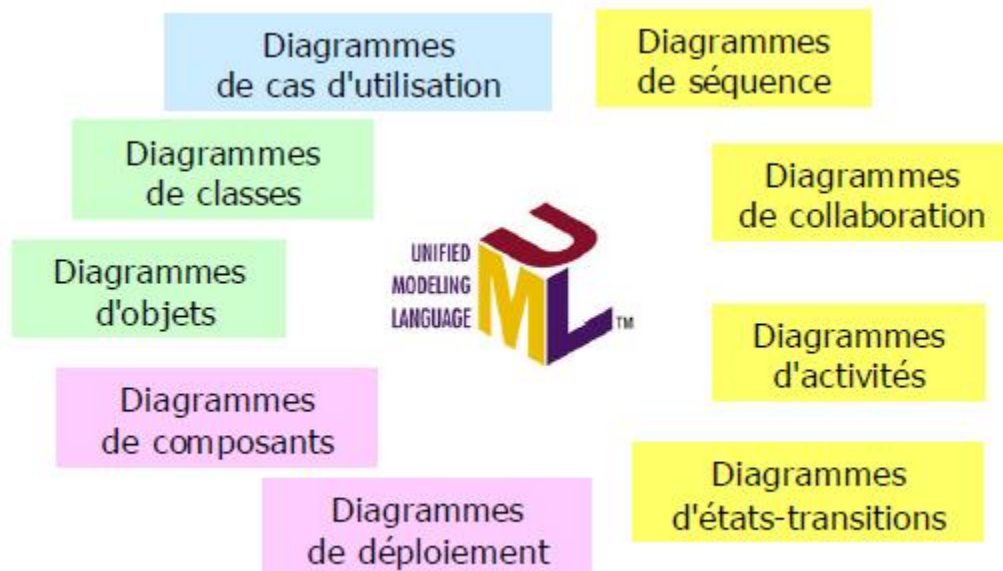
Le projet sera réalisé en utilisant la **stack MERN** :

- **Mongo DB** : Base de données NoSQL pour stocker les utilisateurs, incidents, rôles, etc.
- **Express.js** : Framework minimaliste pour le serveur Node.js.
- **React.js** : Bibliothèque JavaScript pour créer une interface utilisateur dynamique et responsive.
- **Node.js** : Environnement d'exécution pour le back end.

Autres outils et bibliothèques complémentaires :

- **JWT (JSON Web Token)**: pour l'authentification.
- **Axios** : pour les requêtes HTTP côté frontend.
- **Mongoose** : pour la modélisation des données MongoDB.
- **Tailwind CSS** : pour le design de l'interface.
- **Chart.js** : pour les graphiques statistiques.

Chapitre 3: Conception du system



Architecture générale du system

L'application repose sur une architecture **client-serveur** structurée selon le modèle **MERN** :

➤ Frontend (React.js)

- Gère l'interface utilisateur.
- Communique avec le backend via des appels API (REST).
- Affiche dynamiquement les tickets, formulaires, états, etc.

➤ Backend (Node.js + Express.js)

- Gère la logique métier, l'authentification, les rôles, et les routes API.
- Communique avec la base de données MongoDB.

➤ Base de données (MongoDB)

- Stocke les utilisateurs, incidents, rôles, historiques, etc.
- Structure NoSQL permettant une grande flexibilité.

Modèle de donnée (Mongo DB)

◆ **Utilisateur**

```
{  
  "_id": ObjectId,  
  "nom": "Ahmed",  
  "email": "ahmed@example.com",  
  "motDePasse": "*****",  
  "role": "technicien" // ou "admin", "utilisateur"  
},
```

◆ **Incident**

```
{  
  "_id": ObjectId,  
  "titre": "Erreur serveur",  
  "description": "Le serveur ne répond plus",  
  "priorite": "haute",  
  "etat": "en cours", // ou "nouveau", "résolu", "fermé"  
  "creePar": ObjectId, // Référence à un utilisateur  
  "attribueA": ObjectId,  
  "dateCreation": ISODate  
}
```

Diagramme des cas d'utilisation (Use Case)

N°	Cas d'utilisation	Acteur principal	Scénario principal
1	Créer un incident	Client	<ol style="list-style-type: none">1. Le client se connecte à l'application.2. Il accède à la section "Créer un incident".3. Il remplit le formulaire (titre, description).4. Il soumet l'incident.5. Le système enregistre l'incident avec le statut "en attente".
2	Modifier le statut d'un incident	Client / Administration	<ol style="list-style-type: none">1. L'utilisateur se connecte.2. Il accède à la liste des incidents.3. Il sélectionne un incident.4. Il choisit un nouveau statut (en cours, résolu, clôturé...).5. Le système met à jour le statut de l'incident.
3	Traiter un incident	Administration	<ol style="list-style-type: none">1. L'admin se connecte.2. Il consulte les incidents signalés.3. Il sélectionne un incident.4. Il ajoute un commentaire ou une réponse.5. Il change le statut à "en cours" ou "clôturé".

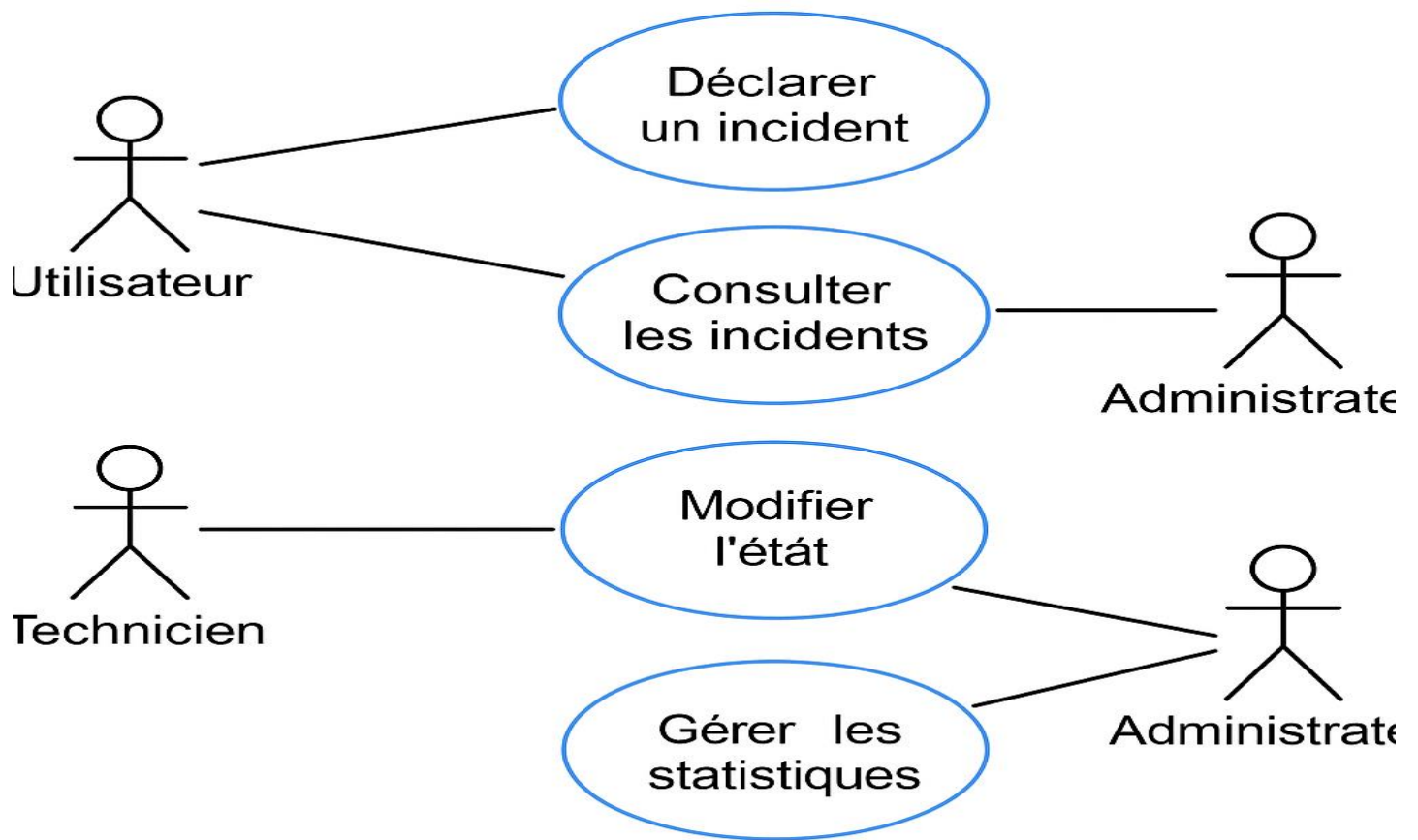
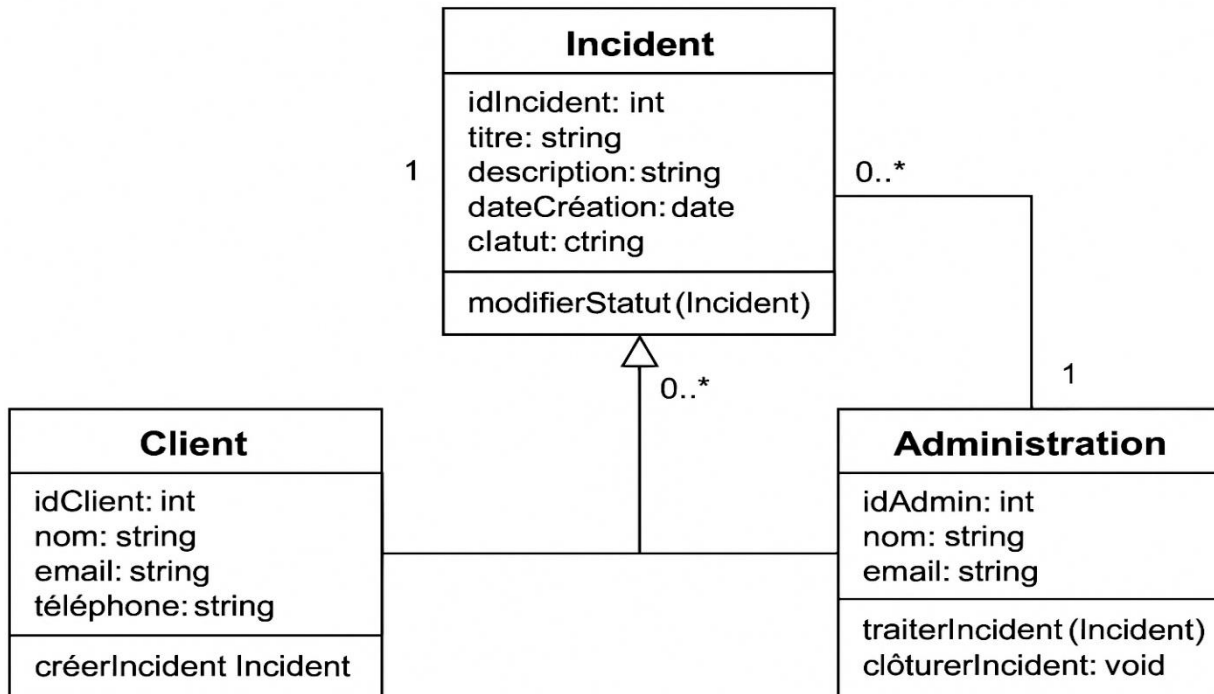


Diagramme de classes



◆ 1. Classe Client

La classe Client représente un utilisateur qui peut signaler un problème ou incident via l'application. Elle contient les informations personnelles du client ainsi qu'une méthode lui permettant de créer un incident.

- **Attributs :**
 - `idClient` : identifiant unique du client.
 - `nom` : nom du client.
 - `email` : adresse e-mail.
 - `téléphone` : numéro de téléphone.
- **Méthode :**
 - `créerIncident()` : permet au client de créer un nouvel incident.
- **Relation :**
 - Un client peut être associé à plusieurs incidents ($1 \rightarrow 0..*$).

❖ 2. Classe Incident

La classe Incident représente un problème ou une anomalie signalée par un client. Elle est au cœur de l'application.

- **Attributs :**
 - idIncident : identifiant unique de l'incident.
 - titre : titre ou objet de l'incident.
 - description : description détaillée du problème.
 - dateCréation : date de signalement de l'incident.
 - statut : état actuel de l'incident (en cours, traité, clôturé...).
 - **Méthode :**
 - modifierStatut() : permet de mettre à jour le statut de l'incident.
 - **Relations :**
 - Chaque incident est lié à un seul client ($0..* \rightarrow 1$ Client).
 - Il peut être traité ou clôturé par l'administration ($0..* \rightarrow 1$ Administration).
-

❖ 3. Classe Administration

La classe Administration représente l'équipe ou les utilisateurs responsables du traitement des incidents signalés.

- **Attributs :**
 - idAdmin : identifiant de l'administrateur.
 - nom : nom de l'administrateur.
 - email : adresse e-mail professionnelle.
- **Méthodes :**
 - traiterIncident(Incident) : permet de commencer le traitement d'un incident.
 - clôturerIncident() : permet de marquer un incident comme résolu ou clôturé.
- **Relation :**
 - Une administration peut traiter plusieurs incidents ($1 \rightarrow 0..*$ Incident).

Chapitre 4: Réalisation et development

Choix technologiques (Stack MERN)

Le choix de la stack MERN s'est imposé pour les raisons suivantes :

- **Mongo DB** : Base de données No SQL, souple et évolutive, adaptée à la gestion de documents complexes comme les incidents et les utilisateurs.
- **Express.js** : Framework back end léger et rapide, basé sur Node.js, utilisé pour construire des API REST.
- **React.js** : Bibliothèque puissante pour créer une interface utilisateur dynamique, responsive, et modulaire.
- **Node.js** : Plateforme d'exécution côté serveur, idéale pour les applications en temps réel avec un grand volume de requêtes.

Outils complémentaires:

- **JWT** pour la gestion des sessions et l'authentification sécurisée.
- **Axios** pour la communication HTTP entre frontend et backend.
- **Mongoose** pour la modélisation des données MongoDB.
- **Tailwind CSS** pour le design moderne et réactif.
- **Chart.js** pour la visualisation des statistiques.

Structure de l'application

L'application est divisée en deux parties principales :

➤ Frontend (React.js)

- Pages principales :
 - Connexion / Inscription
 - Tableau de bord
 - Formulaire de déclaration d'incident
 - Liste des incidents
 - Détail d'un incident
 - Statistiques
- Composants React :
 - Header, Sidebar, IncidentCard, StatusBadge, etc.

➤ Backend (Node.js + Express.js)

- Routes API :
 - `/api/users` : gestion des utilisateurs
 - `/api/incidents` : gestion des incidents
 - `/api/auth` : connexion, token JWT
- Middlewares :
 - Authentification JWT
 - Gestion des rôles
 - Validation des données

Interfaces principaux

Page de connexion:

- Authentification via email et mot de passe.
- Génération d'un token JWT.

Formulaire de déclaration d'incident:

- Titre, description, priorité, catégorie.
- Ajout automatique de la date et de l'auteur.

Liste des incidents:

- Affichage sous forme de tableau avec filtres (état, technicien, priorité).
- Tri dynamique.

Statistiques:

- Graphiques montrant :
 - Nombre d'incidents par état
 - Temps moyen de résolution
 - Répartition par technicien

Sécurité et gestion des rôles

L'application implémente un système de rôles basé sur JWT :

- **Utilisateur** : peut déclarer un incident et consulter son historique.
- **Technicien** : voit les incidents qui lui sont assignés et change leur état.
- **Administrateur** : a tous les droits (ajout d'utilisateurs, affectation, statistiques...).

Les routes sensibles sont protégées via des middlewares d'autorisation.

Chapitre 5: Résultats ET tests

Scénarios de test

N°	Scénario testé	Résultat attendu	Statut
1	Connexion avec des identifiants valides	Accès au tableau de bord	✓ Réussi
2	Déclaration d'un incident	Incident ajouté à la base de données	✓ Réussi
3	Changement de l'état d'un incident	Mise à jour affichée en temps réel	✓ Réussi
4	Accès non autorisé à une route "admin"	Message d'erreur "Accès refusé"	✓ Réussi
5	Statistiques visibles uniquement par admin	Accès restreint confirmé	✓ Réussi
6	Création d'un utilisateur avec email existant	Message d'erreur : "Utilisateur déjà existant"	✓ Réussi

Résultats obtenus

- L'application développée répond globalement aux besoins identifiés dans le cahier des charges. Les objectifs suivants ont été atteints :
 - ✓ Interface claire et ergonomique
 - ✓ Authentification sécurisée via JWT
 - ✓ Gestion complète du cycle de vie d'un incident
 - ✓ Affectation des incidents à des techniciens
 - ✓ Système de rôles avec autorisation des accès
 - ✓ Statistiques dynamiques via des graphiques
 - ✓ Backend RESTful bien structuré

Problemes rencontrés et solutions

Problème	Solution apportée
Intégration difficile entre frontend/backend	Utilisation de CORS et Axios pour simplifier les appels
Authentification mal gérée au début	Implémentation de JWT + middlewares personnalisés
Rafraîchissement de l'interface après actions	Utilisation des hooks React (<code>useEffect</code>) et <code>setState</code>
Mauvaise lisibilité des erreurs serveur	Ajout de logs et messages d'erreur clairs côté backend

Conclusion

Le présent projet a permis de concevoir et de développer une application web de **gestion des incidents** basée sur la stack **MERN (Mongo DB, Express.js, React.js, Node.js)**.

L'application répond efficacement aux besoins identifiés au niveau du cahier des charges, en offrant une interface claire, un processus structuré de traitement des incidents, et une gestion des rôles bien définie.







Tout au long du développement, nous avons pu :

- Mettre en œuvre les bonnes pratiques de développement full-stack ;
- Concevoir une architecture moderne et évolutive ;
- Sécuriser l'application via l'authentification JWT ;
- Offrir un tableau de bord statistique pour les administrateurs ;
- Assurer un suivi en temps réel des incidents.

Ce projet a également été l'occasion d'approfondir nos compétences techniques, de maîtriser la stack MERN, et de mieux comprendre les enjeux liés à la gestion des services informatiques.

Perspectives

Pour aller plus loin, plusieurs pistes d'amélioration et d'évolution peuvent être envisagées :

-  Intégration d'un système de **notifications par email** pour alerter les utilisateurs en temps réel ;
-  Développement d'une version **mobile** de l'application (React Native) ;
-  Archivage automatique des incidents résolus depuis une certaine durée ;
-  Renforcement de la **sécurité** (captcha, 2FA, limitation des tentatives de connexion) ;
-  Déploiement sur un serveur cloud (ex : Heroku, Vercel, ou AWS) ;
-  Ajout de KPIs plus poussés et d'un module d'analyse prédictive via l'IA.

Bibliographie

1. *"Learning React: Modern Patterns for Developing React Apps"* – Alex Banks, Eve Porcello – O'Reilly, 2020
2. *"MongoDB: The Definitive Guide"* – Kristina Chodorow – O'Reilly, 2019
3. *"Node.js Design Patterns"* – Mario Casciaro, Luciano Mammino – Packt Publishing, 2020
4. *"Express.js Guide: The Comprehensive Book on Express.js"* – Azat Mardan, 2014

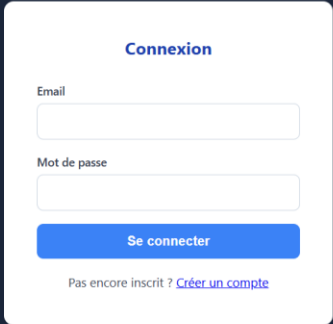
Documentation officielle

- React.js : <https://reactjs.org>
- Node.js : <https://nodejs.org>
- Express.js : <https://expressjs.com>
- MongoDB : <https://www.mongodb.com/docs>
- JWT : <https://jwt.io>
- Chart.js : <https://www.chartjs.org>

Annexes (captures d'écran, scripts...)

➤ Fenêtre d'authentification :

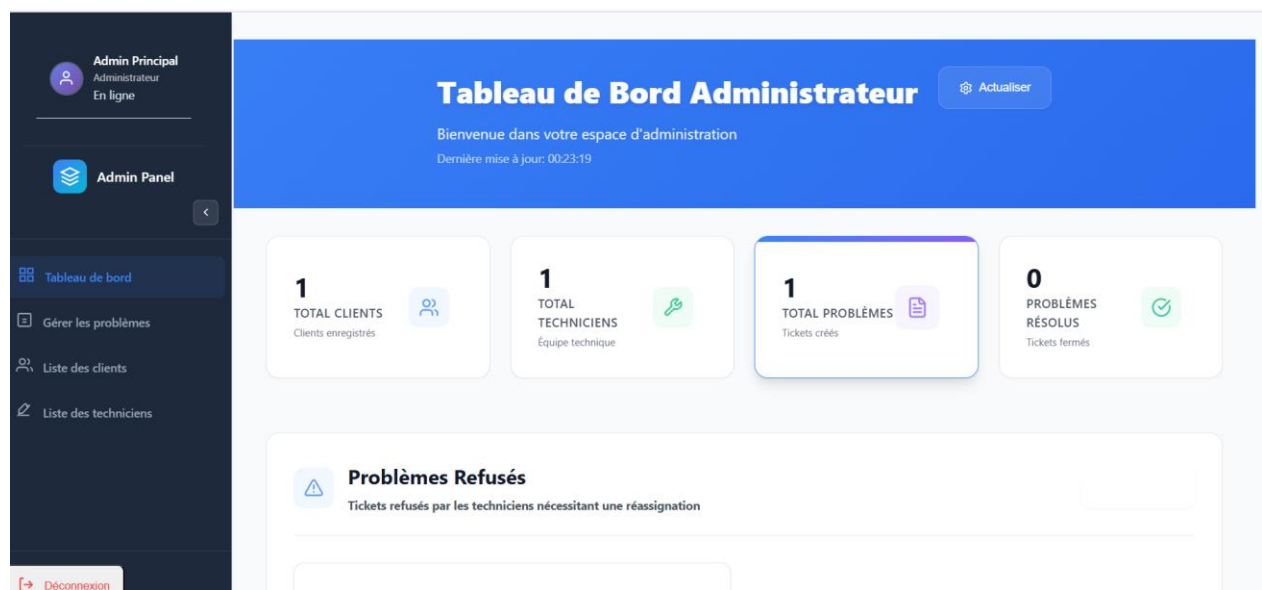
Fenêtre d'authentification, permet a l'administrateur ou l'opérateur de saisie de s'identifier pour avoir accès à un certains traitements spécifiques dans l'application.



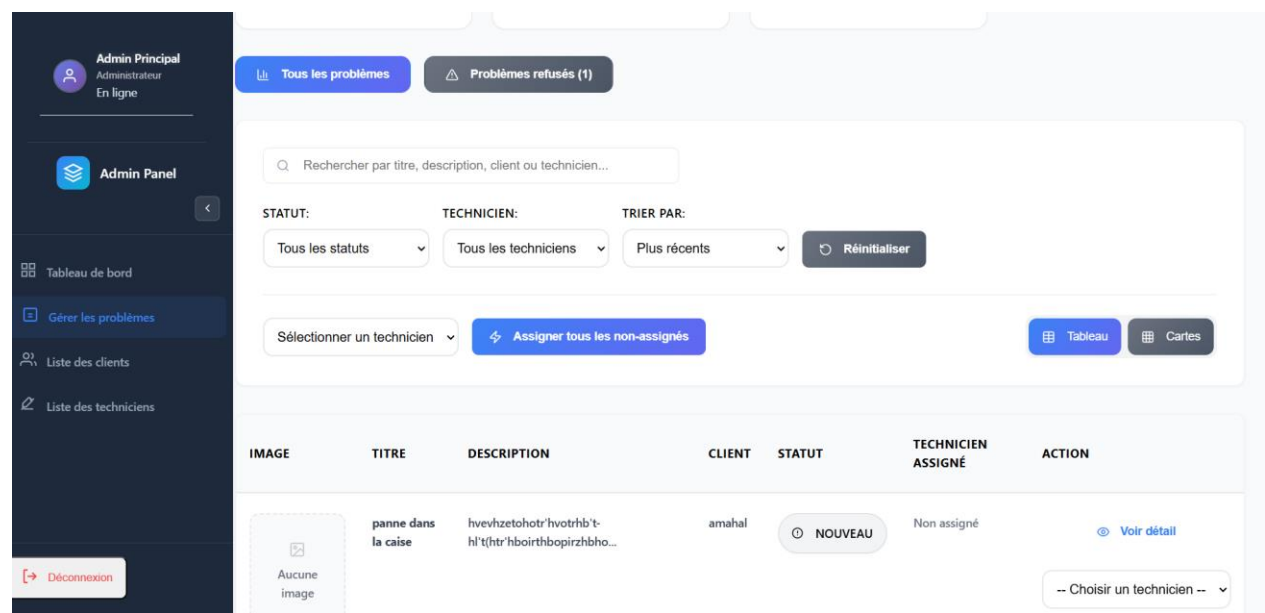
The screenshot shows a login window titled "Connexion" centered on a dark blue background. The window is white with rounded corners and contains the following elements:

- The title "Connexion" in blue text at the top.
- An "Email" label above a white text input field.
- A "Mot de passe" label above a white text input field.
- A blue button with the text "Se connecter" in white.
- A link at the bottom that reads "Pas encore inscrit ? [Créer un compte](#)".

Dashboard pour l'administrateur



Filtrage et la gestion des incidents



La listes des techniciens

Admin Principal
Administrateur
En ligne

Admin Panel

Tableau de bord

Gérer les problèmes

Liste des clients

Liste des techniciens

Déconnexion

Gérez tous vos techniciens depuis cette interface

Rechercher par nom ou email...

1 technicien trouvé

Technicien	Email	Date de création	Actions
othmane	othmaneam888@gmail.com	27/06/2025	

La liste des clients

Admin Principal
Administrateur
En ligne

Admin Panel

Tableau de bord

Gérer les problèmes

Liste des clients

Liste des techniciens

Déconnexion

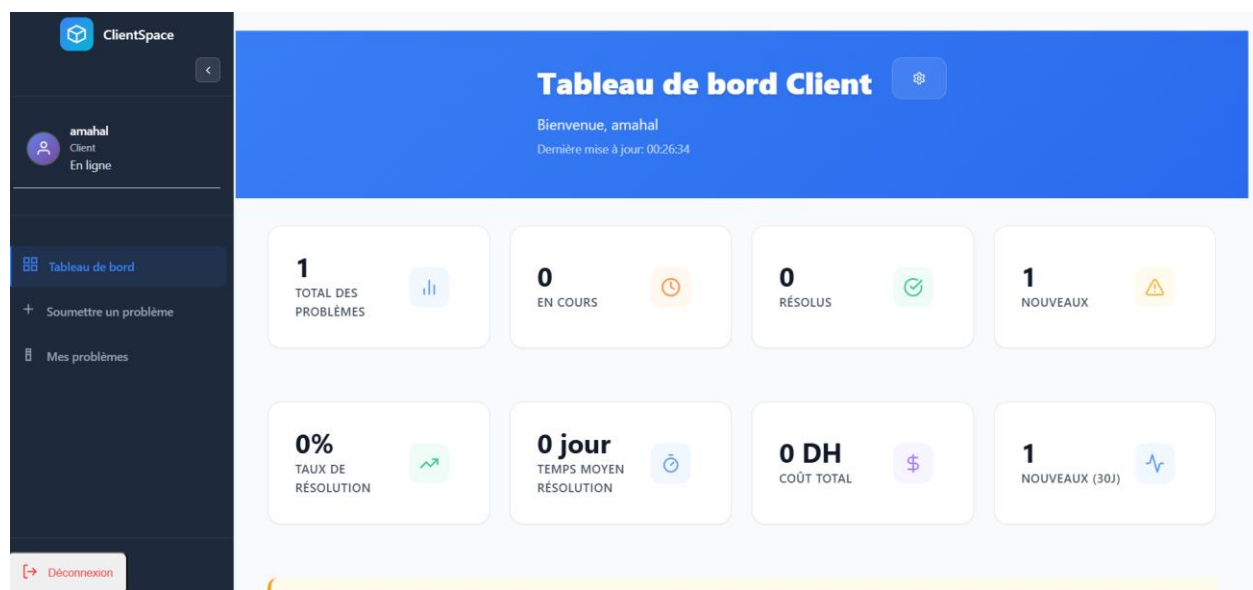
Gérez tous vos clients depuis cette interface

Rechercher par nom ou email...

1 client trouvé

Client	Email	Date de création	Actions
amahal	othmaneamahal00@gmail.com	27/06/2025	

Une Dashboard pour client



Un formulaire pour créer un problème

The screenshot shows the 'Créer un problème' form. It includes a sidebar on the left with the same menu as the dashboard. The form itself has three main sections: 'Titre' with a text input field, 'Description' with a larger text area, and 'Image (optionnelle)' with a dashed box containing an upload icon and text. At the bottom, there is a purple button labeled 'Envoyer le problème'.

Créer un problème

Titre

Décrivez votre problème en quelques mots...

Description

Décrivez votre problème en détail. Plus vous donnez d'informations, plus nous pourrons vous aider efficacement...

Image (optionnelle)

Cliquez pour ajouter une image
JPG, PNG, GIF - Max 5MB

Envoyer le problème

Une Dashboard pour les techniciens

