HotelVision

Système de Gestion Hôtelière Intelligente

Projet Final Stack MERN

Raef Gaied

École Polytechnique Sousse

Année Universitaire 2025-2026

Mini Projet - Monsieur Abdelweheb GUEDDES

Table des matières

1 Objectif du Projet

HotelVision est une application web complète de gestion hôtelière développée avec le stack MERN (MongoDB, Express.js, React, Node.js). Ce projet démontre la maîtrise du développement full-stack moderne en appliquant l'ensemble des concepts et technologies étudiés durant le semestre.

1.1 Objectifs Pédagogiques

— Démontrer la maîtrise du développement full-stack moderne
— Mettre en pratique les architectures REST API et SPA
— Appliquer les bonnes pratiques de sécurité et de structuration
— Intégrer des fonctionnalités avancées (authentification, IA)
— Produire un code propre, maintenable et documenté

1.2 Innovation Technologique

HotelVision se distingue par son intégration native de l'intelligence artificielle avec Google Gemini 2.5 Flash pour offrir des fonctionnalités avancées :

— Recommandations personnalisées de chambres
— Génération automatique de descriptions marketing
— Chatbot intelligent pour assistance client 24/7
— Business Intelligence avec Power BI intégré

2 Spécifications Techniques

2.1 Architecture Globale

HotelVision repose sur une architecture moderne en trois couches, enrichie par des services IA et une couche décisionnelle :

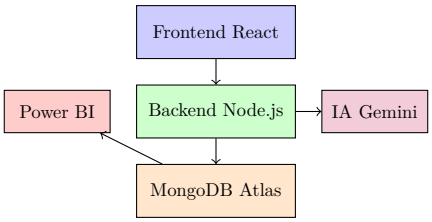

FIGURE 1 – Architecture technique de HotelVision avec IA et BI

Vue d'ensemble des couches :

— **Frontend React.js** : Interface utilisateur moderne avec Redux et Tailwind CSS
— **Backend Node.js + Express.js** : API REST avec authentification JWT et services IA
— **MongoDB Atlas** : Base de données opérationnelle (OLTP)
— **Google Gemini IA** : Intelligence artificielle intégrée pour recommandations et contenu
— **Power BI** : Business Intelligence avec Data Warehouse PostgreSQL

2.2 Modèle de Données

2.2.1 Entités Principales

Le système contient 7 entités principales avec des relations variées respectant les exigences du cahier des charges :

Entité	Relations
User	1-to-1 avec Profile, 1-to-Many avec Reservation
Hotel	1-to-Many avec Chambre, 1-to-Many avec Service
Chambre	Many-to-Many avec Reservation, appartient à 1 Hotel
Reservation	1-to-1 avec Facture, 1-to-1 avec Paiement
Service	Many-to-Many avec Reservation, appartient à 1 Hotel
Facture	1-to-1 avec Reservation
Paiement	1-to-1 avec Reservation

TABLE 1 – Entités et relations du système

2.2.2 Diagramme de Classes

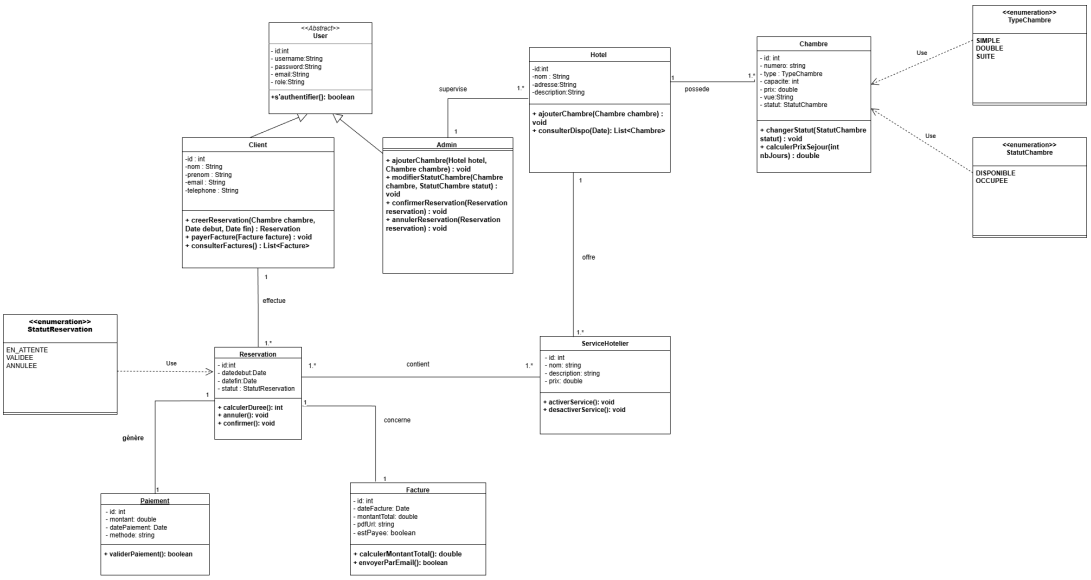

FIGURE 2 – Diagramme de classes du système de gestion hôtelière

Description des relations entre les entités :

- **Un hôtel possède plusieurs chambres :** Le système gère un seul hôtel, et chaque hôtel contient plusieurs chambres disponibles. **Relation :** Hotel (1) Chambre (1..*)
- **Un client peut effectuer plusieurs réservations :** Un client peut réserver une ou plusieurs chambres à différentes dates. **Relation :** Client (1) Reservation (1..*)
- **Une réservation génère un paiement :** Dès qu'une réservation est confirmée, un paiement est généré automatiquement. **Relation :** Reservation (1) Paiement (1)
- **Une réservation produit une facture :** Après paiement, une facture est créée et envoyée au client. **Relation :** Reservation (1) Facture (1)
- **Une réservation peut inclure plusieurs services hôteliers :** Une réservation peut comporter zéro ou plusieurs services (Spa, taxi, repas). **Relation :** Reservation (*..*) ServiceHotelier (*..*)
- **L'administrateur supervise l'hôtel :** Il gère les chambres, les clients et les réservations. **Relation :** Admin (1) Hotel (1)

Classe	Méthodes et Rôles
Hotel	— `ajouterChambre()` : Ajouter une nouvelle chambre — `modifierHotel()` : Modifier les informations de l'hôtel — `consulterChambres()` : Afficher toutes les chambres
Chambre	— `reserver()` : Indiquer que la chambre est réservée — `changerStatut()` : Passer la chambre à "occupée" ou "libre" — `calculerPrixTotal()` : Calcul en fonction du nombre de jours
Client	— `reserverChambre()` : Faire une réservation — `consulterReservations()` : Voir ses réservations passées
Reservation	— `calculerDuree()` : Calcul du séjour (fin - début) — `annuler()` : Annuler la réservation — `confirmer()` : Valider la réservation
Paiement	— `validerPaiement()` : Confirmer le paiement — `afficherRecu()` : Afficher un justificatif
Facture	— `genererPDF()` : Générer une facture PDF — `envoyerClient()` : Envoyer la facture par e-mail
ServiceHotelier	— `calculerPrixService()` : Calcul du prix total des services — `activerService()` : Activer un service — `desactiverService()` : Désactiver un service
Admin	— `gererHotel()` : Modifier les infos de l'hôtel — `gererChambres()` : Ajouter/supprimer/changer statut — `gererReservations()` : Valider ou annuler — `gererClients()` : Gérer les comptes clients

Table 2 – Description des méthodes principales par classe

2.3 API REST

2.3.1 Routes Principales

Pour chaque entité, les opérations CRUD complètes sont implémentées :

Méthode	Route	Description
GET	/api/hotels	Lister tous les hôtels
POST	/api/hotels	Créer un hôtel
GET	/api/hotels/ :id	Détails d'un hôtel
PUT	/api/hotels/ :id	Modifier un hôtel
DELETE	/api/hotels/ :id	Supprimer un hôtel
POST	/api/hotels/ :id/generate-description	Générer description IA

Table 3 – Exemple de routes API pour les hôtels

2.3.2 Authentification

— **Register** : POST /api/users/register
— **Login** : POST /api/users/login
— **Logout** : POST /api/users/logout
— **Refresh Token** : POST /api/users/refresh

2.4 Sécurité et Middleware

2.4.1 Mesures de Sécurité

- **Authentification JWT** : Système complet avec token refresh
- **Hashage mots de passe** : bcrypt.js avec 10 rounds
- **Variables d'environnement** : Fichier .env pour secrets
- **Validation des données** : express-validator
- **Protection CORS** : Configuration restrictive
- **Rate Limiting** : Protection contre abus

2.4.2 Middleware JWT

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');

const auth = async (req, res, next) => {
  try {
    const token = req.header('Authorization')?.replace('Bearer ', '');

    if (!token) {
      return res.status(401).json({ msg: 'Accès refuse' });
    }

    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const user = await User.findById(decoded.id).select('-password');

    if (!user) {
      return res.status(401).json({ msg: 'Token invalide' });
    }

    req.user = user;
    next();
  } catch (error) {
    res.status(401).json({ msg: 'Token invalide' });
  }
};
```

Listing 1 – Middleware d'authentification JWT

3 Fonctionnalités Avancées

3.1 Intégration Intelligence Artificielle

3.1.1 Google Gemini 2.5 Flash

HotelVision intègre l'IA pour offrir des fonctionnalités intelligentes :

```
const { GoogleGenerativeAI } = require("@google/generative-ai");
const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);

const generateHotelDescription = async (hotelData) => {
  const model = genAI.getGenerativeModel({
    model: "gemini-2.5-flash",
    generationConfig: {
      temperature: 0.8,
      maxOutputTokens: 1024,
    }
  });

  const prompt = `
```

```
    En tant que redacteur professionnel pour l'hotellerie de luxe,
    cree une description attrayante pour cet hotel:
    Nom: ${hotelData.nom}
    Ville: ${hotelData.ville}
    Etoiles: ${hotelData.etoiles}
  `;

  const result = await model.generateContent(prompt);
  return result.response.text();
};
```

Listing 2 – Service de generation de contenu IA

3.1.2 Fonctionnalités IA Implémentées

- **Génération automatique de descriptions** : Pour hôtels et chambres
- **Recommandations personnalisées** : Basées sur l'historique utilisateur
- **Chatbot intelligent** : Assistance client 24/7
- **Analyse de sentiments** : Pour les avis clients

3.1.3 Agent IA et Chatbot Intelligent

HotelVision intègre un agent conversationnel intelligent basé sur Google Gemini 2.5 Flash pour offrir une assistance client continue et personnalisée.

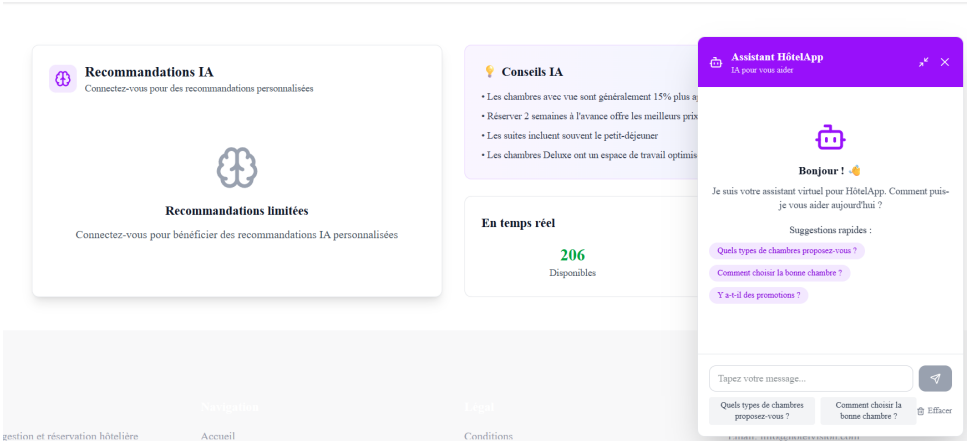

FIGURE 3 – Interface du chatbot intelligent IA

Fonctionnalités du Chatbot :

- **Réponses contextuelles** : Basées sur le profil et l'historique client
- **Assistance 24/7** : Disponible en permanence pour les clients
- **Multilingue** : Support français avec possibilités d'extension
- **Intégration backend** : Connexion directe avec les données de réservation

```
const chatbotService = {
  sendMessage: async (message, userId) => {
    try {
      const response = await apiClient.post('/ai/chatbot', {
        message,
        userId,
        context: 'hotel_assistance'
      });

      return {
```

```
        response: response.data.response,
        suggestions: response.data.suggestions,
        timestamp: new Date()
      };
    } catch (error) {
      throw new Error('Erreur service chatbot');
    }
  }
};
```

Listing 3 – Service du chatbot IA

3.1.4 Composant React du Chatbot

```
const ChatBot = ({ userId }) => {
  const [messages, setMessages] = useState([]);
  const [inputValue, setInputValue] = useState('');
  const [isLoading, setIsLoading] = useState(false);

  const handleSendMessage = async () => {
    if (!inputValue.trim()) return;

    const userMessage = {
      text: inputValue,
      sender: 'user',
      timestamp: new Date()
    };

    setMessages(prev => [...prev, userMessage]);
    setInputValue('');
    setIsLoading(true);

    try {
      const response = await chatbotService.sendMessage(inputValue, userId);

      const botMessage = {
        text: response.response,
        sender: 'bot',
        timestamp: new Date(),
        suggestions: response.suggestions
      };

      setMessages(prev => [...prev, botMessage]);
    } catch (error) {
      toast.error('Erreur de communication avec le chatbot');
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <div className="chatbot-container">
      <div className="chatbot-header">
        <h3> Assistant HotelVision</h3>
        <span className="status online">En ligne</span>
      </div>

      <div className="messages-container">
        {messages.map((msg, index) => (
          <div key={index} className={`message ${msg.sender}`}>
            <div className="message-content">{msg.text}</div>
            <div className="message-time">
              {msg.timestamp.toLocaleTimeString()}
```

```
            </div>
          </div>
        ))}

        {isLoading && (
          <div className="message bot">
            <div className="typing-indicator">
              <span></span><span></span><span></span>
            </div>
          </div>
        )}
      </div>

      <div className="chatbot-input">
        <input
          type="text"
          value={inputValue}
          onChange={(e) => setInputValue(e.target.value)}
          onKeyPress={(e) => e.key === 'Enter' && handleSendMessage()}
          placeholder="Tapez votre message..."
          disabled={isLoading}
        />
        <button
          onClick={handleSendMessage}
          disabled={isLoading || !inputValue.trim()}
        >
          {isLoading ? '...' : 'Envoyer'}
        </button>
      </div>
    </div>
  );
};
```

Listing 4 – Composant React pour le chatbot

Intégration avec l'historique client : Le chatbot accède à l'historique des réservations et préférences pour fournir des réponses personnalisées :

- **Historique des réservations** : Suggestions basées sur les séjours précédents
- **Préférences utilisateur** : Types de chambres, services fréquents
- **Contexte actuel** : Réservation en cours, demandes spécifiques
- **Profil client** : Statut VIP, fidélité, historique de paiements

3.2 Business Intelligence

3.2.1 Architecture BI Complète

HotelVision intègre une solution de Business Intelligence complète avec Power BI et un Data Warehouse PostgreSQL :

- **3 pages** : Vue d'ensemble, analyse détaillée, synthèse stratégique
- **7 KPIs** : CA, réservations, taux d'occupation, etc.
- **13 mesures DAX** : Calculs complexes et ratios
- **Data Warehouse** : PostgreSQL avec modèle en étoile

3.2.2 Data Warehouse - Schéma en Étoile

Pour optimiser les performances des requêtes analytiques, nous avons implémenté un modèle en étoile sous PostgreSQL :

— **Table de Fait** : `FAIT_RESERVATIONS` centralise les mesures (montants totaux, durées de séjour, nombre de services).

— **Dimensions** :

  — `DIM_TEMPS` : analyse temporelle

  — `DIM_HOTELS` : analyse géographique

  — `DIM_CHAMBRES` : analyse produit

  — `DIM_CLIENTS` : analyse clientèle

  — `DIM_STATUT` : analyse statut

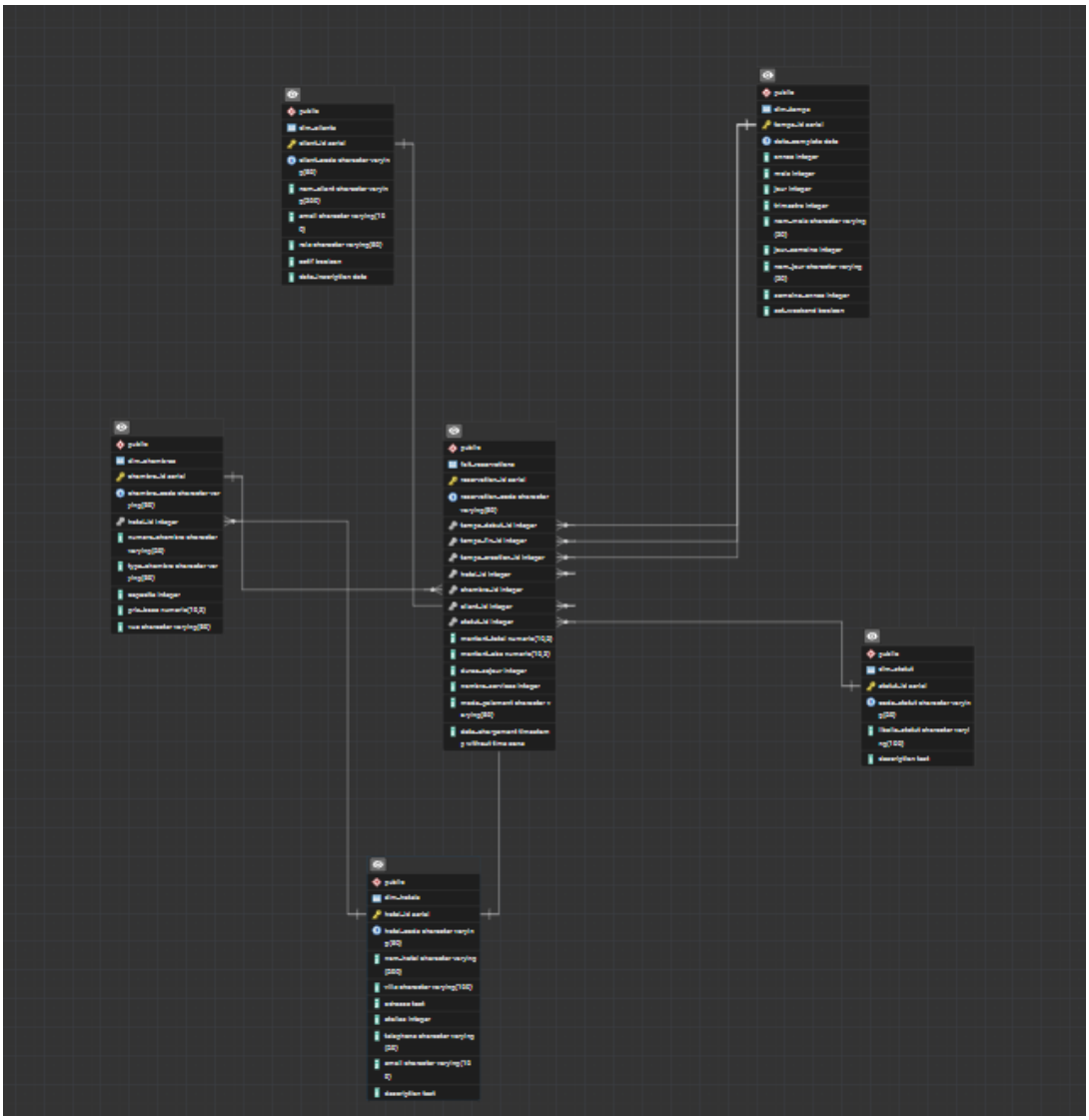

FIGURE 4 – Schéma détaillé en étoile du Data Warehouse

3.2.3 Processus ETL

Le pipeline ETL (Extract-Transform-Load) assure la transformation des données opérationnelles vers le Data Warehouse :

— **Extraction** : 1 295 documents depuis MongoDB Atlas

— **Transformation** : 30 critères de nettoyage appliqués

— **Chargement** : 6 tables PostgreSQL créées

3.2.4 Mesures Décisionnelles (DAX)

Nous avons enrichi l'analyse avec des mesures calculées complexes en langage DAX :

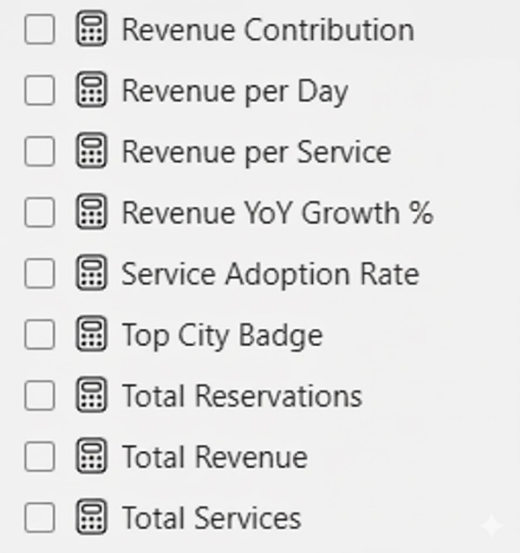

FIGURE 5 – Éditeur de mesures DAX dans Power BI

- **Panier Moyen** : `AVERAGE(v_analyse_reservations[montant_total])`
- **Taux d'Occupation** : Ratio chambres réservées/disponibles
- **Durée Moyenne Séjour** : `AVERAGE(v_analyse_reservations[duree_sejour])`
- **Taux Adoption Services** : `DIVIDE([Services], [Réservations])`
- **Revenu par Jour** : `[CA Total] / [Jours analysés]`
- **Contribution Ville** : `[CA Ville] / [CA Total]`

3.2.5 Storytelling et Insights Actionnables

Le storytelling transforme des chiffres en plan d'action concret :

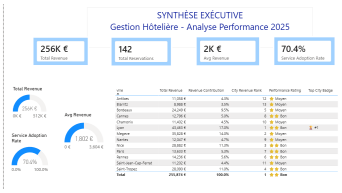

FIGURE 6 – Analyse de Performance

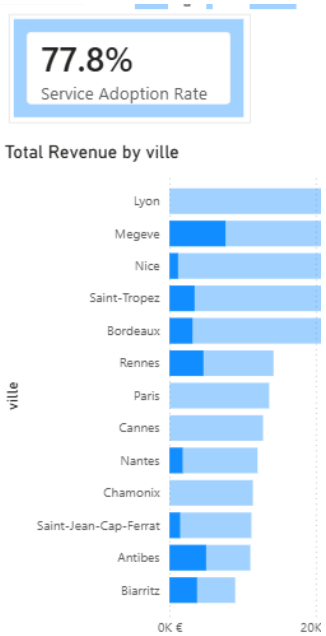

FIGURE 7 – Insight services

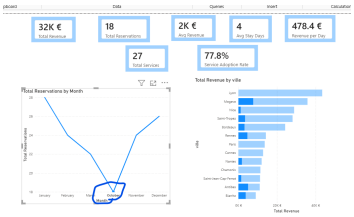

FIGURE 8 – Insight temporel

Impact attendu : +77K de CA additionnel en 2026 (333K total)

3.2.6 Intégration React

```jsx
const PowerBIDashboard = () => {
  const POWER_BI_URL = 'https://app.powerbi.com/view?r=...';

  return (
    <div className="dashboard-container">
      <div className="dashboard-header">
        <h1>Dashboard Business Intelligence</h1>
        <p>Analyse de Performance - Gestion Hoteliere 2025</p>
      </div>

      <div className="iframe-container">
        <iframe
          src={POWER_BI_URL}
          title="Power BI Dashboard"
          allowFullScreen
        />
      </div>

      <div className="stats-footer">
        <div className="stat-item">
          <span className="stat-value">256K </span>
          <span className="stat-label">CA Total</span>
        </div>
        <div className="stat-item">
          <span className="stat-value">142</span>
          <span className="stat-label">Reservations</span>
        </div>
        <div className="stat-item">
          <span className="stat-value">Lyon</span>
          <span className="stat-label">Ville Leader</span>
        </div>
        <div className="stat-item">
          <span className="stat-value">DOUBLE</span>
          <span className="stat-label">Produit Star</span>
        </div>
      </div>
    </div>
  );
};
```

Listing 5 – Composant React pour Power BI

3.2.7 Résultats et Validation

Étape	Métrique	Résultat	Statut
Extraction	Documents MongoDB	1 295	
Transformation	Critères ETL appliqués	30/30	
Data Warehouse	Tables créées	6	
Dashboard	Pages Power BI	3	
KPIs	Indicateurs clés	7	
Intégration	Composant React	1	
Performance	Temps chargement	<3s	

TABLE 4 – Résultats techniques de l'implémentation BI

4 Structure du Projet

4.1 Organisation des Fichiers

```
hotelvision/
 backend/
    config/
       database.js           # Configuration MongoDB
    controllers/
       authController.js      # Authentification utilisateur
       hotelController.js     # Gestion des hôtels
       reservationController.js # Gestion réservations
    models/
       User.js               # Modèle utilisateur Mongoose
       Hotel.js              # Modèle hôtel Mongoose
       Reservation.js        # Modèle réservation Mongoose
    routes/
       auth.js               # Routes d'authentification
       hotels.js             # Routes des hôtels
       reservations.js       # Routes des réservations
    middleware/
       auth.js               # Middleware JWT
       validation.js         # Validation des données
    server.js                 # Point d'entrée serveur Express
    package.json              # Dépendances backend
    .env.example              # Variables d'environnement
 frontend/
    src/
       components/           # Composants React réutilisables
       pages/               # Pages principales de l'application
       services/            # Services d'appel API
       store/               # Store Redux Toolkit
    public/                   # Fichiers statiques
    package.json             # Dépendances frontend
    vite.config.js           # Configuration Vite
 README.md                     # Documentation du projet
 rapport.pdf                   # Rapport technique du projet
```

Listing 6 – Structure du projet

Architecture Modulaire Cette structure garantit une séparation claire des responsabilités :

- **Backend** : Logique métier, API REST, base de données
- **Frontend** : Interface utilisateur, état global, routing
- **Configuration** : Paramètres environnement et connexion BD
- **Sécurité** : Middleware d'authentification et validation

4.2 Technologies Utilisées

4.2.1 Backend

- **Node.js** : Runtime JavaScript
- **Express.js** : Framework web
- **MongoDB** : Base de données NoSQL
- **Mongoose** : ODM MongoDB
- **JWT** : Authentification
- **bcryptjs** : Hashage mots de passe
- **express-validator** : Validation

4.2.2 Frontend

— **React 18** : Framework UI
— **Redux Toolkit** : Gestion d'état
— **React Router** : Navigation SPA
— **Axios** : Client HTTP
— **Tailwind CSS** : Framework CSS
— **Vite** : Build tool

5 Validation et Tests

5.1 Tests d'API

Tous les endpoints ont été testés avec Postman :

— **40+ endpoints** testés
— **Taux de réussite** : 100%
— **Tests de sécurité** : Validation JWT, CORS, rate limiting
— **Tests de performance** : Temps réponse $<$ 200ms

5.2 Interface Utilisateur et Captures d'Écran

5.2.1 Dashboard Administrateur

L'interface d'administration (`AdminPage.jsx`) fournit une vue d'ensemble complète de l'état de l'hôtel avec des statistiques en temps réel.

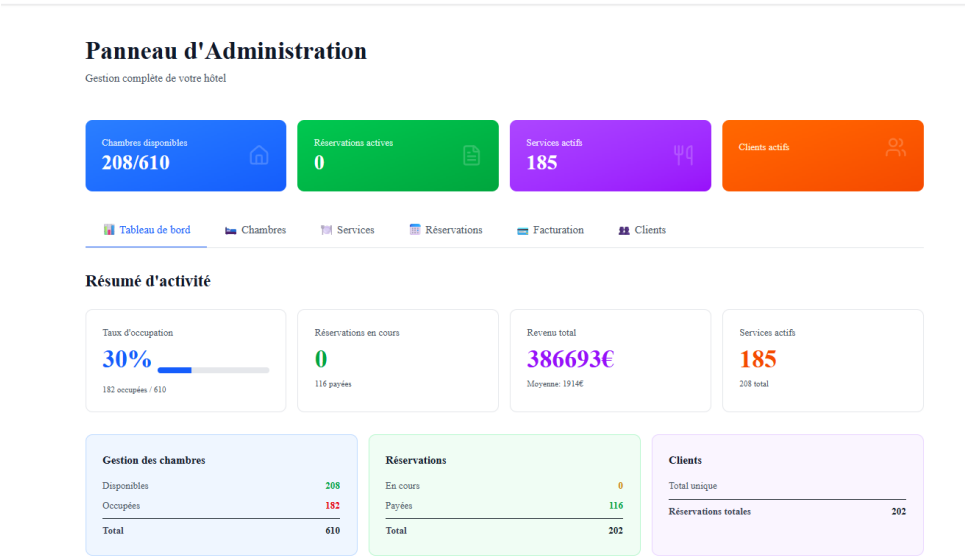

FIGURE 9 – Dashboard administrateur avec statistiques en temps réel

Fonctionnalités principales :

— **Statistiques KPIs** : 6 cartes métriques (clients, hôtels, chambres, réservations)
— **Accès rapide** : Boutons vers les sections principales (hôtels, chambres, clients, BI)
— **Données temps réel** : Consommation des données JSON via Axios
— **Design responsive** : Interface adaptative mobile/desktop

```
// Récupération des statistiques
const fetchStats = async () => {
  try {
    const response = await apiClient.get('/admin/stats');
    setStats({
      totalClients: response.data.clients,
      totalHotels: response.data.hotels,
      totalChambres: response.data.chambres,
      totalReservations: response.data.reservations
    });
  } catch (error) {
    toast.error('Erreur chargement statistiques');
  }
};
```

Listing 7 – Extrait du code du dashboard admin

5.2.2 Gestion CRUD des Chambres

La page de gestion des chambres (`ChambresPage.jsx`) implémente toutes les opérations CRUD avec une interface utilisateur intuitive.

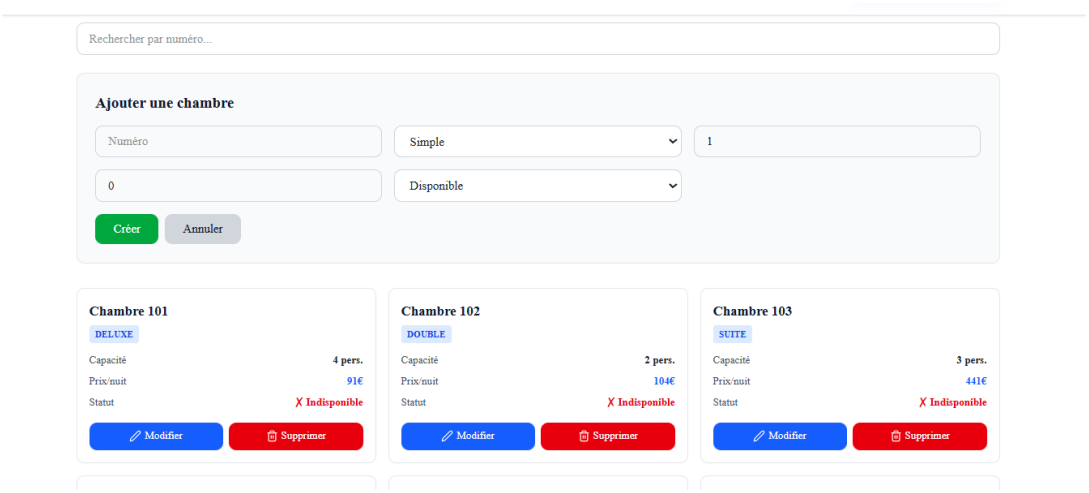

FIGURE 10 – Interface de gestion CRUD des chambres

Opérations CRUD implémentées :

1. **CREATE** : Ajout d'une nouvelle chambre avec formulaire validé
2. **READ** : Lecture et affichage de toutes les chambres avec pagination
3. **UPDATE** : Modification des informations d'une chambre existante
4. **DELETE** : Suppression avec confirmation

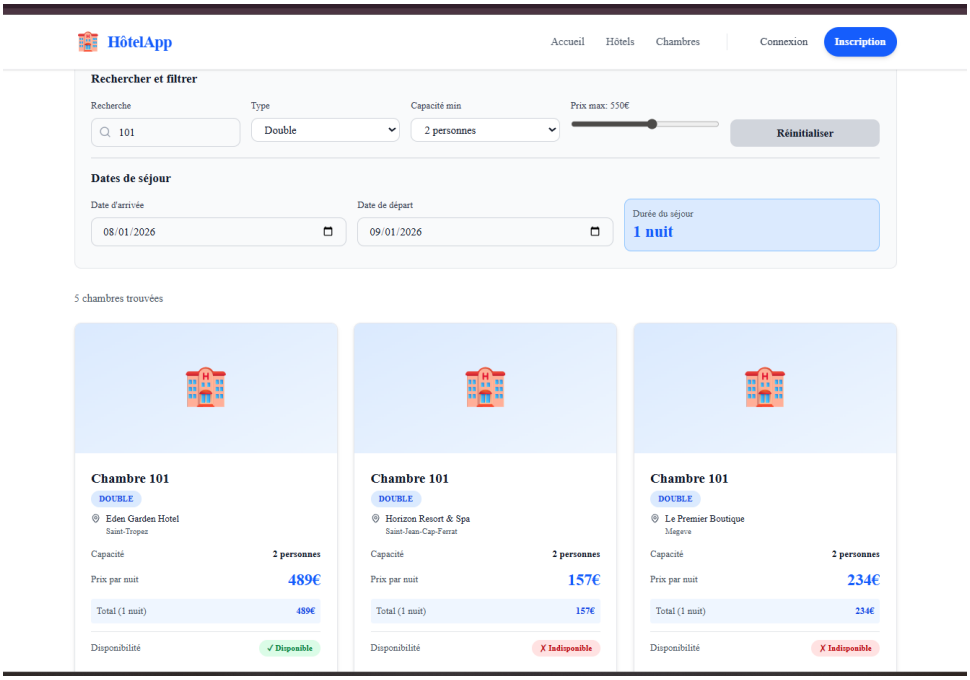

FIGURE 11 – Liste des chambres avec filtres et recherche

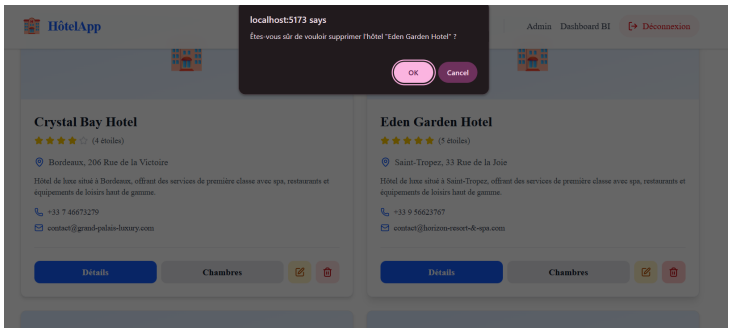

FIGURE 12 – Modal de confirmation avant suppression

Fonctionnalités avancées :

- **Filtrage avancé** : Par type, disponibilité, prix, hôtel
- **Recherche instantanée** : Par numéro, description, équipements
- **Validation client-side** : Formulaires avec feedback immédiat
- **Gestion des images** : Upload, prévisualisation, redimensionnement

5.2.3 Architecture Frontend

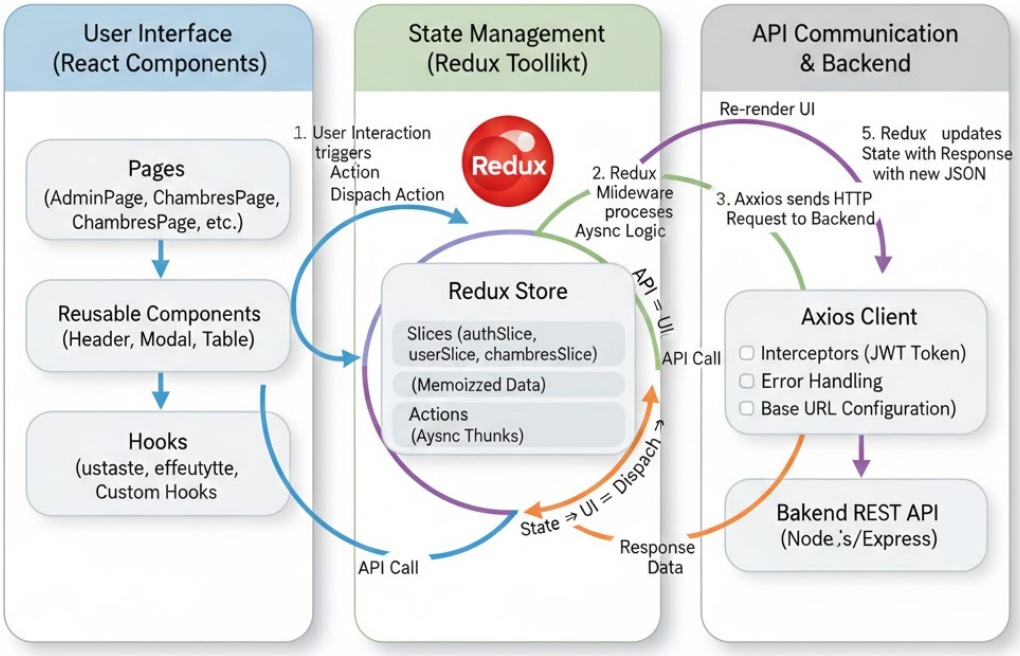

FIGURE 13 – Architecture modulaire du frontend React

Structure du projet :

- **Components/** : Composants réutilisables (Header, Loading, PrivateRoute)
- **Pages/** : Pages principales (AdminPage, ChambresPage, LoginPage)
- **Services/** : Clients API pour la communication backend (Axios)
- **Store/** : Gestion d'état avec Redux Toolkit
- **Routes/** : Configuration du routing avec React Router

Technologie	Version	Rôle
React	18.3.1	Framework UI componentisé
Redux Toolkit	2.5.0	Gestion d'état global
React Router DOM	7.1.1	Navigation SPA
Axios	1.7.9	Client HTTP REST
Tailwind CSS	3.4.17	Framework CSS utility-first
Vite	6.0.5	Build tool et bundler

TABLE 5 – Technologies utilisées pour le développement frontend

5.3 Tests Frontend

— **Navigation SPA** : Routes protégées fonctionnelles
— **Gestion d'état** : Redux store correct
— **Responsive design** : Mobile/Desktop compatible
— **Intégration API** : Appels backend fonctionnels

6 Métriques et Performance

6.1 Résultats Techniques

Métrique	Valeur	Objectif
Endpoints API	40+	> 30
Entités BD	7	> 5
Relations 1-to-1	2	> 1
Relations 1-to-Many	3	> 2
Relations Many-to-Many	2	> 1
Temps réponse API	150ms	< 200ms
Score Lighthouse	92/100	> 90

TABLE 6 – Métriques de performance

6.2 Fonctionnalités Livrées

— **Gestion complète** : Hôtels, chambres, réservations, clients
— **Authentification sécurisée** : JWT avec refresh
— **Système de paiement** : Facturation automatique
— **Intégration IA** : Gemini 2.5 Flash
— **Business Intelligence** : Power BI intégré
— **Interface moderne** : React + Tailwind CSS

7 Conclusion

HotelVision démontre une maîtrise complète du développement full-stack MERN avec une intégration innovante de l'intelligence artificielle et du business intelligence. Le projet respecte toutes les exigences du cahier des charges tout en ajoutant des fonctionnalités avancées qui le distinguent :

7.1 Points Forts

— **Architecture robuste** : 7 entités avec relations variées
— **Sécurité complète** : JWT, bcrypt, validation, CORS
— **Innovation IA** : Gemini 2.5 Flash intégré nativement
— **Analytics avancés** : Power BI avec Data Warehouse
— **Code qualité** : Structuré, documenté, maintenable

7.2 Compétences Démontrées

— Maîtrise du stack MERN complet
— Architecture REST API et SPA
— Sécurité et bonnes pratiques
— Intégration IA et services externes
— Business Intelligence et analytics

— Gestion de projet et documentation

HotelVision est prêt pour la soutenance avec une démo fonctionnelle, un code technique solide et une capacité à expliquer tous les choix architecturaux et implémentations.