



الكلية المتعددة التخصصات بالعرائش

FACULTE POLYDISCIPLINAIRE A LARACHE

Master DevOps et Cloud Computing

Projet de Fin de Module

Architecture orientée services et APIs

Gestion hôtelière intelligente : réservations  
et organisation interne

Réalisé Par

Zahra JBARI  
Imane BAIDER  
Hanae BERTILI  
Salwa ELGHAILANI

Encadré Par

Pr. Mohamed Walid Hajoub

# *Dédicaces*

Nous dédions ce projet à nos chers parents, pour leur amour inconditionnel, leur soutien constant et leurs encouragements tout au long de notre parcours académique.

Nous l'adressons également à nos frères et sœurs, ainsi qu'à nos amis, pour leur présence rassurante, leurs précieux conseils et leur soutien moral durant toutes les étapes de ce travail.

Nous remercions tout particulièrement notre équipe pour leur engagement et leur collaboration fructueuse dans la réalisation de ce projet : Imane, Salwa, Zahra et Hanae, chacune ayant contribué par ses compétences et son travail assidu au succès de ce projet.

Enfin, nous nous dédions ce projet à nous-mêmes, comme symbole de notre détermination, de notre passion et de notre engagement envers la réussite de notre formation et la réalisation de nos objectifs professionnels.

## Table des matières

<b>Introduction Générale .....</b>	<b>7</b>
<b>Chapitre 1 : Contexte général .....</b>	<b>8</b>
<b>1. Introduction .....</b>	<b>8</b>
<b>2. Contexte du projet.....</b>	<b>8</b>
<b>3. Description du projet .....</b>	<b>8</b>
<b>4. Cadrage logique du projet .....</b>	<b>9</b>
<b>4.1. Objectifs généraux :.....</b>	<b>9</b>
<b>4.2. Méthodologie de conception et mise en œuvre.....</b>	<b>9</b>
<b>4.3. Portée du système.....</b>	<b>10</b>
<b>5. Méthodologie et approche.....</b>	<b>10</b>
<b>Chapitre 2 : État de l'art.....</b>	<b>12</b>
<b>1. Introduction .....</b>	<b>12</b>
<b>2. Étude des solutions actuelles de gestion hôtelière.....</b>	<b>12</b>
<b>3. Analyse des technologies et frameworks utilisés.....</b>	<b>13</b>
<b>3.1. Planification et organisation (Plan) .....</b>	<b>13</b>
<b>3.2. Développement des microservices et interfaces .....</b>	<b>14</b>
<b>3.2.1. Développement Back-end : .....</b>	<b>14</b>
<b>3.2.2. Développement Frontend : .....</b>	<b>15</b>
<b>3.3. Bases de Données :.....</b>	<b>15</b>
<b>3.4. Communication Interservices et Frontend :.....</b>	<b>15</b>
<b>3.5. Documentation et Gestion des API .....</b>	<b>16</b>
<b>3.6. Infrastructure et Déploiement : .....</b>	<b>16</b>
<b>3.6.2. Gestion du Versionnement :.....</b>	<b>17</b>
<b>3.6.3. Tests et Validation.....</b>	<b>17</b>
<b>3.6.3.1. Vérification du système :.....</b>	<b>17</b>
<b>3.6.3.2. Sécurité :.....</b>	<b>17</b>
<b>3.6.3.3. Prometheus et Grafana : .....</b>	<b>18</b>
<b>Chapitre 3 : Conception du système .....</b>	<b>19</b>
<b>1. Introduction .....</b>	<b>19</b>
<b>2. Modélisation des acteurs et de leurs fonctions.....</b>	<b>19</b>
<b>2.1. Utilisateurs internes : .....</b>	<b>19</b>
<b>2.2. Utilisateurs externes : Clients.....</b>	<b>23</b>
<b>3. Conception et Méthodologie de Développement.....</b>	<b>23</b>
<b>3.1. Conception UML : .....</b>	<b>23</b>
<b>3.2. Application du Domain-Driven Design (DDD) : .....</b>	<b>28</b>
<b>Chapitre 4 : Implémentation, intégration et validation du système .....</b>	<b>32</b>

<b>1. Architecture Technique :</b>	<b>32</b>
1.1 Développement Back-end :	33
1.2 Développement Front-end :	37
1.3 Bases de Données :	40
1.4. Communication entre microservices et interface utilisateur (Front-end) :	41
1.5 Architecture des Services :	41
1.6 Documentation et Gestion des API :	42
1.7 Infrastructure et Déploiement :	43
➤ <i>Notifications et Retour d'Information</i> .....	<b>46</b>
1.8 Monitoring avec Prometheus et Grafana :	46
<b>2. Présentation des Interfaces Applicatives :</b>	<b>52</b>
2.1. Page d'accueil de la plateforme ROYELLAS HOTEL.....	52
2.2. Page d'Inscription .....	55
2.3. Interface d'authentification .....	55
2.4. Espace Client .....	56
2.6. Espace réceptionniste .....	69
2.7. Espace Manager .....	73
2.8. Espace Housekeeping .....	77
2.9. Espace Admin .....	79
<b>Conclusion .....</b>	<b>87</b>
<b>Perspective .....</b>	<b>88</b>
<b>Webographie : .....</b>	<b>89</b>

## Table des Figures

Figure 1 : interface Trello .....	14
Figure 2 :Diagramme de classe.....	25
Figure 3 : Diagramme de cas d'utilisation d'administrateur .....	25
Figure 4 : Diagramme de cas d'utilisation Client.....	26
Figure 5 : Diagramme de cas d'utilisation Housekeeping .....	26
Figure 6 : Diagramme de cas d'utilisation Réceptionniste .....	27
Figure 7 : Diagramme de cas d'utilisation Manager.....	27
Figure 8 : Diagramme de cas d'utilisation Personnel Maintenance .....	28
Figure 9 : Diagramme de cas d'utilisation Comptable.....	28
Figure 10 : Processus d'application de la méthodologie DDD.....	29
Figure 11 : Cartographie des Contextes Métiers selon DDD.....	31
Figure 12 : Architecture Microservices avec API Gateway et Découverte de Services.....	32
Figure 13 : Structure du Projet BackEnd (microservices) .....	37
Figure 14 : Structure du Projet Frontend Angular (Gestion_hoteliere) .....	38
Figure 15 : Tableau des Services Inscrits dans Eureka .....	42
Figure 16 : exemple d'interface swagger pour la gestion des utilisateurs.....	43
Figure 17 : Tableau de Supervision des Conteneurs Docker en Production .....	43
Figure 18 : État des Pods Kubernetes en Temps Réel .....	44
Figure 19 : Tableau de bord SonarQube – Analyse du Service Interne.....	44
Figure 20 : Vue détaillée des étapes du pipeline CI/CD avec temps d'exécution .....	45
Figure 21 : Représentation graphique du pipeline CI/CD dans Jenkins .....	46
Figure 22 : notifications automatiques Build Success .....	46
Figure 23 : Interface de Surveillance avec Prometheus – Tableau de Bord des Métriques en Temps Réel.....	47
Figure 24 : Alerte Prometheus – Détection d'Erreurs 401 Élevées sur le Microservice de Sécurité ....	48
Figure 25 : Interface Prometheus – Requêtes et Métriques HTTP des Microservices.....	48
Figure 26 : Graphique Prometheus – Surveillance de l'Utilisation CPU en Temps Réel .....	48
Figure 27 : Page de Connexion à l'Interface Grafana .....	49
Figure 28 : Configuration de la Source de Données Prometheus dans Grafana .....	50
Figure 29 : Tableau de Bord Grafana – Métriques de Performance et Surveillance Système .....	50
Figure 30 : Vue Synthétique du Dashboard Grafana – Indicateurs de Performance Système et CPU..	51
Figure 31 : Alerte Grafana – Absence de Données (DatasourceNoData) dans les Règles de Sécurité .	51
Figure 32 : Interface Web du Site ROYELLA SHOTEL .....	52
Figure 33 :Pages de Présentation et Services.....	53
Figure 34 : Catalogue des Chambres et Présentation de la Restauration.....	54
Figure 35 : Pied de Page (Footer) du Site ROYELLA SHOTEL .....	54
Figure 36 : Page d'Inscription .....	55
Figure 37 :Page d'authentification .....	55
Figure 38 : Interface Client – Catalogue des Chambres avec Recherche et Filtrage.....	56
Figure 39 : Recherche par numéro de chambre.....	57
Figure 40 : Recherche par type de chambre .....	57
Figure 41 : Recherche par prix.....	58
Figure 42 : Page de détail de la chambre .....	58
Figure 43 : Chambres similaires.....	59
Figure 44 : Page Profile client.....	59

Figure 45 : Page pour modifier profile Client .....	60
Figure 46 : Page de détail – Chambre Double 4 (Statut : Occupée).....	60
Figure 47 : Page de détail – Chambre Single 2 (Statut : Maintenance).....	61
Figure 48 : Page de détail – Suite Room 7 (Statut : Disponible).....	61
Figure 49 : Formulaire de Réservation en Ligne.....	62
Figure 50 : Profil Utilisateur avec Suivi de Réservation.....	62
Figure 51 : Espace Client – Profil Utilisateur avec Suivi de Réservation .....	62
Figure 52 : Section Notifications avec Suivi de Réservation.....	63
Figure 53 : Notification de Réservation Rejetée .....	63
Figure 54 : Notification de Réservation Confirmée avec Accès au Paiement .....	64
Figure 55 : Interface de gestion des paiements des réservations.....	64
Figure 56 : Paiement réussi .....	67
Figure 57 : Profil du réceptionniste .....	69
Figure 58 : Modifier profile Réceptionniste .....	69
Figure 59 : page chambres ( gérer les chambres) .....	70
Figure 60 : suivi les Statistiques des chambres .....	71
Figure 61 : Page Réservations (gérer les réservations et suivi les statistiques) .....	72
Figure 62 : Mise à jour statut de la chambre .....	72
Figure 63 : modifier le statut d'une chambre.....	73
Figure 64 : Dashboard de Manager .....	73
Figure 65 : Profile Manager.....	74
Figure 66 : Modifier le Profile Manager .....	74
Figure 67 : Liste des chambres .....	75
Figure 68 : formulaire pour Crée une nouvelle chambre .....	75
Figure 69 : chambre crée avec succès.....	75
Figure 70 : formulaire pour modifier une chambre existante .....	76
Figure 71 : L'interface de gestion des services internes .....	77
Figure 72 : L'interface profile Housekeeping .....	77
Figure 73 : l'interface pour modifier profile Housekeeping .....	78
Figure 74 : Dashboard de Housekeeping .....	78
Figure 75 :L'interface chambre pour Housekeeping .....	79
Figure 76: L'interface de Profile Administrateur.....	79
Figure 77 :L'interface pour gérer les utilisateurs internes .....	80
Figure 78 : Mise à jour le statut d'une utilisateur .....	80
Figure 79 : seule une connexion active permet l'accès au système.....	81
Figure 80 : L'interface pour gérer les rôles et les permissions.....	82
Figure 81 : Création d'un rôle.....	83
Figure 82 : Création d'une permission .....	84
Figure 83 : Étape d'Assignation d'une Permission à un Rôle (GATEKEEPER) .....	85
Figure 84 : Confirmation de l'Assignation – Permission 'OPEN DOOR' au Rôle GATEKEEPER .....	85
Figure 85 : Tableau de Bord Administrateur – Liste et Statistiques des Réservations.....	86
Figure 86 : Tableau de Bord Administrateur – Liste et Statistiques des Réservations.....	86

# Introduction Générale

Le secteur hôtelier évolue dans un environnement de plus en plus compétitif, marqué par une transformation numérique rapide et des exigences croissantes en matière de qualité de service, de performance opérationnelle et de personnalisation de l'expérience client. Dans ce contexte, la gestion efficace des ressources, des réservations et des interactions clients constitue un enjeu stratégique majeur pour les établissements hôteliers.

Face à ces exigences, l'adoption de solutions informatiques modernes devient indispensable afin d'automatiser les processus internes, assurer la fiabilité des données et garantir la scalabilité des systèmes d'information. Les architectures logicielles traditionnelles montrent aujourd'hui leurs limites, notamment en termes de flexibilité, d'évolutivité et de maintenance.

C'est dans cette optique que s'inscrit le projet ROYELLAS HOTEL, une application de gestion hôtelière conçue autour d'une architecture microservices. Cette solution vise à fournir une plateforme modulaire et évolutive permettant la gestion des réservations, des chambres et des services hôteliers, tout en offrant des interfaces adaptées aux différents profils d'utilisateurs, tels que les réceptionnistes et les gestionnaires.

Le projet repose sur l'utilisation de technologies récentes telles que Spring Boot, Flask et Docker, garantissant une meilleure séparation des responsabilités, une facilité de déploiement et une maintenance optimisée. À travers cette approche, ROYELLAS HOTEL propose une solution digitale cohérente et performante, répondant aux besoins actuels des établissements hôteliers en matière de gestion et de modernisation des systèmes d'information.

# Chapitre 1 : Contexte général

## 1. Introduction

Dans ce chapitre, nous présentons le contexte du projet ROYELLAS HOTEL, dont l'objectif est de concevoir et de développer un système digital intégré de gestion hôtelière. Ce projet vise à répondre aux besoins opérationnels des établissements hôteliers en proposant une plateforme centralisée permettant la gestion des réservations, des chambres, de la facturation ainsi que la communication entre les différents acteurs du système.

Le projet ROYELLAS HOTEL s'inscrit dans une démarche de modernisation des systèmes d'information hôteliers, en mettant à disposition des fonctionnalités adaptées aux usages quotidiens des hôtels. Il a pour ambition de faciliter les tâches du personnel, d'améliorer la fiabilité des données et d'optimiser les processus internes à travers une solution simple, efficace et évolutive.

## 2. Contexte du projet

Le projet ROYELLAS HOTEL consiste à développer un système digital de gestion hôtelière intelligent, destiné à améliorer l'organisation interne et la qualité des services des établissements hôteliers. Cette application web permet de centraliser et automatiser toutes les opérations quotidiennes de l'hôtel, depuis la gestion des réservations et des chambres, jusqu'à la facturation et la coordination du personnel.

Le système s'adresse à différents types d'utilisateurs : les clients, qui peuvent réserver et gérer leur séjour en ligne ; les réceptionnistes, responsables de l'accueil et du suivi des chambres ; le personnel housekeeping et maintenance, chargés de l'entretien et du bon fonctionnement des installations ; le manager, qui supervise l'ensemble des opérations ; le comptable, pour le suivi financier ; et l'administrateur, qui assure la sécurité et la configuration du système.

ROYELLAS HOTEL repose sur une architecture microservices organisée autour de plusieurs domaines métiers (réservations, chambres, clients, facturation, services internes, administration et sécurité). Chaque domaine est autonome mais interagit avec les autres pour garantir une gestion efficace et cohérente de l'hôtel, tout en assurant la sécurité, la performance et la fiabilité du système.

Cette plateforme innovante vise à offrir une expérience utilisateur fluide et intuitive, améliorer la coordination entre les différents services de l'hôtel, réduire les erreurs humaines, et fournir des outils d'aide à la décision grâce à des rapports et statistiques détaillés. Ainsi, ROYELLA SHOTEL représente une solution moderne, évolutive et performante pour accompagner la transformation numérique des hôtels.

## 3. Description du projet

Le système de gestion ROYELLA SHOTEL est une application modulaire conçue pour automatiser et simplifier les opérations quotidiennes d'un hôtel, tout en garantissant la fiabilité, la sécurité et la performance. Les principales fonctionnalités incluent :

### A. Gestion des réservations :

- Création de nouvelles réservations en fonction de la disponibilité des chambres.

- Modification ou mise à jour des réservations existantes pour répondre aux besoins des clients.
- Annulation des réservations avec **mise à jour automatique de la disponibilité des chambres**.

#### **B. Gestion des chambres :**

- Suivi en temps réel de l'état des chambres (libres, occupées, en nettoyage, en maintenance).
- Classification des chambres selon leurs catégories (standard, luxe, suite).
- Planification et suivi des opérations de maintenance et de nettoyage.

#### **C. Gestion des clients et facturation :**

- Enregistrement et suivi des informations clients et de l'historique des séjours.
- Suivi détaillé des paiements et génération automatique des factures, incluant les services supplémentaires.
- Intégration avec des systèmes de paiement électronique sécurisés.

#### **D. Notifications et alertes :**

- Envoi d'alertes automatisées aux clients concernant leurs réservations, paiements et services.
- Notifications internes pour le personnel concernant les tâches prioritaires (check-in, maintenance, housekeeping).

#### **E. Gestion des utilisateurs et sécurité :**

- Gestion des rôles et permissions pour tous les types d'utilisateurs (clients, réceptionnistes, manager, personnel technique).
- Authentification sécurisée et contrôle d'accès pour garantir la confidentialité des données.

#### **F. Rapports et statistiques :**

- Génération de rapports détaillés pour la prise de décision et le suivi des performances de l'hôtel.
- Statistiques sur le taux d'occupation, chiffre d'affaires, et performance des services internes.

### **4. Cadrage logique du projet**

Ce projet vise à concevoir et développer un système intégré de gestion hôtelière, offrant une solution centralisée et efficace pour la gestion quotidienne des opérations, avec un accent sur l'automatisation, l'organisation et l'expérience utilisateur.

#### **4.1. Objectifs généraux :**

- Automatiser les processus hôteliers pour réduire la charge de travail manuel et minimiser les erreurs.
- Améliorer l'expérience client en offrant des services personnalisés et réactifs.
- Rationaliser la gestion interne en optimisant les flux de travail et les communications.

#### **4.2. Méthodologie de conception et mise en œuvre**

Le projet repose sur une architecture modulaire et basée sur les microservices, permettant de diviser le système en unités indépendantes, facilitant ainsi le développement, la maintenance et l'évolution future.

Chaque module est responsable d'un domaine spécifique de l'hôtel et communique avec les autres via des interfaces clairement définies, garantissant l'intégration et l'efficacité opérationnelle.

#### 4.3. Portée du système

- Aspects couverts : Gestion complète des réservations, des chambres, des paiements et des communications internes.
- Plateforme modulaire : Chaque fonctionnalité est développée comme un module indépendant pouvant être facilement ajouté ou mis à jour.
- Scalabilité : Adaptation pour les petits établissements comme pour les chaînes hôtelières de grande envergure.

### 5. Méthodologie et approche

Afin d'assurer la réussite du projet ROYELLA SHOTEL, nous avons adopté une approche rigoureuse, itérative et collaborative, basée sur les bonnes pratiques du génie logiciel. Dès le lancement du projet, nous avons utilisé Trello comme outil de gestion de projet pour organiser le travail en équipe, répartir les tâches entre les membres, suivre l'avancement et respecter les délais.

Les principales étapes de notre méthodologie sont les suivantes :

#### 1. Analyse des besoins :

- Identification des besoins des utilisateurs principaux (réceptionnistes, clients, gestionnaires).
- Recueil des exigences fonctionnelles et techniques.
- Priorisation des fonctionnalités clés telles que la gestion des réservations et des chambres.

#### 2. Conception et modélisation :

- Utilisation des diagrammes UML (cas d'utilisation, classes) pour modéliser le système.
- Définition d'une architecture basée sur les principes du **Domain-Driven Design (DDD)**.

#### 3. Développement itératif :

- Implémentation des fonctionnalités par cycles courts avec livraisons incrémentales.
- Utilisation de **Spring Boot**, **Nodejs** et **Flask** pour le développement des microservices.
- Mise en place d'API **REST** pour une communication standardisée.

#### 4. Tests et validation :

- Réalisation de tests unitaires et d'intégration pour garantir la qualité du système.
- Validation des fonctionnalités avec des utilisateurs finaux dans des scénarios réels.

## **5. Déploiement et mise en production :**

- Utilisation de **Docker** et **Kubernetes** pour le déploiement en environnement cloud.
- Garantie de la scalabilité, de la sécurité et de la disponibilité du système.

## **6. Amélioration continue :**

- Collecte des retours utilisateurs pour améliorer les fonctionnalités existantes.
- Ajout de nouveaux modules selon les besoins émergents.

# Chapitre 2 : État de l'art

## 1. Introduction

La présente revue de littérature vise à analyser les systèmes d'information touristiques, et plus particulièrement les systèmes de gestion hôtelière, afin d'identifier les solutions existantes, les technologies mobilisées ainsi que les meilleures pratiques adoptées dans ce domaine. Cette analyse permet de mettre en évidence les forces et les limites des approches actuelles, tout en fournissant un cadre de référence scientifique pour orienter la conception et le développement du système ROYELLA SHOTEL.

Les travaux récents soulignent que l'intégration des systèmes numériques dans le secteur hôtelier contribue significativement à l'amélioration de l'efficacité opérationnelle et de la qualité des services. Cette évolution repose notamment sur l'automatisation des processus de réservation, la gestion en temps réel de l'état des chambres et l'optimisation des opérations de facturation et de paiement. Ces fonctionnalités permettent une meilleure organisation interne et une réduction des erreurs liées aux traitements manuels.

Par ailleurs, la littérature met en avant l'importance croissante des architectures modulaires, en particulier les architectures microservices, dans la conception de systèmes flexibles et évolutifs. Ces architectures facilitent l'adaptation aux besoins spécifiques des établissements hôteliers, tout en améliorant la maintenabilité et la scalabilité des applications.

Enfin, plusieurs études soulignent le rôle stratégique des technologies modernes telles que les interfaces REST, le cloud computing et la conteneurisation à travers Docker dans l'amélioration de la performance, de la fiabilité et de la sécurité des systèmes d'information hôteliers. L'adoption de ces technologies favorise le développement de solutions robustes et évolutives, capables de soutenir efficacement les activités des établissements hôteliers.

Dans cette perspective, le projet ROYELLA SHOTEL s'inscrit dans une démarche visant à tirer parti des meilleures pratiques et des avancées technologiques identifiées dans la littérature afin de concevoir un système intégré, sécurisé et évolutif, répondant aux exigences actuelles du secteur hôtelier.

## 2. Étude des solutions actuelles de gestion hôtelière

Les systèmes de gestion hôtelière (Property Management Systems – PMS) constituent aujourd'hui des outils centraux dans la modernisation des établissements hôteliers. Ils permettent une gestion intégrée des réservations, des chambres, de la facturation et de la relation client, contribuant ainsi à l'optimisation des processus internes et à l'amélioration de la qualité des services.

Parmi les solutions les plus reconnues dans ce domaine figurent :



Développé par Oracle, Opera PMS est considéré comme l'une des solutions les plus complètes et les plus répandues à l'échelle internationale. Il offre une gestion centralisée des réservations,

des paiements et des inventaires, tout en se distinguant par ses capacités avancées d'intégration avec des systèmes tiers et ses fonctionnalités de reporting détaillé, destinées à soutenir la prise de décision stratégique.

#### **Cloudbeds**

Lancé en 2012, Cloudbeds est une plateforme cloud destinée principalement aux hôtels indépendants et aux petites chaînes. Elle se caractérise par une interface conviviale et propose, en plus de la gestion des réservations, des outils de tarification dynamique ainsi qu'une intégration avec les principales plateformes de réservation en ligne telles que Booking.com et Expedia.

#### **Hotelogix**

Hotelogix est un système cloud conçu pour les établissements hôteliers de petite et moyenne taille. Il se distingue par sa simplicité d'utilisation et par des fonctionnalités telles que le suivi des disponibilités en temps réel, la gestion des paiements et la génération de rapports personnalisés, facilitant ainsi le pilotage opérationnel et stratégique.

#### **RoomRaccoon**

Ce PMS met l'accent sur l'automatisation des opérations hôtelières, en proposant des fonctionnalités telles que le check-in en ligne, la gestion des revenus et l'intégration avec les systèmes de distribution multicanaux. Il contribue ainsi à l'amélioration de l'efficacité opérationnelle et à l'optimisation des performances commerciales.

#### **Mews**

Fondé en 2012, Mews est une solution moderne axée sur l'expérience utilisateur, caractérisée par une interface intuitive et des outils avancés pour la gestion des clients, des paiements et des opérations quotidiennes. Son architecture cloud-native favorise la flexibilité, la scalabilité et l'intégration avec d'autres services numériques.

Ces systèmes constituent des références majeures pour la compréhension des besoins et des attentes des acteurs du secteur hôtelier. L'analyse de leurs fonctionnalités et de leurs limites a permis d'orienter la conception du système ROYELLA SHOTEL vers une architecture modulaire, évolutive et orientée vers l'interopérabilité, en adéquation avec les exigences contemporaines des établissements hôteliers.

### **3. Analyse des technologies et frameworks utilisés**

Dans le cadre du développement du système de gestion hôtelière ROYELLA SHOTEL, une étude approfondie et exhaustive des différentes technologies et frameworks a été réalisée afin de garantir la robustesse, la scalabilité et la facilité d'intégration avec d'autres systèmes. Cette analyse a permis d'identifier les outils logiciels les plus adaptés pour répondre aux exigences de performance élevée, de sécurité des données et de maintenance future. Elle a également établi les fondations techniques nécessaires à la conception d'un système intégré et évolutif, capable de satisfaire les besoins des établissements modernes tout en améliorant efficacement l'expérience utilisateur finale.

#### **3.1. Planification et organisation (Plan)**

La phase de planification et d'organisation constitue la première étape essentielle pour le succès de tout projet de développement logiciel. Dans ce projet, les objectifs du projet ont été clairement définis, le travail a été divisé en tâches gérables, et les priorités pour chaque tâche ont été établies. Les responsabilités ont été réparties entre les membres de l'équipe afin de garantir une collaboration efficace et d'assurer l'avancement de toutes les composantes du projet.

Nous avons utilisé **Trello** comme principal outil de gestion des tâches. Chaque tâche a été représentée par une carte, avec le responsable assigné, les délais estimés et l'état d'avancement. Cette approche a permis de suivre le progrès avec précision, d'éviter les retards et de garantir l'atteinte des objectifs dans les délais impartis.

De plus, nous avons appliqué les techniques de Design Thinking durant la phase de planification afin de centrer les solutions sur les besoins réels des utilisateurs finaux, de stimuler l'innovation et de proposer des méthodes flexibles pour résoudre les problèmes tout au long du développement.

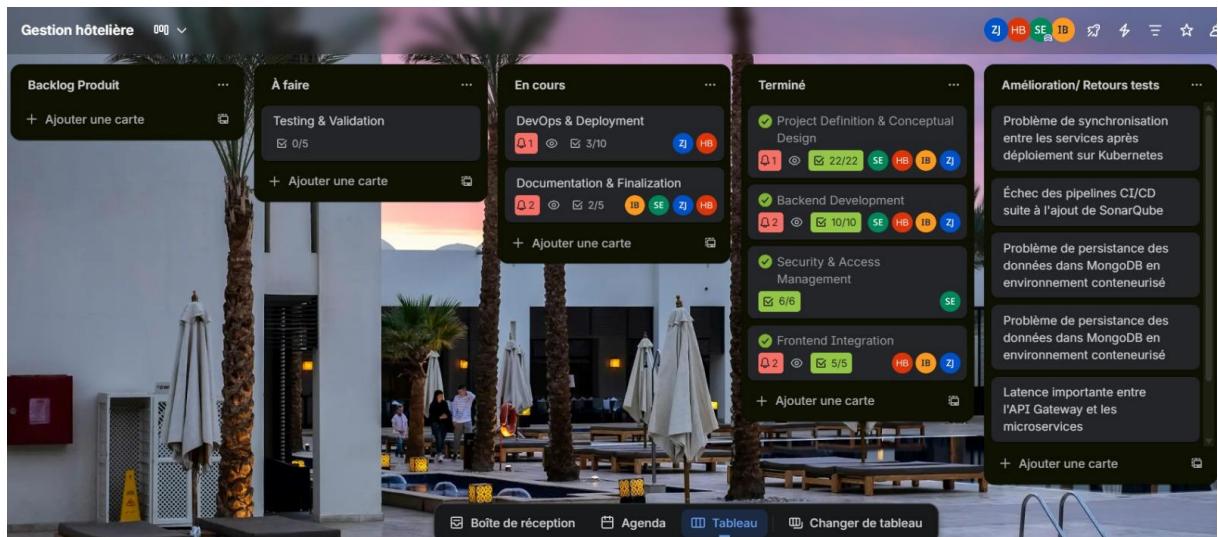


Figure 1 : interface Trello

### 3.2. Développement des microservices et interfaces

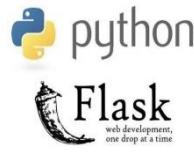
#### 3.2.1. Développement Back-end :

##### Java - Spring Boot :



Spring Boot est utilisé pour développer des microservices robustes et scalables. Grâce à sa configuration automatique et son vaste écosystème, il simplifie la création d'applications RESTful et d'API sécurisées. Son intégration avec Spring Security permet de gérer l'authentification et l'autorisation via JWT, tout en offrant un support pour les bases de données relationnelles et NoSQL.

#### Python - Flask :



Flask est un micro-framework léger et flexible pour Python, idéal pour la création d'API simples et performantes. Sa simplicité d'utilisation, combinée à sa capacité à gérer des requêtes HTTP et à s'intégrer facilement avec des bases de données et des systèmes externes, en fait un choix judicieux pour des microservices nécessitant des réponses rapides et une faible surcharge.

#### Node.js :



Node.js est une plateforme côté serveur open-source qui permet d'exécuter du JavaScript en dehors du navigateur. Elle est utilisée pour développer des applications web rapides, évolutives et asynchrones, notamment des API et des microservices.

### 3.2.2. Développement Frontend :

#### Angular :



Angular est un framework front-end puissant développé par Google, idéal pour la création d'applications web dynamiques. Il utilise HTML pour structurer les vues, CSS pour la mise en page et TypeScript pour un développement robuste et typé. Grâce à son architecture basée sur les composants, Angular permet de structurer les applications de manière modulaire et réutilisable. Son intégration fluide avec les API REST et GraphQL facilite la récupération des données du backend et leur affichage interactif, offrant ainsi une expérience utilisateur fluide et réactive.

### 3.3. Bases de Données :

#### Base de données relationnelle (MySQL) :

MySQL est une base de données relationnelle populaire, offrant des performances élevées et une gestion des transactions ACID. Elle est utilisée dans ce projet pour gérer les données transactionnelles (réservations, informations sur les chambres), assurant une forte cohérence et des requêtes complexes.



#### Base de données NoSQL (MongoDB) :

MongoDB est une base de données NoSQL orientée document, adaptée aux données non structurées ou semi-structurées. Dans notre service de paiement, elle permet de gérer efficacement les informations de transaction grâce à son modèle JSON, facilitant l'évolution du schéma et offrant une scalabilité horizontale pour traiter un grand nombre de paiements simultanément.

### 3.4. Communication Interservices et Frontend :

#### REST API :



Les API RESTful sont utilisées pour la communication entre les microservices et pour l'interaction avec le frontend. Elles permettent de définir des points d'entrée pour chaque ressource du système, favorisant une architecture sans état (stateless) et facilitant l'intégration avec des systèmes externes. Grâce à la simplicité du protocole HTTP et aux méthodes standards (GET, POST, PUT, DELETE), REST assure une communication rapide et fiable entre les différents composants du système.

### 3.5. Documentation et Gestion des API

#### Swagger :



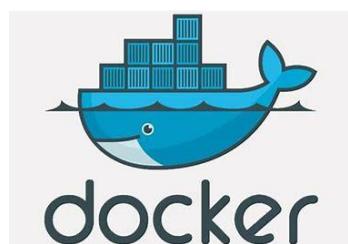
l'interface web.

Swagger est un ensemble d'outils pour la documentation et la gestion des API REST. Il permet de générer automatiquement une documentation interactive et lisible pour les développeurs et les utilisateurs de l'API. Cette documentation inclut des descriptions détaillées des endpoints, des types de données acceptés et renvoyés, ainsi que des exemples de requêtes et de réponses. Swagger facilite la collaboration entre les équipes de développement et permet de tester rapidement les API depuis

### 3.6. Infrastructure et Déploiement :

#### 3.6.1. Conteneurisation et Orchestration :

#### Docker :



Docker est utilisé pour conteneuriser chaque microservice, garantissant ainsi que l'environnement de développement, de test et de production soit identique, réduisant ainsi les risques d'incohérences et d'erreurs. Cela facilite également le déploiement rapide de chaque service sur n'importe quelle infrastructure, qu'il s'agisse d'une machine locale, d'un serveur ou d'un cloud.

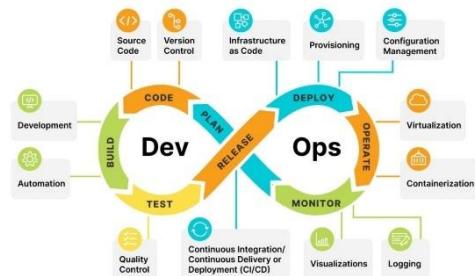
#### Kubernetes :



configurations.

Kubernetes est un système d'orchestration de conteneurs utilisé pour automatiser le déploiement, la gestion, et la mise à l'échelle des applications. Kubernetes permet de gérer efficacement les microservices en assurant leur haute disponibilité, leur scalabilité et leur tolérance aux pannes. Grâce à Kubernetes, il est possible de déployer, mettre à jour et maintenir les microservices avec une gestion simplifiée des ressources et des

#### CI/CD Pipeline :



jour est validée avant d'être déployée.

#### Jenkins



Jenkins

Jenkins est un outil open-source d'intégration continue (CI) et de déploiement continu (CD). Il permet d'automatiser les processus de compilation, tests et déploiement des applications, garantissant ainsi des livraisons plus rapides, fiables et sécurisées tout en réduisant les risques d'erreurs humaines.

#### SonarQube

SonarQube est une plateforme open-source d'analyse continue de la qualité du code. Elle permet de détecter automatiquement les bugs, vulnérabilités, codes dupliqués et problèmes de maintenabilité, afin d'améliorer la qualité globale du logiciel et de faciliter sa maintenance sur le long terme.

### 3.6.2. Gestion du Versionnement :

#### Git :

Git est utilisé pour le contrôle de version du code source. Il permet de suivre les modifications du code, de collaborer efficacement entre les membres de l'équipe et de gérer les différentes versions du projet. Les branches Git permettent de travailler sur des fonctionnalités spécifiques sans perturber la version principale du projet, facilitant ainsi le travail en équipe.

### 3.6.3. Tests et Validation

#### 3.6.3.1. Vérification du système :

Des tests unitaires, d'intégration, fonctionnels et d'acceptation sont réalisés pour s'assurer que le système respecte les exigences et fonctionne correctement. Les tests permettent de détecter les bugs tôt dans le processus de développement, garantissant ainsi la stabilité du système avant sa mise en production.

#### 3.6.3.2. Sécurité :

#### Spring Security et JWT :



serveur. Cette approche assure la sécurité des échanges tout en améliorant les performances et la scalabilité du système.

#### OAuth2



dans l'accès aux services.

Spring Security est utilisé pour sécuriser l'accès aux microservices et gérer l'authentification des utilisateurs. Dans ce projet, nous avons employé JWT (JSON Web Token) pour fournir une méthode d'authentification stateless (sans état). Lorsqu'un utilisateur se connecte, un jeton JWT est généré et transmis au client, permettant au backend de vérifier l'authentification lors des requêtes suivantes sans avoir besoin de maintenir une session côté

OAuth2 est un protocole d'autorisation sécurisé qui permet à des applications tierces d'accéder aux ressources d'un utilisateur sans partager ses identifiants. En s'appuyant sur des tokens d'accès, OAuth2 fournit une méthode d'authentification flexible et sécurisée, facilitant l'intégration avec des fournisseurs d'identité externes tels que Google ou Facebook. Cette approche garantit à la fois la protection des données de l'utilisateur et la souplesse

#### 3.6.3.3.Prometheus et Grafana :



Prometheus est utilisé pour la collecte de métriques système en temps réel, offrant une vue d'ensemble sur la santé du système et ses performances. Grafana est ensuite utilisé pour visualiser ces métriques à travers des tableaux de bord interactifs. Ensemble, Prometheus et Grafana permettent de surveiller les applications et d'identifier rapidement les problèmes de performance ou de disponibilité.

# Chapitre 3 : Conception du système

## 1. Introduction

Le système ROYELLA SHOTEL repose sur une architecture modulaire basée sur les microservices, permettant de séparer les différentes fonctionnalités en services indépendants mais interconnectés. Cette approche assure une scalabilité optimale, une grande flexibilité et une facilité de maintenance, tout en garantissant la sécurité des données et la performance des services.

Le présent chapitre est consacré à la conception du système. Il a pour objectif de présenter de manière détaillée les choix techniques et architecturaux effectués, ainsi que l'organisation des différents composants logiciels. L'accent est mis sur la structuration des microservices, la modélisation des données, la communication entre les services, ainsi que sur les méthodes utilisées pour garantir une intégration efficace avec le frontend et une expérience utilisateur fluide.

Plus précisément, ce chapitre couvre :

La modélisation du système à l'aide de diagrammes UML, permettant de représenter les entités, leurs relations et les flux de données.

La conception détaillée des microservices, incluant le rôle de chaque service, leur organisation et leur interaction avec le frontend et entre eux.

Les principes de conception adoptés, tels que le Domain-Driven Design (DDD), pour structurer les domaines métiers et assurer la cohérence globale du système.

Ce chapitre constitue ainsi la base de la compréhension technique du projet et prépare le lecteur à appréhender le développement, l'intégration et le déploiement des différentes fonctionnalités du système.

## 2. Modélisation des acteurs et de leurs fonctions

Le système ROYELLAS HOTEL est conçu pour gérer efficacement les opérations hôtelières tout en offrant une expérience utilisateur fluide et sécurisée. Il prend en charge deux types principaux d'utilisateurs :

- Les utilisateurs internes : ce groupe comprend les réceptionnistes, les Managers, le personnel Housekeeping et le Personnel Maintenance. Chacun de ces acteurs dispose de rôles spécifiques permettant de gérer les réservations, les chambres, la facturation, le suivi du personnel et la sécurité du système.
- Les utilisateurs externes : ce groupe regroupe les clients de l'hôtel, qui peuvent accéder à un portail en ligne pour effectuer des réservations, consulter leurs informations personnelles et gérer leurs paiements de manière autonome.

L'identification claire de ces acteurs et de leurs rôles est essentielle pour concevoir un système modulable, sécurisé et adapté aux besoins de chaque utilisateur, garantissant ainsi une gestion optimale et une expérience personnalisée pour tous les usagers.

### 2.1. Utilisateurs internes :

### Réceptionniste :

Le réceptionniste est responsable de la gestion quotidienne des clients et des réservations à la réception, assurant ainsi le bon déroulement des opérations hôtelières et la satisfaction des clients.

#### Fonctions principales :

- Gestion du profil réceptionniste :
  - Consulter et modifier ses informations personnelles et paramètres de compte de manière sécurisée.
- Gestion des chambres :
  - Visualiser le tableau de bord des chambres pour suivre en temps réel leur statut (disponible, occupée, en maintenance).
  - Effectuer un check-in et check-out manuel des clients, en mettant à jour l'état des chambres et les informations de séjour.
- Gestion des réservations :
  - Lister les réservations en attente, confirmées ou rejetées.
  - Voir la liste des réservations en attente et les traiter rapidement.
  - Approuver ou rejeter une réservation client selon la disponibilité et les politiques de l'hôtel.
  - Consulter les statistiques liées aux réservations pour une meilleure planification et suivi.
- Tableau de bord et suivi :
  - Consulter les statistiques générales (occupation, revenus, performance des services) via une interface centralisée.
  - Accéder à une vue synthétique des opérations courantes pour une prise de décision rapide et éclairée.
- Authentification et sécurité :
  - S'authentifier de manière sécurisée pour accéder à l'interface réceptionniste et garantir la confidentialité des données clients.

### **Manager / Responsable de l'hôtel**

Le manager supervise l'ensemble des activités de l'hôtel et assure la prise de décisions stratégiques afin d'optimiser la performance globale et la qualité des services fournis aux clients.

#### Fonctions principales :

- Gestion du profil Manager :

- Consulter et mettre à jour les informations personnelles et professionnelles du manager.
- Supervision via le Dashboard :
- Consulter le tableau de bord pour suivre les opérations essentielles de l'hôtel, incluant les statistiques des réservations, du personnel et des chambres.
- Surveiller les réservations et les revenus afin d'assurer une gestion financière efficace.
- Générer des rapports détaillés sur le taux d'occupation, le chiffre d'affaires et la performance des services.
- Gestion des chambres :
- Lister toutes les chambres avec leurs informations détaillées (type, prix, état).
- Créer de nouvelles chambres dans le système.
- Modifier les caractéristiques des chambres existantes.
- Supprimer des chambres si nécessaire.
- Voir l'état de disponibilité des chambres et suivre leur occupation en temps réel.
- Gestion du personnel interne :
- Voir l'état de disponibilité du personnel interne (présence, statut, affectations).
- Gérer les utilisateurs du système, incluant les réceptionnistes et le personnel hôtelier.
- Contrôler la performance globale des services et du personnel pour garantir un service de haute qualité et la satisfaction des clients.
- Authentification et sécurité :
- S'authentifier de manière sécurisée pour accéder à l'interface dédiée.

### **Personnel Housekeeping (Ménage / Netteté)**

Le personnel housekeeping est chargé de maintenir les chambres propres, disponibles et prêtées à l'usage, contribuant ainsi à la satisfaction et au confort des clients.

#### Fonctions principales :

- Authentification :
- S'authentifier de manière sécurisée pour accéder à l'interface dédiée au personnel de nettoyage.
- Gestion du profil :
- Consulter le profil : Visualiser ses informations personnelles et professionnelles.

- Modifier le profil (extension) : Mettre à jour ses coordonnées et paramètres de compte.
- Tableau de bord (Dashboard) :
  - Consulter le tableau de bord pour une vue d'ensemble des tâches et de l'état des chambres.
  - Inclut la visualisation des indicateurs clés : chambres à nettoyer, chambres en cours, tâches terminées.
- Gestion des chambres :
  - Consulter la liste des chambres avec leur statut (à nettoyer, en cours, nettoyée).
  - Marquer une chambre comme nettoyée après intervention, ce qui met à jour automatiquement son statut dans le système et la rend disponible pour de nouvelles réservations.

### **Personnel Maintenance (Technique / Technique Hôtelier)**

Le personnel de maintenance assure le bon fonctionnement technique des chambres et des installations de l'hôtel, garantissant sécurité et confort pour les clients et le personnel.

#### Fonctions principales :

- Recevoir et traiter les demandes de maintenance des différents services.
- Identifier et analyser les pannes ou dysfonctionnements techniques.
- Effectuer les réparations et résoudre les problèmes signalés.
- Mettre à jour le statut des interventions pour un suivi efficace.
- Déclarer les chambres opérationnelles après réparation.
- Enregistrer la date, la durée et les détails des interventions pour la traçabilité.

### **⊕ Comptable / Service financier**

Le comptable est responsable de la gestion financière et comptable de l'hôtel, assurant la cohérence entre les réservations, les paiements et la facturation.

#### Fonctions principales :

- Vérifier et gérer les factures des clients.
- Suivre les paiements et effectuer les remboursements si nécessaire.
- Contrôler et analyser les rapports financiers.
- Établir des rapports comptables périodiques pour la direction.
- Assurer la cohérence et l'exactitude entre les réservations effectuées et la facturation correspondante.

### **⊕ Administrateur Système (Admin)**

L'administrateur système est responsable de la gestion technique, de la sécurité et de la stabilité de l'application. Il assure le bon fonctionnement du système et veille à la protection des données tout en supervisant les opérations critiques.

Fonctions principales :

- Créer, modifier et supprimer les comptes utilisateurs.
- Définir et gérer les rôles ainsi que les permissions associées.
- Assurer la sécurité globale du système et des données.
- Surveiller les journaux d'activité (logs) et effectuer les sauvegardes régulières.
- Intervenir en cas d'incident technique ou de dysfonctionnement pour maintenir la disponibilité du système.
- Superviser et visualiser les informations liées aux chambres et aux réservations à travers un tableau de bord (dashboard), permettant un suivi centralisé et une prise de décision rapide.

## **2.2. Utilisateurs externes : Clients**

Le client est l'utilisateur final de l'application, qui utilise la plateforme pour réserver et gérer son séjour à l'hôtel de manière autonome et sécurisée. Il constitue la principale source d'interactions avec le système et bénéficie d'un accès direct aux services proposés.

Fonctions principales :

- Consulter la liste des chambres disponibles, avec leurs caractéristiques et tarifs.
- Sélectionner le type de chambre et définir les dates de séjour.
- Créer, modifier ou annuler une réservation conformément aux disponibilités.
- Effectuer le check-in et le check-out en ligne, si cette fonctionnalité est activée.
- Effectuer les paiements en ligne de manière sécurisée.
- Recevoir des notifications concernant les réservations, les paiements ou les services additionnels.
- Consulter et télécharger les factures liées à ses séjours.
- Accéder à l'historique complet de ses réservations pour un suivi personnalisé.

## **3. Conception et Méthodologie de Développement**

### **3.1. Conception UML :**

La modélisation du système ROYELLA SHOTEL a été réalisée à l'aide de la notation UML afin d'assurer une conception claire, structurée et professionnelle. Cette modélisation permet de représenter les entités principales du système, leurs relations ainsi que les interactions entre les utilisateurs et les différents composants du système.

Le design UML comprend notamment :

- Diagrammes de cas d'utilisation (Use Case Diagrams) : pour illustrer les fonctionnalités principales du système et les interactions entre les utilisateurs (internes et externes) et le système.
- Diagrammes de classes (Class Diagrams) : pour représenter la structure interne du système, les attributs et les relations entre les différentes entités, facilitant la compréhension du modèle conceptuel.

Cette approche méthodologique assure la clarté du design, facilite le développement et la maintenance, tout en garantissant la scalabilité et l'adaptabilité du système aux besoins évolutifs des établissements hôteliers modernes.

### 3.1.1. Diagramme de classe :

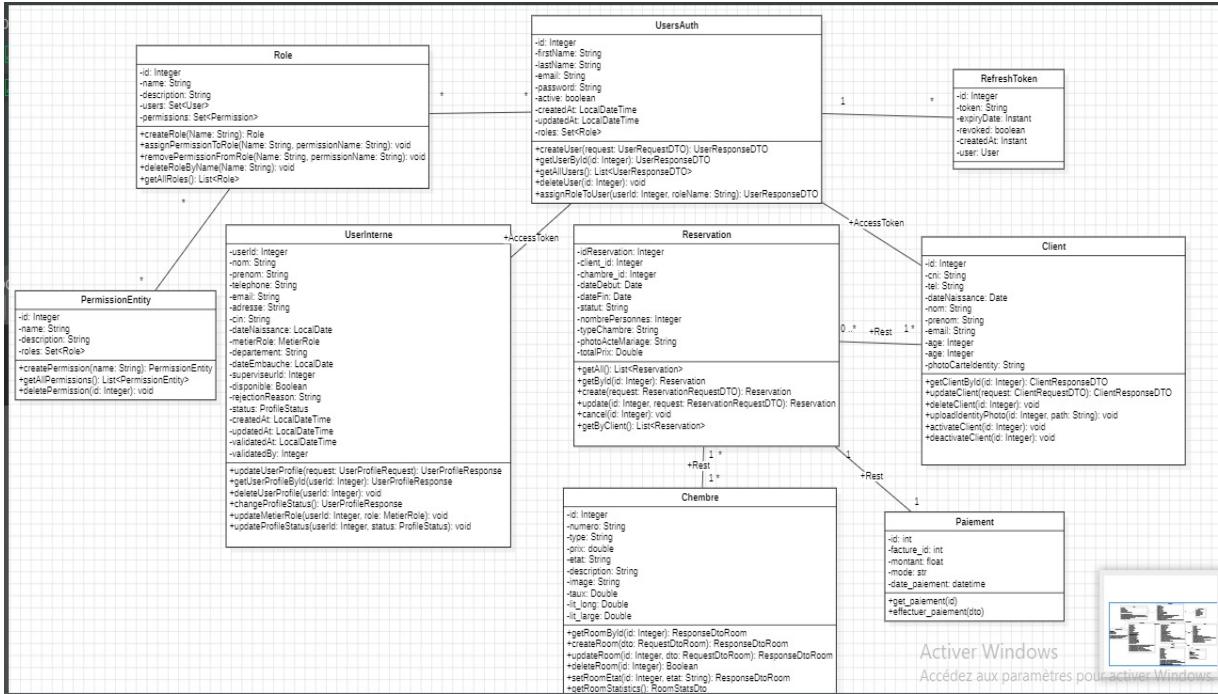


Figure 2 :Diagramme de classe

### 3.1.2. Diagramme de Cas d'utilisation :

#### Cas d'utilisation d'Administrateur

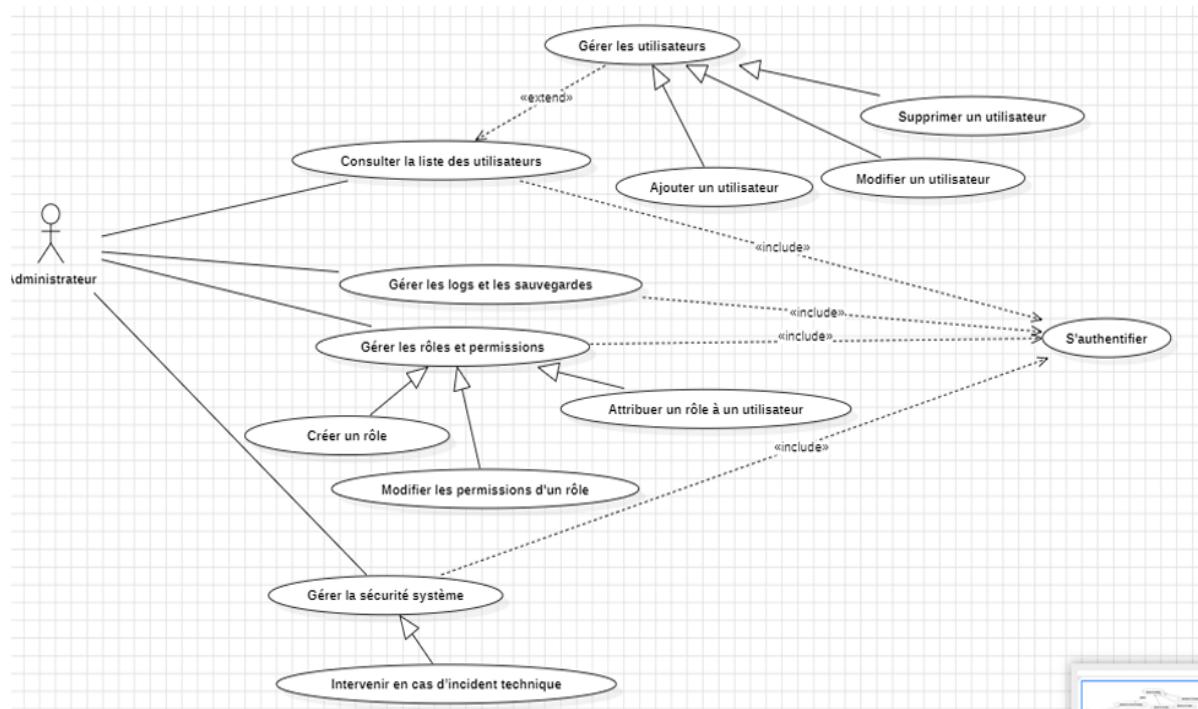


Figure 3 : Diagramme de cas d'utilisation d'administrateur

## Cas d'utilisation client

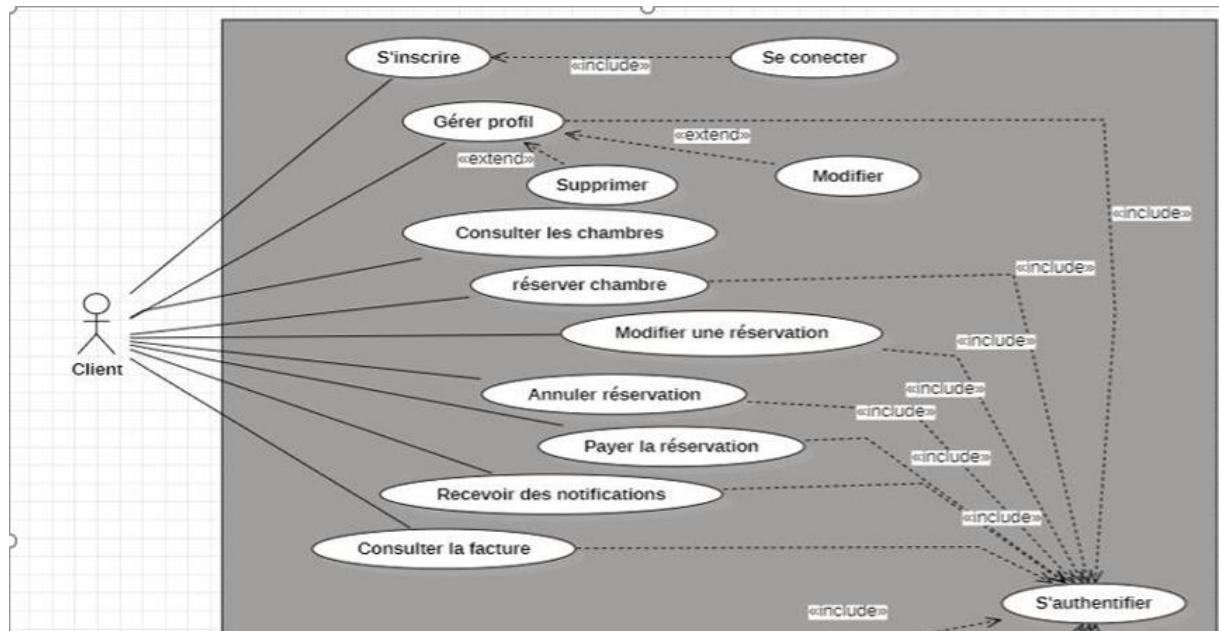


Figure 4 : Diagramme de cas d'utilisation Client

## Cas d'utilisation Housekeeping

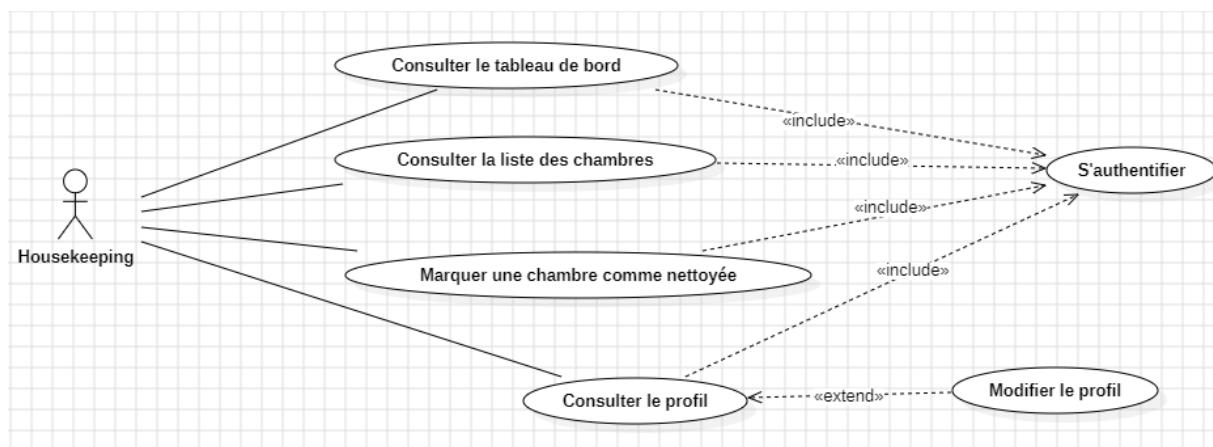


Figure 5 : Diagramme de cas d'utilisation Housekeeping

## Cas d'utilisation Réceptionniste

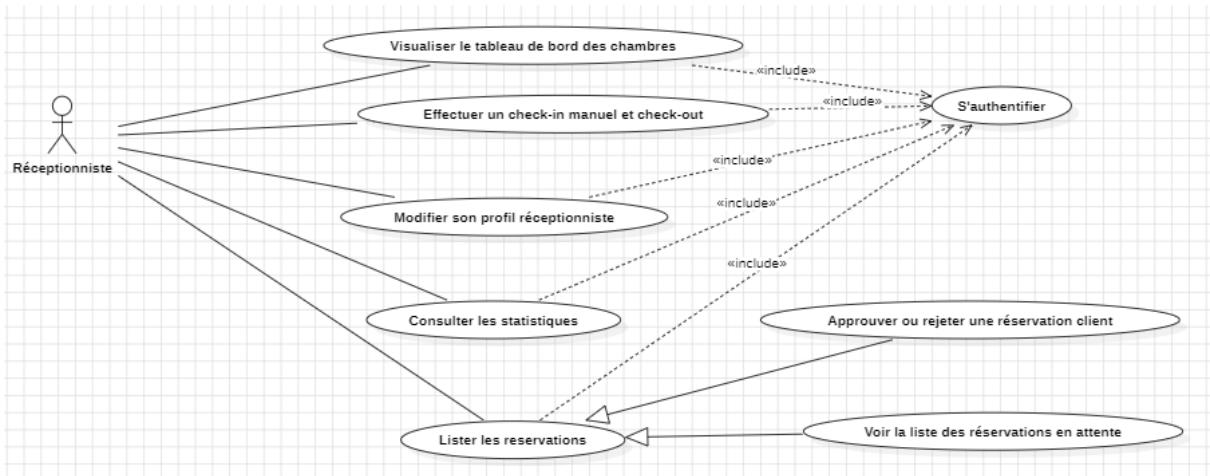


Figure 6 : Diagramme de cas d'utilisation Réceptionniste

## Cas d'utilisation Manager

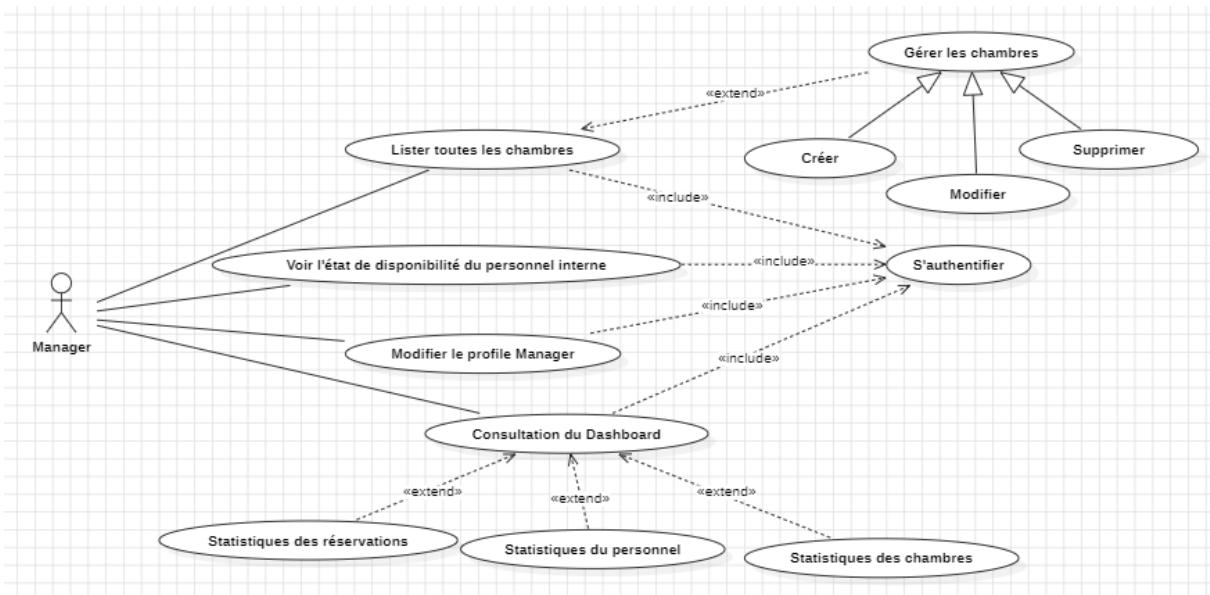


Figure 7 : Diagramme de cas d'utilisation Manager

## Cas d'utilisation Maintenance

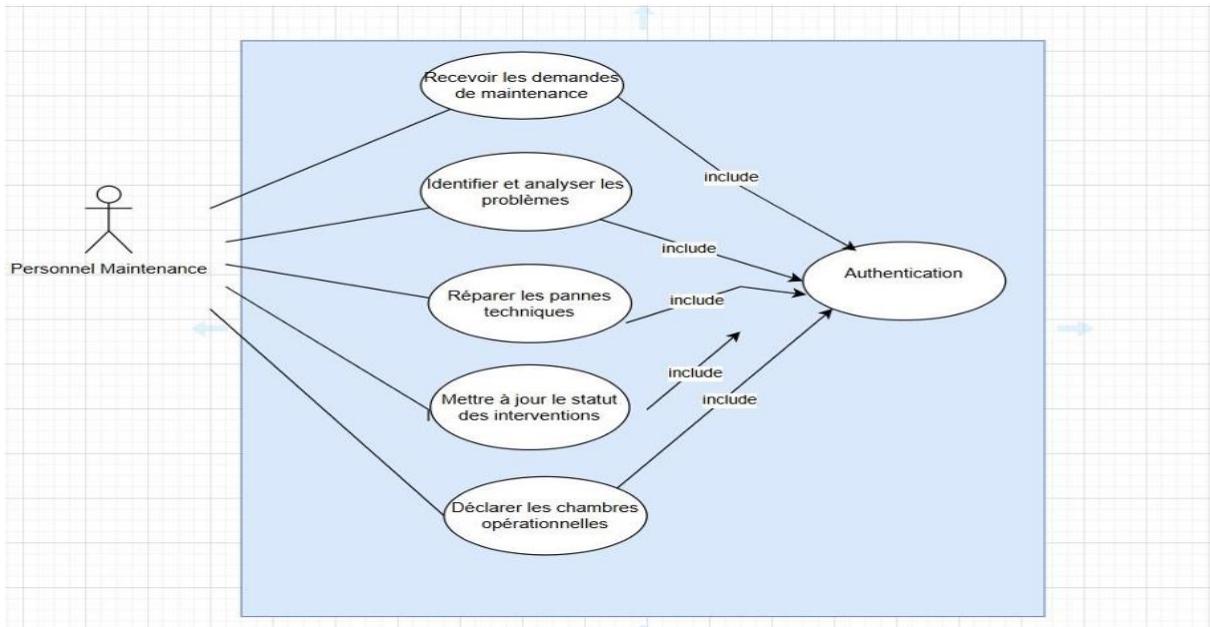


Figure 8 : Diagramme de cas d'utilisation Personnel Maintenance

## ✚ Cas d'utilisation Comptable

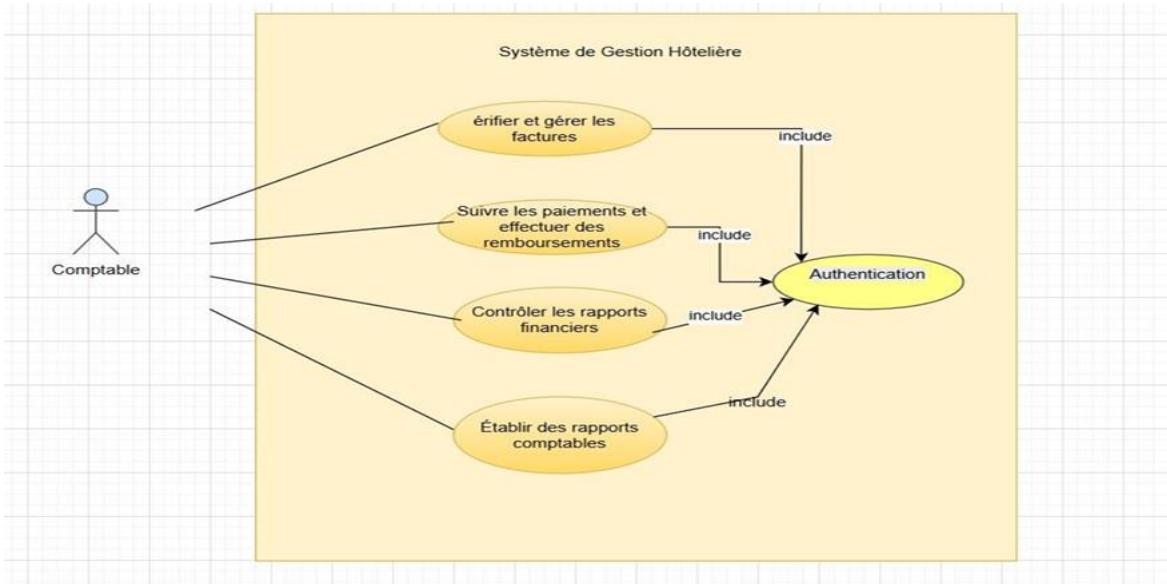


Figure 9 : Diagramme de cas d'utilisation Comptable

### 3.2. Application du Domain-Driven Design (DDD) :

Le projet adopte la méthodologie Domain-Driven Design (DDD) comme approche principale pour la conception de l'architecture basée sur les microservices.

Cette approche permet d'identifier et de structurer les domaines métiers du système, de les organiser en bounded contexts indépendants, et de concevoir des microservices autonomes présentant un faible couplage et un fort degré de cohésion.

Chaque microservice est responsable de son propre domaine métier, applique ses règles métiers spécifiques, et communique avec les autres services via des interfaces clairement définies,

assurant ainsi la scalabilité, la maintenabilité et l'évolutivité du système de manière efficace et durable.

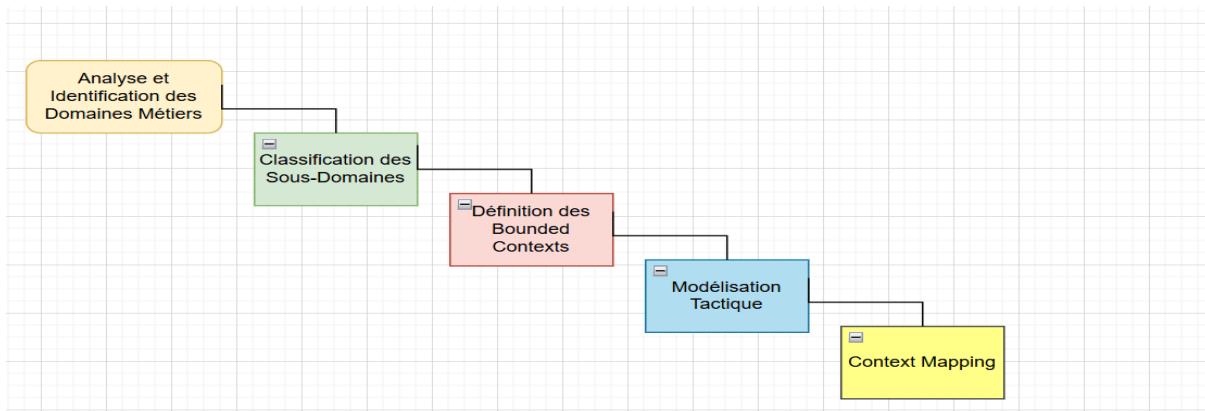


Figure 10 : Processus d'application de la méthodologie DDD

### 3.2.1. Analyse et Identification des Domaines Métiers

Une analyse approfondie du système a été menée à partir d'une perspective métier afin d'identifier les domaines fonctionnels essentiels au bon fonctionnement de l'établissement hôtelier. Cette démarche a permis de structurer le système autour de plusieurs domaines fondamentaux :

- **Domaine Clients** : gestion des informations personnelles, des comptes clients et de l'historique des séjours.
- **Domaine Réservations** : gestion du cycle de vie des réservations (création, modification, annulation) ainsi que la vérification de la disponibilité des chambres.
- **Domaine Chambres** : gestion des chambres, de leurs catégories, de leurs états (libre, occupée, en nettoyage ou en maintenance) et de leur tarification.
- **Domaine Facturation et Paiements** : génération des factures, traitement des paiements et remboursements, suivi financier.
- **Domaine Services Internes** : gestion des activités du personnel de réception, de housekeeping et de maintenance.
- **Domaine Administration et Sécurité** : gestion des utilisateurs, des rôles, des permissions et des mécanismes de sécurité.

Cette structuration constitue la base conceptuelle de l'architecture du système et garantit une meilleure compréhension du métier ainsi qu'une forte cohérence fonctionnelle.

### 3.2.2. Classification des sous-domaines

Conformément aux principes du Domain-Driven Design, les domaines identifiés ont été classés en trois catégories :

#### Core Domain

- Gestion des réservations

- Gestion des chambres

Ces domaines représentent le cœur métier du système, car ils ont un impact direct sur l'activité hôtelière et la satisfaction des clients.

### **Supporting Domains**

- Gestion des clients
- Facturation et paiements
- Services internes (réception, housekeeping, maintenance)

Ils soutiennent les domaines centraux et assurent la continuité et l'efficacité des processus opérationnels.

### **Generic Domains**

- Administration du système
- Authentification et sécurité

Ces domaines sont nécessaires au fonctionnement global du système, sans constituer un avantage métier spécifique.

#### **3.2.3. Définition des Bounded Contexts**

Chaque domaine a été encapsulé dans un **Bounded Context**, disposant de son propre modèle métier, de ses règles de gestion et de ses responsabilités spécifiques :

- **Contexte Gestion des Clients** : création et gestion des comptes clients, suivi des informations personnelles et de l'historique des réservations.
- **Contexte Gestion des Réservations** : traitement des demandes de réservation, modification, annulation et validation de la disponibilité des chambres.
- **Contexte Gestion des Chambres** : administration des chambres, gestion de leurs états et de leur affectation aux réservations.
- **Contexte Facturation et Paiements** : génération automatique des factures, enregistrement des paiements et suivi financier.
- **Contexte Services Internes** : coordination des activités du personnel de réception, housekeeping et maintenance.
- **Contexte Administration et Sécurité** : gestion des utilisateurs, des rôles, des permissions et supervision de la sécurité du système.

Cette organisation favorise l'indépendance des composants et réduit le couplage entre les différents modules applicatifs.

#### **3.2.4. Modélisation tactique**

Au sein de chaque Bounded Context, une modélisation tactique a été réalisée afin d'identifier les principaux éléments structurants du système :

- **Entités** : Client, Réservation, Chambre, Facture, Utilisateur, Intervention.

- **Services métiers** : gestion des réservations, attribution des chambres, génération des factures, gestion des interventions techniques.
- **Règles métier principales :**
  - Une réservation ne peut être validée que si une chambre est disponible.
  - Une facture est automatiquement générée après un check-out.
  - Une chambre ne peut être déclarée occupée sans réservation active.
  - L'accès aux fonctionnalités du système est contrôlé selon les rôles et permissions des utilisateurs.

Ces règles assurent la cohérence fonctionnelle du système et traduisent fidèlement les contraintes métier de l'environnement hôtelier.

### 3.2.5. Context Mapping

La phase de **Context Mapping** a permis de formaliser les interactions entre les différents Bounded Contexts :

- Le contexte Réservations interagit avec le contexte Chambres pour vérifier la disponibilité avant validation.
- Le contexte Réservations déclenche le contexte Facturation lors du check-out afin de générer la facture correspondante.
- Le contexte Administration et Sécurité fournit les mécanismes d'authentification et d'autorisation à l'ensemble des autres contextes.
- Les contextes Housekeeping et Maintenance communiquent avec le contexte Chambres pour mettre à jour l'état opérationnel des chambres.

Cette cartographie favorise la mise en place d'une architecture modulaire, évolutive et alignée avec une approche microservices, garantissant ainsi la maintenabilité, la scalabilité et la robustesse du système.

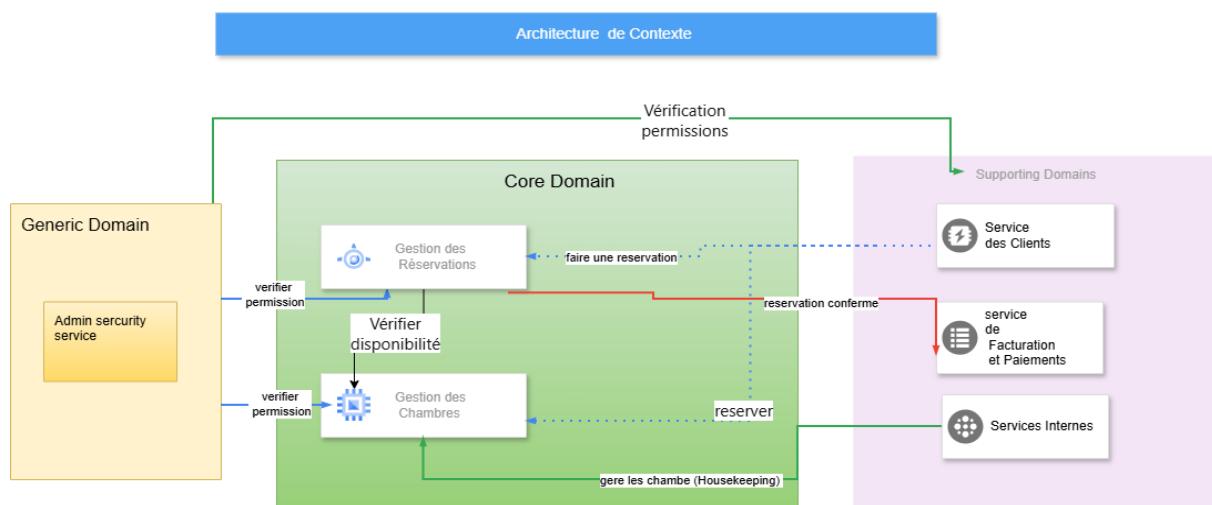


Figure 11 : Cartographie des Contextes Métiers selon DDD

# Chapitre 4 : Implémentation, intégration et validation du système

## 1. Architecture Technique :

L'architecture technique de ce système repose sur une approche modulaire et distribuée, centrée sur le modèle microservices, qui constitue une solution moderne pour le développement d'applications complexes et évolutives. Ce choix architectural a été motivé par la nécessité de garantir l'indépendance fonctionnelle des composants, une meilleure maintenabilité, ainsi qu'une scalabilité horizontale adaptée à l'augmentation du volume des utilisateurs et des transactions.

Chaque domaine fonctionnel critique du système, tel que la gestion des réservations, la facturation, la gestion des chambres, ou la coordination des services internes, est implémenté sous forme de microservice autonome. Chaque microservice encapsule ses propres entités métier, règles de gestion et processus, tout en interagissant avec les autres services via des interfaces clairement définies (APIs REST ou événements asynchrones).

Cette architecture permet une évolution indépendante de chaque service, réduisant ainsi le risque d'impacts négatifs lors de modifications ou de déploiements. Par ailleurs, elle facilite la répartition de la charge, l'optimisation des performances et la flexibilité dans le choix des technologies pour chaque microservice.

Enfin, cette conception technique s'aligne parfaitement avec la méthodologie Domain-Driven Design (DDD) adoptée pour le projet, garantissant que chaque microservice reflète fidèlement le domaine métier correspondant, tout en favorisant la cohérence fonctionnelle, la robustesse du système et une mise en production progressive et sécurisée.

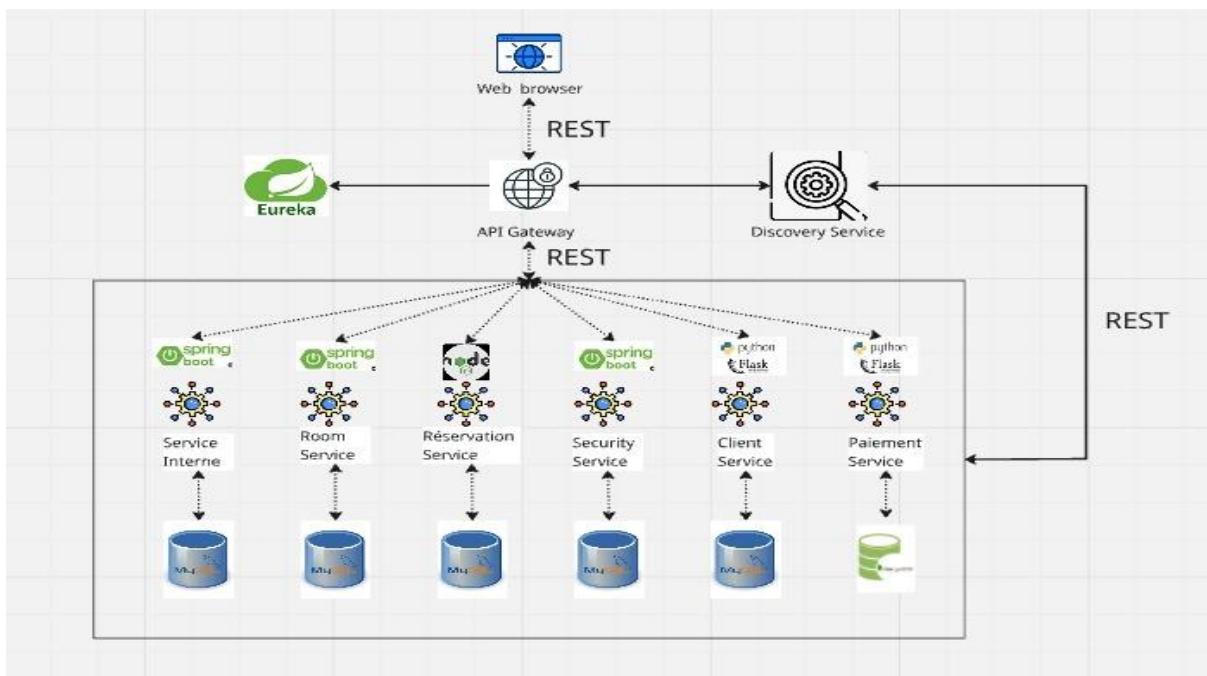


Figure 12 : Architecture Microservices avec API Gateway et Découverte de Services

## 1.1 Développement Back-end :

### a. Mise en place de l'environnement de développement :

Afin d'assurer une configuration cohérente, optimale et standardisée durant le processus de développement des microservices, un ensemble d'outils et de technologies a été rigoureusement sélectionné et mis en œuvre.

#### ⊕ IDE :



L'environnement de développement IntelliJ IDEA a été employé pour la conception et l'implémentation des microservices backend, en tirant parti des frameworks Java (Spring Boot) et Python (Flask) afin d'assurer un développement structuré, modulable et conforme aux bonnes pratiques de l'ingénierie logicielle.



Visual Studio Code a été employé pour le développement de la partie frontale (Frontend) du système, en utilisant le framework Angular. Cet outil fournit un environnement de programmation léger et performant, favorisant une écriture structurée du code, tout en améliorant la productivité et en garantissant la maintenabilité et l'évolutivité de

l'application.

#### ⊕ Gestion des dépendances :



Maven a été adopté comme outil de gestion des bibliothèques et des dépendances au sein des projets développés avec Spring Boot. Il permet une intégration simplifiée et standardisée des frameworks et bibliothèques nécessaires, garantissant ainsi la cohérence, la reproductibilité des builds et la maintenabilité du projet.



Pip a été utilisé pour la gestion des dépendances dans les microservices développés avec Flask. Cet outil assure l'installation, la mise à jour et la maintenance des bibliothèques Python nécessaires, garantissant ainsi un environnement cohérent et reproductible pour le développement et le déploiement des services



Axios est une bibliothèque JavaScript open source basée sur les promesses, utilisée pour effectuer des requêtes HTTP asynchrones entre une application cliente (front-end ou back-end) et des services web (API REST). Elle facilite la communication avec les serveurs en prenant en charge automatiquement la sérialisation des données en JSON, la gestion des erreurs, ainsi que la configuration des en-têtes HTTP, ce qui améliore la lisibilité, la maintenabilité et la fiabilité du code applicatif.

#### ⊕ Serveurs locaux :



#### Apache Tomcat

Tomcat a été utilisé comme serveur local afin de déployer, exécuter et tester les microservices développés en Java, permettant ainsi de vérifier leur bon fonctionnement dans un environnement contrôlé avant leur intégration au système global.



#### Flask

Flask a été employé comme serveur local pour déployer et tester les API développées en Python, permettant de valider leur fonctionnement et leur conformité avant leur intégration dans l'architecture globale des microservices.

#### ⊕ Systèmes de gestion de bases de données locaux :



XAMPP a été utilisé comme environnement local pour l'administration et la gestion des bases de données MySQL, permettant le test, la validation et l'évolution des schémas relationnels.



MongoDB Compass a servi d'outil graphique pour l'exploration des collections MongoDB, la visualisation des documents et la vérification de l'intégrité des données NoSQL.

#### b. Spécifications techniques des microservices

##### ⊕ Java - Spring Boot :

###### ❖ Service de Découverte :

- Gestion centralisée de l'enregistrement des microservices dans le registre Eureka.
- Permet aux microservices de se localiser mutuellement dynamiquement sans connaître les adresses IP ou ports fixes.
- Facilite la communication et l'interaction entre les services distribués.
- Améliore la résilience et la scalabilité de l'architecture microservices.
- Supporte l'intégration avec d'autres services comme le Configuration Service pour une configuration dynamique et cohérente.

###### ❖ Service Passerelle (Gateway Service)

- Point d'entrée unique pour toutes les requêtes client vers les microservices.
- Routage dynamique des requêtes vers les services appropriés selon l'URL ou les règles définies.
- Gestion de la sécurité : authentification et autorisation centralisées, souvent intégrée avec OAuth2 et JWT.
- Gestion de la limitation du trafic (rate limiting), du caching et du logging des requêtes.
- Facilite l'intégration et la communication avec les autres services comme le Discovery Service et le Configuration Service.

#### ❖ Service de Sécurité

- Gestion centralisée de l'authentification et de l'autorisation des utilisateurs.
- Intégration avec OAuth2 et JWT (JSON Web Token) pour sécuriser les communications entre clients et microservices.
- Gestion des rôles et permissions pour contrôler l'accès aux différentes ressources et services.
- Protection contre les accès non autorisés et suivi des activités pour améliorer la sécurité globale du système.
- Interaction avec le Gateway Service pour appliquer la sécurité au niveau des points d'entrée.

#### ❖ Service des Chambres (Room Service)

- Gestion des informations des chambres, incluant l'ajout, la modification, la suppression et la consultation selon différents critères comme le numéro de chambre, le prix ou le type.
- Suivi de l'état des chambres (disponible, réservée, en maintenance) avec possibilité de modification par les utilisateurs autorisés (Manager, Réceptionniste).
- Intégration avec le système de rôles et permissions pour contrôler l'accès aux différentes opérations.
- Prise en charge du téléversement et stockage des images des chambres, avec accès via des chemins définis.
- Fourniture de statistiques sur les chambres (nombre de chambres par type et état) pour aider à la prise de décision.
- Interaction avec le Configuration Service et le Discovery Service pour assurer la communication dynamique et la cohérence des configurations entre services.

#### ❖ Service Interne (User Profile Service)

- Gestion des profils des utilisateurs internes, incluant la création, la modification, la suppression et la consultation des informations personnelles et professionnelles.
- Support de la gestion des rôles et permissions pour contrôler l'accès aux fonctionnalités selon le rôle (Admin, Manager, Housekeeping, Réceptionniste, Maintenance, Comptable).
- Suivi et gestion des statuts des profils (Draft, On Work, Out Work, Validated, Rejected) avec mise à jour automatique selon l'activité ou intervention de l'administrateur.
- Sécurisation des accès via JWT et OAuth2, garantissant que les opérations sont effectuées uniquement par les utilisateurs autorisés.
- Intégration avec le Configuration Service et le Discovery Service pour assurer la communication dynamique et la cohérence des configurations.

#### ❖ Python (Flask)

#### ❖ Service Client (Client Service)

- Gestion des informations des clients, incluant la création de nouveaux comptes, la mise à jour des données personnelles et la récupération des informations clients par identifiant ou compte connecté.

- Support de la gestion des rôles et permissions pour contrôler l'accès aux différentes opérations (Admin, Client, Réceptionniste).
- Intégration avec la base de données MySQL pour stocker et gérer durablement les données des clients.

#### ❖ Service de Paiement (Payment Service)

- Gestion des transactions de paiement (enregistrement, mise à jour du statut) via MongoDB comme base de données NoSQL.
- Vérification et validation des paiements pour garantir l'exactitude et la fiabilité des transactions.
- Intégration avec les autres services (Client, Réservation) pour assurer le lien entre paiement et réservation.

#### Node.js

#### ❖ Service de Réservation

- Gestion des réservations des clients (création, mise à jour, suppression, annulation).
- Agrégation des informations liées à la réservation via communication avec les services Room et Client.
- Calcul automatique du prix total de la réservation en fonction du type de chambre, de la durée et du nombre de personnes.
- Vérification des conditions de réservation (dates valides, nombre de personnes limité).
- Gestion des statuts de réservation (pending, validée, annulée) et des droits d'accès selon le rôle de l'utilisateur (CLIENT, ADMIN, RECEPTIONNISTE).
- Sécurisation des endpoints via JWT (JSON Web Token) pour l'authentification et l'autorisation des utilisateurs.

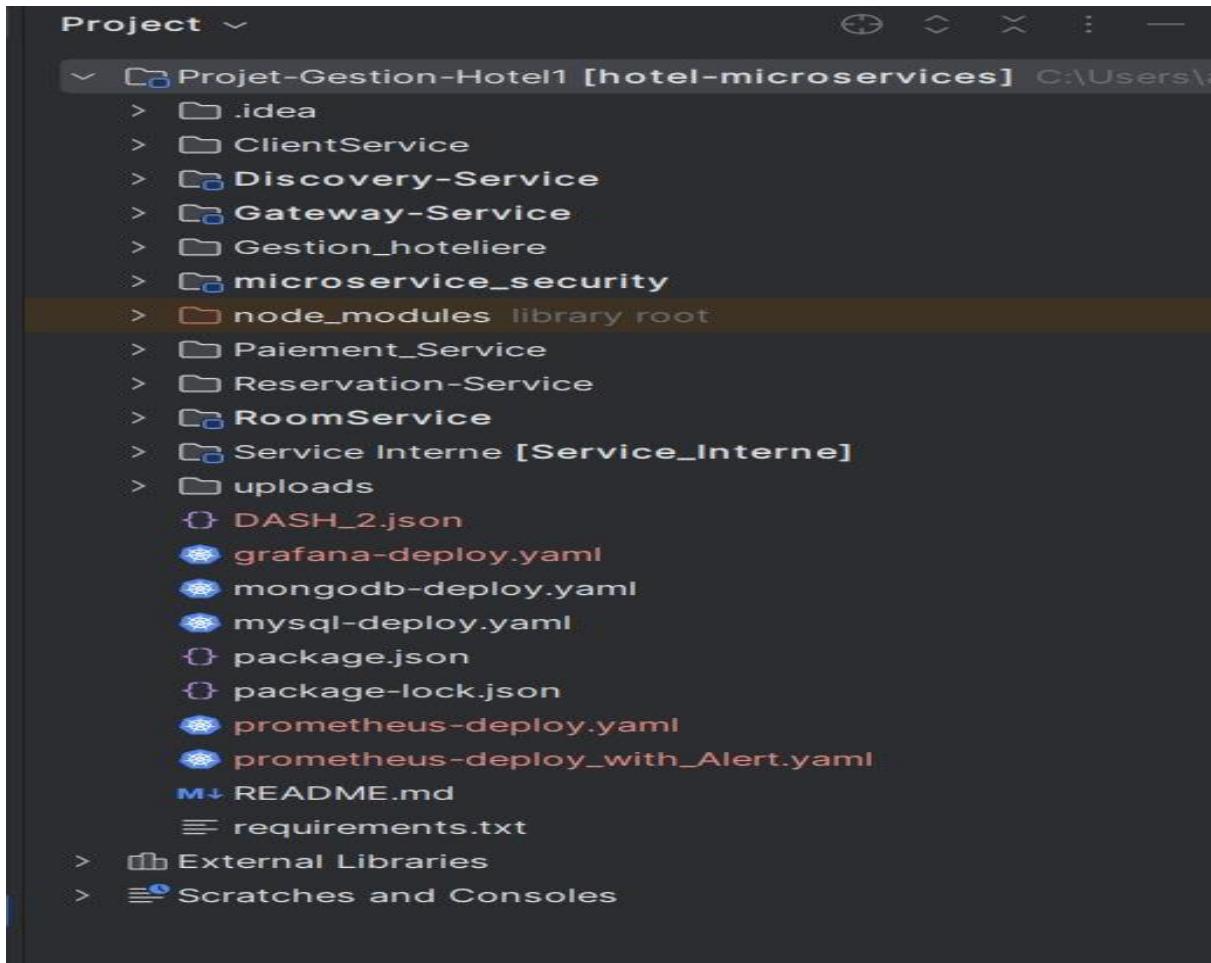


Figure 13 : Structure du Projet BackEnd (microservices)

## 1.2 Développement Front-end :

### Angular :

Mise en place de deux interfaces utilisateur distinctes afin de répondre aux besoins spécifiques de chaque catégorie d'utilisateurs :

- Une interface destinée aux clients pour faciliter la consultation des services, la réservation et la gestion de leur compte.
- Une interface interne réservée au personnel de l'hôtel (Administrateurs, Managers, Housekeeping), permettant la gestion opérationnelle et administrative des ressources et des services.

Ces interfaces ont été conçues pour offrir une expérience utilisateur fluide et moderne, avec une ergonomie optimisée.

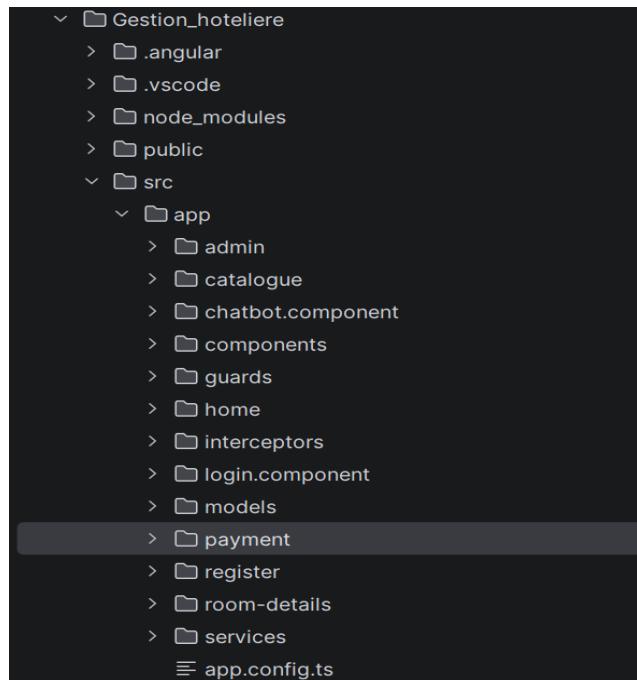


Figure 14 : Structure du Projet Frontend Angular (Gestion\_hoteliere)

### ❖ **Interface Client :**

L’interface client a été conçue dans une optique de convivialité et d’efficacité, afin de fournir une expérience utilisateur fluide et intuitive. Elle permet notamment de :

- **Explorer les offres disponibles** : Présenter les chambres et services accessibles avec un niveau de détail complet incluant les tarifs, les équipements, et les visuels illustratifs.
- **Gestion des réservations** : Permettre la création et le suivi des réservations de manière simple et sécurisée, optimisant ainsi la planification pour le client.
- **Consultation des factures** : Fournir un accès clair aux factures générées, avec la possibilité de les visualiser et de les télécharger en toute sécurité.

### ❖ **Interface d’Administration**

L’interface d’administration a été conçue pour fournir une gestion complète et efficace de toutes les opérations administratives de l’hôtel, en mettant l’accent sur le contrôle des utilisateurs et des ressources essentielles. Les principales fonctionnalités comprennent :

- **Gestion des utilisateurs** : Création, mise à jour et suppression des comptes utilisateurs, avec l’attribution des rôles et permissions appropriés afin d’assurer un contrôle précis de l’accès au système.
- **Gestion des rôles et permissions** : Définition et administration des différents rôles (par exemple : Manager, Réceptionniste, Housekeeping, Comptable), avec l’association des permissions adéquates à chaque rôle pour garantir l’organisation fonctionnelle et le respect des politiques de sécurité internes.

- **Suivi des réservations** : Consultation et suivi de toutes les réservations effectuées par les clients, avec possibilité de mettre à jour leur statut afin de faciliter la planification quotidienne des chambres et d'assurer la qualité du service.
- **Suivi de l'état des chambres** : Surveillance de l'état des chambres (disponible, réservée, en maintenance) pour garantir une gestion efficace des ressources disponibles.
- **Tableau de bord interactif** : Présentation d'indicateurs clés tels que le nombre de réservations, l'état des chambres et les utilisateurs actifs, afin de soutenir la prise de décision administrative basée sur des données fiables.

### ❖ Interface Manager

L'interface Manager constitue une plateforme centralisée dédiée à la supervision globale des opérations hôtelières. Elle permet au responsable de piloter efficacement les ressources humaines, les chambres et les performances générales de l'établissement à travers des outils de suivi et de visualisation avancés.

Elle offre les fonctionnalités suivantes :

- **Gestion du profil utilisateur** : Consultation et mise à jour des informations personnelles du manager dans un environnement sécurisé.
- **Tableau de bord décisionnel** : Visualisation d'indicateurs clés de performance (KPI) tels que le nombre de réservations, les revenus, les chambres disponibles et les services internes, accompagnés de graphiques analytiques pour faciliter la prise de décision.
- **Gestion des chambres** : Création, modification, suppression et consultation des chambres avec leurs caractéristiques (type, prix, état, équipements, images et dimensions), permettant une gestion optimale de la capacité hôtelière.

Cette interface permet ainsi au manager d'assurer une gestion stratégique, fluide et centralisée de l'ensemble des activités de l'hôtel.

### ❖ Interface du Housekeeping

L'interface dédiée au personnel de housekeeping a été conçue afin d'optimiser la gestion opérationnelle des tâches de nettoyage et de maintenance des chambres, tout en assurant un haut niveau de qualité de service et une meilleure coordination entre les équipes internes de l'hôtel.

Elle offre notamment les fonctionnalités suivantes :

- **Tableau de bord interactif (Dashboard)** : Visualisation des indicateurs clés tels que le nombre total de chambres, les chambres à nettoyer, les tâches assignées et celles finalisées, accompagnées de graphiques illustrant l'état global des chambres.
- **Gestion des chambres** : Consultation et filtrage des chambres selon leur statut (disponible, occupée, en maintenance), avec possibilité de déclencher des tâches de nettoyage.
- **Gestion des tâches** : Suivi des missions attribuées au personnel, avec mise à jour de leur progression (en attente, en cours, terminée).

- **Gestion du profil utilisateur** : Modification sécurisée des informations personnelles.
- **Système de notifications** : Réception d'alertes en temps réel concernant les nouvelles tâches et les changements d'état des chambres.

Cette interface contribue à améliorer l'efficacité opérationnelle, la réactivité du personnel et la qualité globale des services hôteliers proposés.

#### ❖ **Interface Réceptionniste**

Une interface dédiée a été conçue pour le personnel de réception afin de soutenir efficacement la gestion quotidienne des opérations hôtelières. Cette interface propose une expérience utilisateur intuitive et sécurisée, reposant sur une architecture moderne et une présentation ergonomique des données.

Elle offre notamment les fonctionnalités suivantes :

- **Gestion du profil utilisateur** : Consultation et mise à jour sécurisée des informations personnelles du réceptionniste.
- **Suivi des chambres** : Visualisation en temps réel de l'ensemble des chambres avec leurs caractéristiques (numéro, type, tarif, état : disponible, occupée, en maintenance), accompagnée d'indicateurs visuels facilitant l'analyse opérationnelle.
- **Gestion des réservations** : Consultation, validation, modification et suivi des réservations, avec accès aux informations clients associées et à l'historique des séjours.

Cette interface contribue à l'optimisation des processus opérationnels, à la réduction des délais de traitement des réservations et à l'amélioration globale de la qualité du service rendu aux clients.

### **1.3 Bases de Données :**

#### ➤ **Base de données relationnelle (MySQL) :**

MySQL a été utilisée pour gérer les informations relatives aux clients, aux réservations et aux chambres. Cette base relationnelle permet une organisation structurée des données et l'établissement de relations complexes entre les entités, garantissant ainsi l'intégrité et la cohérence des informations. MySQL facilite également l'exécution de requêtes complexes et l'analyse précise des données, ce qui en fait un choix idéal pour la gestion opérationnelle quotidienne de l'hôtel.

#### ➤ **Base de données NoSQL (MongoDB) :**

MongoDB a été adoptée pour stocker les données non structurées ou semi-structurées, telles que la gestion du service de paiement (Payment Service). MongoDB se distingue par sa grande flexibilité et sa capacité à évoluer horizontalement (Horizontal Scaling), ce qui permet de traiter de grands volumes de données dynamiques sans modifier la structure de la base. Son schéma dynamique (Dynamic Schema) est particulièrement adapté aux données dont la structure peut varier au fil du temps, ce qui est crucial pour les services de paiement nécessitant un traitement rapide et des mises à jour constantes des transactions financières.

## **Avantages de la combinaison MySQL et MongoDB dans le projet :**

Intégrité et flexibilité des données : MySQL est utilisée pour les données structurées et sensibles comme les réservations, tandis que MongoDB gère les données semi-structurées ou dynamiques.

Performance et adaptabilité : MongoDB offre une flexibilité élevée pour les données modernes et variées, alors que MySQL garantit la stabilité des données critiques.

Scalabilité : Cette combinaison permet une gestion efficace des opérations quotidiennes tout en assurant une évolutivité future pour le traitement des données de paiement et autres informations dynamiques, sans compromettre l'ensemble du système.

## **1.4. Communication entre microservices et interface utilisateur (Front-end) :**

### **REST API :**

Dans le cadre de notre architecture microservices, la communication entre le backend et le frontend repose sur des APIs RESTful, permettant un échange standardisé et performant des données entre les différents services et l'interface utilisateur. Cette approche garantit :

- Une interaction structurée entre microservices, où chaque service expose ses fonctionnalités via des endpoints clairement définis.
- Un accès sécurisé et contrôlé depuis le frontend, permettant aux clients et au personnel de l'hôtel d'interagir avec le système de manière fluide et fiable.

### **CORS (Cross-Origin Resource Sharing)**

Afin de permettre aux applications front-end, hébergées sur des domaines différents, d'accéder aux services backend, le mécanisme CORS (Cross-Origin Resource Sharing) a été mis en place. CORS permet :

- De contrôler l'accès aux ressources en spécifiant quels domaines externes sont autorisés à effectuer des requêtes vers le serveur.
- D'assurer la sécurité des échanges entre le frontend et le backend, en évitant les accès non autorisés tout en maintenant la compatibilité avec les navigateurs modernes.
- D'optimiser l'interopérabilité entre différentes applications web et services, en respectant les standards de communication sécurisés.

Cette combinaison RESTful + CORS offre une solution robuste, extensible et sécurisée pour la communication interservices et avec l'interface utilisateur, tout en garantissant la maintenabilité et la scalabilité de l'architecture du système.

## **1.5 Architecture des Services :**

### **API Gateway :**

La passerelle API assure un point d'accès centralisé à l'ensemble des microservices et agrège les réponses provenant de chaque service. Elle permet d'optimiser le traitement des requêtes en renvoyant au frontend que les données strictement nécessaires, contribuant ainsi à améliorer les performances et l'efficacité du système.

### **Microservices (REST API) :**

Chaque microservice fonctionne de manière autonome, avec un endpoint RESTful propre, garantissant une indépendance fonctionnelle et facilitant la maintenance, l'évolution et l'extensibilité du système. Les services principaux incluent :

- Service Client : gestion des informations clients et de leurs interactions avec le système.
- Service Interne : gestion des utilisateurs internes, de leurs rôles et permissions.
- Service Chambre : gestion des données et de l'état des chambres.
- Service Réservation : gestion complète des réservations et suivi des statuts.
- Service Paiement : traitement sécurisé des paiements et gestion des factures.

➤ **Service de Découverte (Eureka) :**

L'utilisation de Eureka permet la découverte dynamique des microservices, facilitant l'équilibrage de charge et permettant à chaque service de localiser et interagir avec les autres services automatiquement, sans nécessiter de configuration statique préalable. Cette approche renforce la flexibilité, la scalabilité et la robustesse de l'architecture microservices.

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
CLIENT-SERVICE	n/a (1)	(1)	UP (1) - <a href="http://10.108.151.91:client-service:8088">10.108.151.91:client-service:8088</a>
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - <a href="http://gateway-service-8667b5dd55-4m6gn:Gateway-Service:8888">gateway-service-8667b5dd55-4m6gn:Gateway-Service:8888</a>
MICROSERVICE-SECURITY	n/a (1)	(1)	UP (1) - <a href="http://microservice-security-c85bb89c6-2wvhs:microservice-security:8070">microservice-security-c85bb89c6-2wvhs:microservice-security:8070</a>
PAIEMENT-SERVICE	n/a (1)	(1)	UP (1) - <a href="http://10.110.90.192:paientment-service:8090">10.110.90.192:paientment-service:8090</a>
RESERVATION-SERVICE	n/a (1)	(1)	UP (1) - <a href="http://reservation-service">reservation-service</a>
ROOMSERVICE	n/a (1)	(1)	UP (1) - <a href="http://roomservice-847bb468c6-kws89:RoomService:8093">roomservice-847bb468c6-kws89:RoomService:8093</a>
SERVICE_INTERNE	n/a (1)	(1)	UP (1) - <a href="http://service-interne-55bf64b74f-9jgqx:Service_Interne:8071">service-interne-55bf64b74f-9jgqx:Service_Interne:8071</a>

Figure 15 : Tableau des Services Inscrits dans Eureka

## 1.6 Documentation et Gestion des API :

➤ **Swagger**

- L'outil Swagger a été utilisé pour assurer la documentation complète et standardisée de l'ensemble des API exposées par les microservices. Cette approche permet une visualisation claire et interactive des différents endpoints, facilitant ainsi la compréhension et l'exploitation des services.
- L'intégration de Swagger UI fournit une interface web ergonomique et intuitive, permettant aux développeurs de tester les API en temps réel, d'optimiser le développement et d'assurer une maintenance simplifiée et sécurisée de l'architecture microservices.

The screenshot shows a Swagger UI interface for an 'Users' API. At the top, it says 'API pour la gestion des utilisateurs'. Below is a list of endpoints:

- POST /v1/users/{userId}/roles**
- POST /v1/users/register**
- POST /v1/users/refresh**
- POST /v1/users/logout**
- POST /v1/users/logout-all**
- POST /v1/users/login**
- POST /v1/users/create**
- PATCH /v1/users/{id}/status**
- GET /v1/users**
- GET /v1/users/{id}**
- DELETE /v1/users/{id}**

Figure 16 : exemple d'interface swagger pour la gestion des utilisateurs

## 1.7 Infrastructure et Déploiement :

### ✚ Conteneurisation et Orchestration :

- **Docker** pour la conteneurisation des microservices, permettant un déploiement uniforme et une isolation parfaite.

□	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
□	k8s_mongodb_mongodb_ 85ae8af0556e	4510cf3d7050			0.79%	2 hours ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_discovery-service_d 7d522f17212c	021719ea9401			7.09%	2 hours ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_service-interne_serv 750b7380884b	002f8a89fb83			6.29%	14 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_mysql_mysql-688b6 e096370ed1cd	9c3380eac945			2.02%	13 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_client-service_client 39fb38e54416	1d7a356ea09d			5.68%	13 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_reservation-service_ 54124ee2f0b0	25477b9a9002			1.87%	13 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_roomservice_rooms ebc2a1d88d0e	3c48c7b7346c			4.64%	13 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_microservice-securi 49cf82b3c85	e1bf65e54421			4.42%	13 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_paiement-service_p_ 32a43b3c1ad3	99988f32f75b			0.1%	12 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡
□	k8s_gestion-hoteliere_ge 7b7ae1272543	71a860111f58			0%	10 minutes ago	⋮ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ ⚡

Figure 17 : Tableau de Supervision des Conteneurs Docker en Production

- **Kubernetes** pour l'orchestration des conteneurs, garantissant la gestion des déploiements et la scalabilité des services

NAME	READY	STATUS	RESTARTS	AGE
client-service-695bdd9884-bsvkb	1/1	Running	0	171m
discovery-service-fc55897c7-lkwjq	1/1	Running	0	3h
gateway-service-8667b5dd55-dl7rx	1/1	Running	0	3h
gestion-hoteliere-78fbf4ff7b-g6sb2	1/1	Running	0	165m
grafana-service-87b8d87db-2dvcf5	1/1	Running	0	59m
microservice-security-c85bb89c6-gmpfh	1/1	Running	0	46m
mongodb-6897f97d59-pth5h	1/1	Running	0	3h1m
mysql-688b68b86f-crkbb	1/1	Running	0	3h1m
paiement-service-b99fc7fdd-j5vcz	1/1	Running	0	166m
prometheus-service-79c4b6d64-p442j	1/1	Running	0	83m
prometheus-service-867598f55b-bxkhv	0/1	ContainerCreating	0	4s
reservation-service-5fd9ff54f4-drv8j	1/1	Running	0	167m
roomservice-847bb468c6-x9mb2	1/1	Running	0	175m
service-interne-55bf64b74f-wzj2b	1/1	Running	0	173m

Figure 18 : État des Pods Kubernetes en Temps Réel

## ✚ CI/CD Pipeline :

### ➤ Intégration de l'Analyse de Qualité avec SonarQube

Afin de garantir un haut niveau de qualité, de sécurité et de maintenabilité du code source, notre pipeline CI/CD intègre une analyse statique automatique à l'aide de l'outil SonarQube.

Cette étape est exécutée systématiquement après les tests unitaires et avant la construction de l'image Docker, ce qui permet de détecter précolement les problèmes potentiels.

L'analyse SonarQube évalue chaque microservice selon plusieurs métriques essentielles liées à la qualité logicielle.

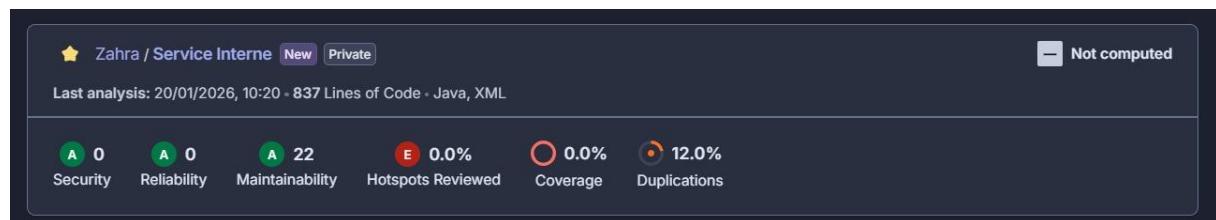


Figure 19 : Tableau de bord SonarQube – Analyse du Service Interne

Pour le service interne, les résultats obtenus sont les suivants :

- Sécurité (Security) : Grade A, aucune vulnérabilité détectée
- Fiabilité (Reliability) : Grade A, aucun bug potentiel identifié
- Maintenabilité (Maintainability) : Grade A, avec une dette technique estimée à 22 points
- Couverture de tests (Coverage) : 0.0 %, indiquant un besoin d'amélioration des tests unitaires
- Duplications de code : 12.0 %, signalant la présence de portions de code redondantes

Cette analyse porte sur 837 lignes de code, réparties entre fichiers Java et fichiers de configuration XML, offrant ainsi une vision globale et détaillée de la qualité du service analysé.

Concernant les tests unitaires, la couverture actuelle reste limitée en raison des contraintes de temps liées au développement et à l'intégration de plusieurs microservices hétérogènes. Toutefois, l'architecture du projet a été conçue pour faciliter l'ajout futur de tests unitaires et d'intégration, afin d'améliorer la qualité logicielle et la maintenabilité du système.

#### ➤ *Détails d'Exécution du Pipeline CI/CD*

Le pipeline CI/CD est orchestré à l'aide de Jenkins, en suivant une séquence d'étapes bien définie.

La Figure présente le détail des étapes du pipeline ainsi que le temps d'exécution associé à chacune d'elles.

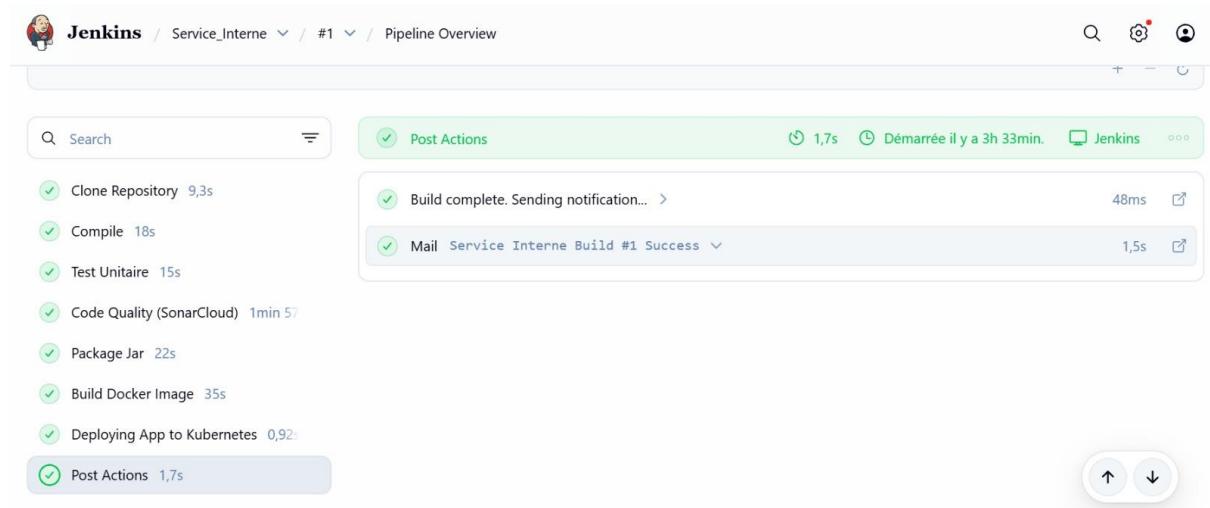


Figure 20 : Vue détaillée des étapes du pipeline CI/CD avec temps d'exécution

La chronologie d'exécution est la suivante :

- Clone Repository : 9,3 secondes
- Compilation du code : 18 secondes
- Tests unitaires : 15 secondes
- Analyse de qualité du code (SonarCloud) : 1 minute et 57 secondes
- Packaging du fichier JAR : 22 secondes
- Construction de l'image Docker : 35 secondes
- Déploiement sur Kubernetes : 0,92 seconde

La durée totale du pipeline est d'environ 3 minutes et 43 secondes, ce qui permet d'assurer des cycles de livraison rapides tout en conservant des contrôles de qualité stricts et automatisés.

#### ➤ *Visualisation Graphique du Pipeline*

La progression du pipeline est visualisée en temps réel via l'interface graphique de Jenkins, comme illustré dans ce Figure.

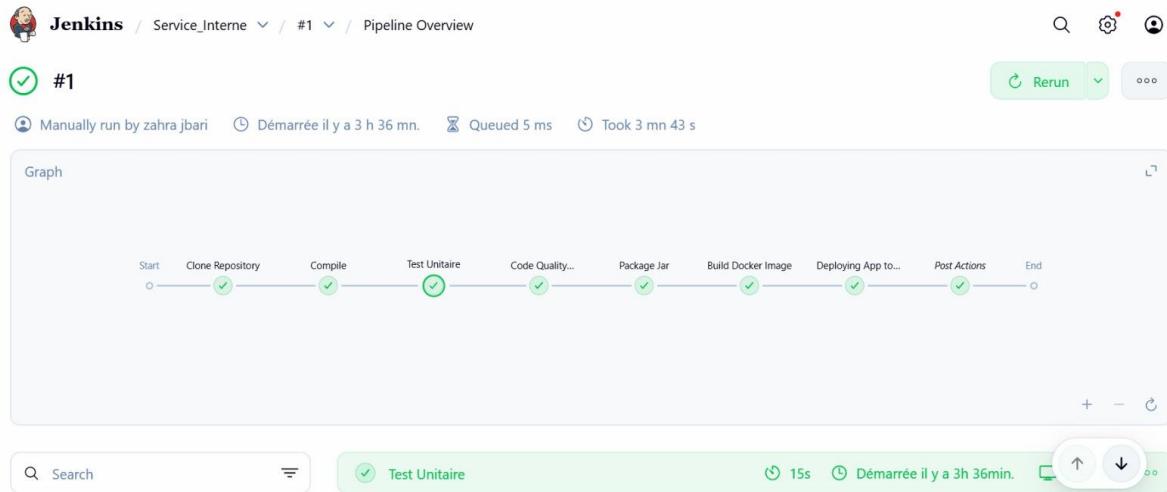


Figure 21 : Représentation graphique du pipeline CI/CD dans Jenkins

Le diagramme met en évidence l'enchaînement logique des différentes étapes

Chaque étape validée apparaît en vert, ce qui permet de suivre facilement l'état d'avancement du pipeline et d'identifier rapidement toute anomalie en cas d'échec.

#### ➤ Notifications et Retour d'Information

À la fin du pipeline, des notifications automatiques sont envoyées pour informer l'équipe du résultat

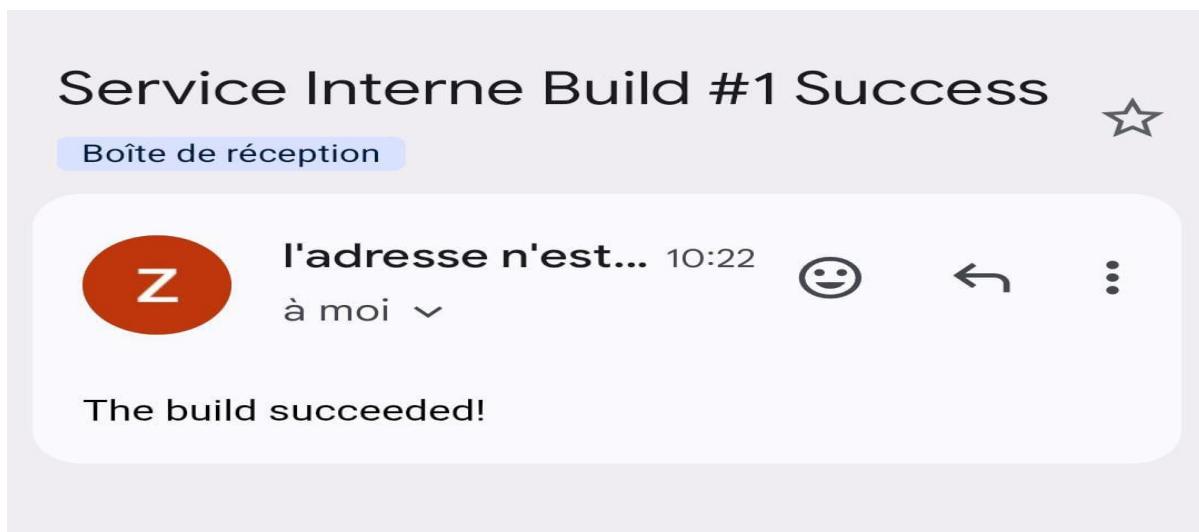


Figure 22 : notifications automatiques Build Success

#### ➤ Gestion du Versionnement (Git) :

- Utilisation de Git pour gérer le versionnement du code source et faciliter la collaboration entre les développeurs.

### 1.8 Monitoring avec Prometheus et Grafana :

#### ➤ Prometheus :

Intégration de Prometheus en tant qu'outil de collecte et de suivi des métriques de performance et de santé des microservices. Prometheus permet la surveillance en temps réel des ressources système, le suivi des erreurs, la mesure de la latence des services, ainsi que le monitoring d'autres indicateurs de performance essentiels pour assurer la stabilité et l'efficacité du système.

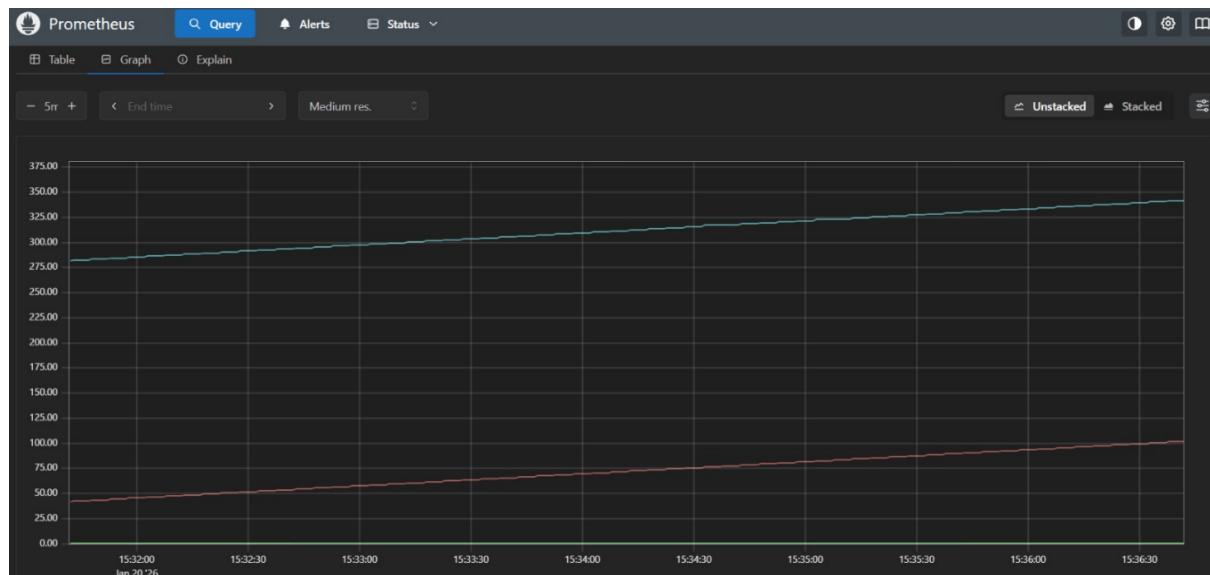


Figure 23 : Interface de Surveillance avec Prometheus – Tableau de Bord des Métriques en Temps Réel

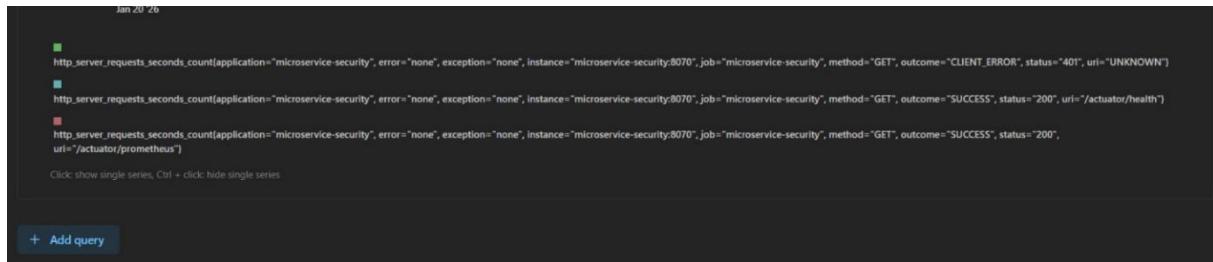


Figure 25 : Interface Prometheus – Requêtes et Métriques HTTP des Microservices



Figure 26 : Graphique Prometheus – Surveillance de l'Utilisation CPU en Temps Réel

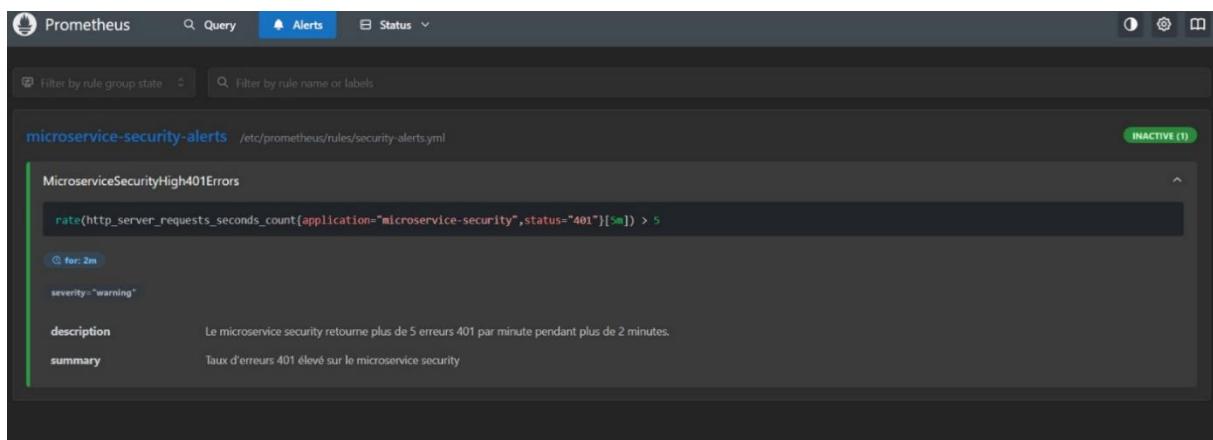


Figure 24 : Alerte Prometheus – Détection d'Erreurs 401 Élevées sur le Microservice de Sécurité

## Grafana :

### ○ Visualisation des métriques collectées par Prometheus

Grafana a été utilisé pour la visualisation des métriques collectées par Prometheus. Grâce à des tableaux de bord interactifs et personnalisables, cet outil permet une analyse détaillée des performances du système et facilite la détection rapide des problèmes potentiels.

- **Surveillance des microservices**

Grafana offre la possibilité d'agréger et de visualiser les données de chaque microservice, ce qui permet d'identifier efficacement les goulets d'étranglement et les anomalies au sein du système.

- **Suivi des dépendances**

Les visualisations générées fournissent une représentation claire des interactions entre les microservices, contribuant à la détection des erreurs ou des latences dans les flux de communication.

- **Analyse centralisée**

En centralisant les métriques provenant de différentes sources, Grafana fournit une vision unifiée des systèmes distribués, simplifiant leur gestion et leur supervision.

- **Alertes proactives**

Des seuils critiques peuvent être configurés pour générer des notifications automatiques en cas d'incidents ou de dépassement de limites, permettant ainsi une intervention rapide.

- **Gestion optimisée des ressources**

Les indicateurs relatifs à l'utilisation des ressources (CPU, mémoire, réseau) permettent d'ajuster le dimensionnement des microservices dans des environnements évolutifs et scalables, optimisant ainsi les performances globales du système.

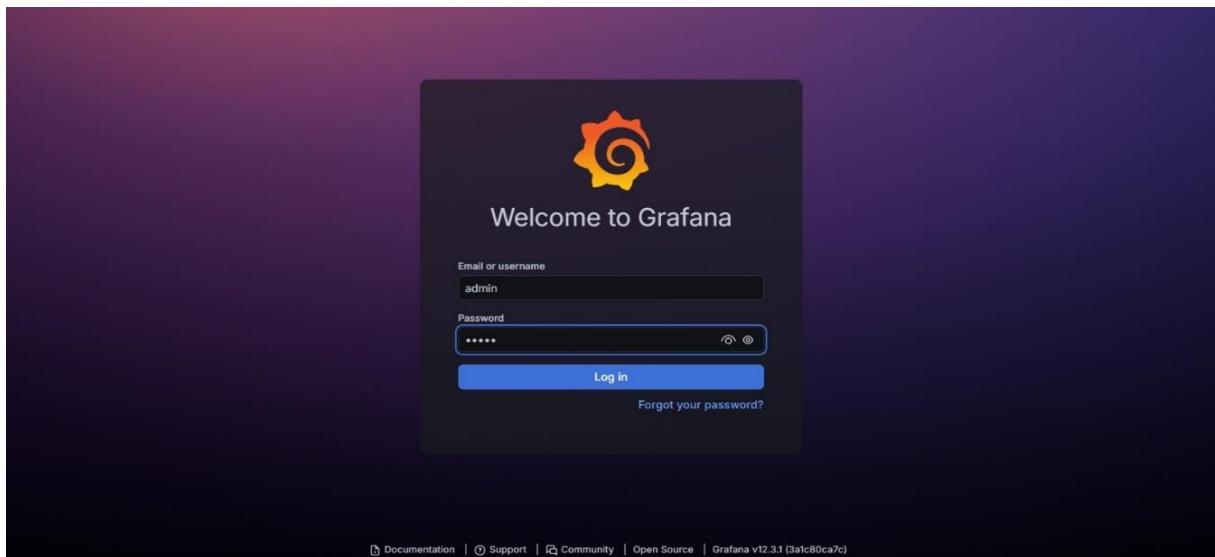


Figure 27 : Page de Connexion à l'Interface Grafana

L'interface présente la configuration d'une source de données Prometheus dans Grafana, où l'utilisateur peut renseigner l'URL du serveur (déjà pré-remplie avec `http://prometheus-service:5050`) et consulter les onglets de paramétrage tels que les tableaux de bord, les

permissions, les insights et le cache, avec une mention de l'alternative Grafana Cloud pour une solution entièrement managée.

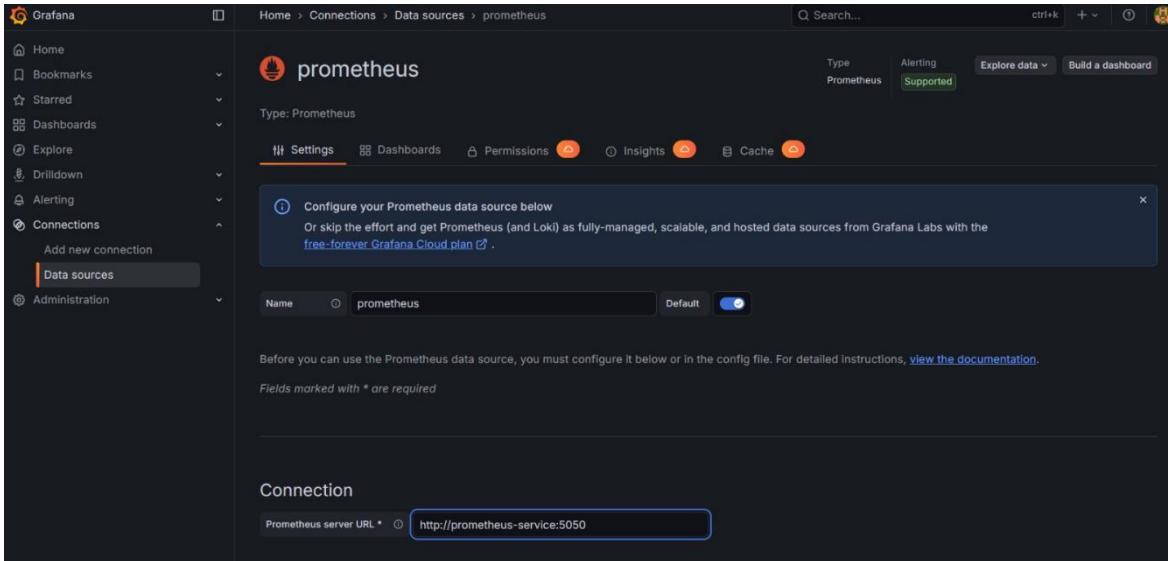


Figure 28 : Configuration de la Source de Données Prometheus dans Grafana

Le tableau de bord Grafana pour la surveillance des applications fournit un ensemble complet de métriques essentielles pour l'évaluation des performances des microservices. Ces métriques incluent le temps de fonctionnement des services, l'utilisation de la mémoire (Heap et Non-Heap), le nombre de fichiers ouverts par processus, ainsi que l'usage du CPU et la charge moyenne du système. Le tableau de bord permet une analyse centralisée et détaillée des performances de chaque microservice, facilitant ainsi l'identification des goulets d'étranglement, des anomalies et des latences dans les interactions entre services, tout en contribuant à une gestion optimisée des ressources et à l'amélioration globale de l'efficacité du système.

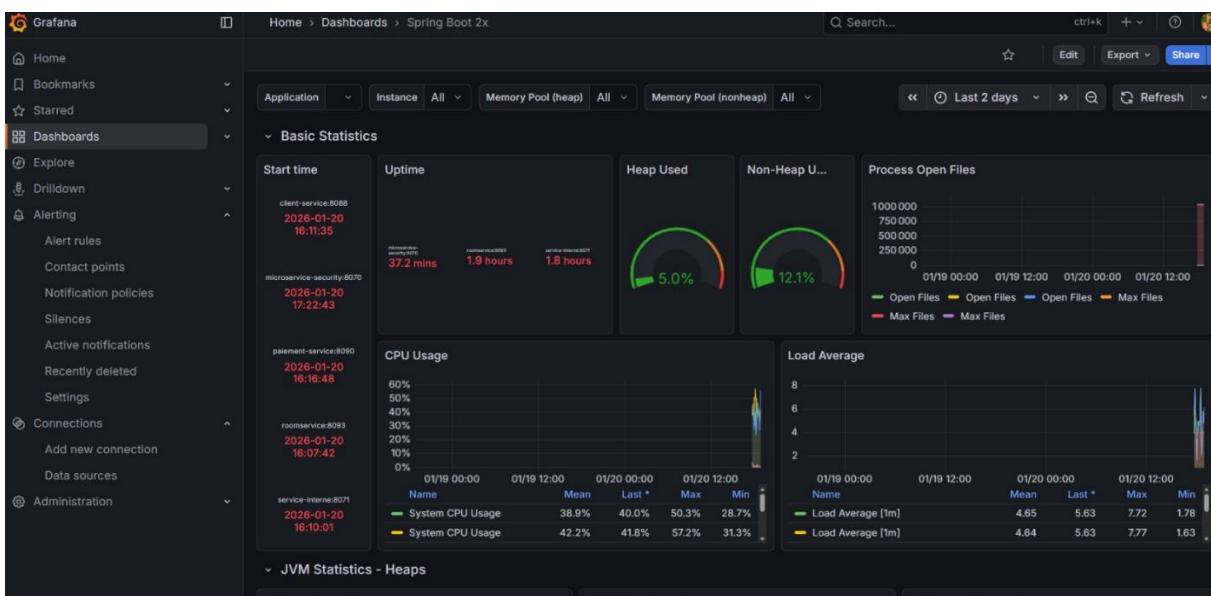


Figure 29 : Tableau de Bord Grafana – Métriques de Performance et Surveillance Système

Dashboard Grafana pour surveiller les performances d'un microservice Security. Affiche l'uptime, l'utilisation de la mémoire Heap et Non-Heap, le CPU et le nombre de fichiers ouverts.

Permet de suivre en temps réel la charge du système et l'état des ressources critiques.

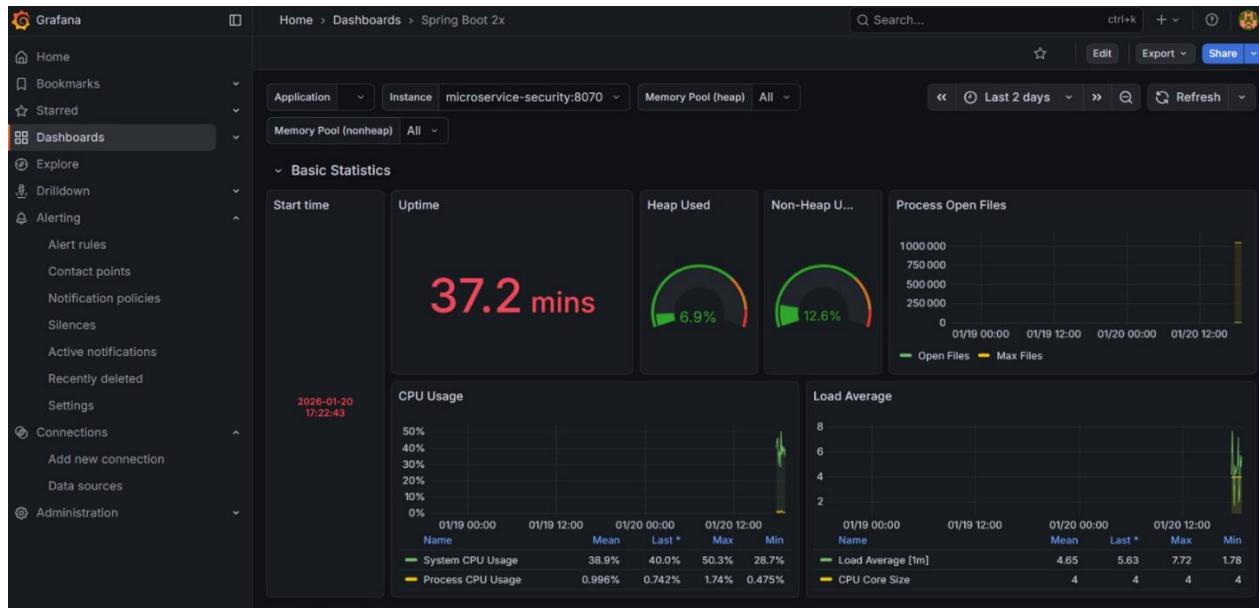


Figure 30 : Vue Synthétique du Dashboard Grafana – Indicateurs de Performance Système et CPU

L'image montre une alerte Grafana intitulée **DatasourceNoData** dans le dossier *security-alerts*, indiquant qu'une instance de l'alerte est en cours de déclenchement. Les détails incluent le nom de l'alerte, l'UID de la source de données, le dossier Grafana et le nom de la règle associée.

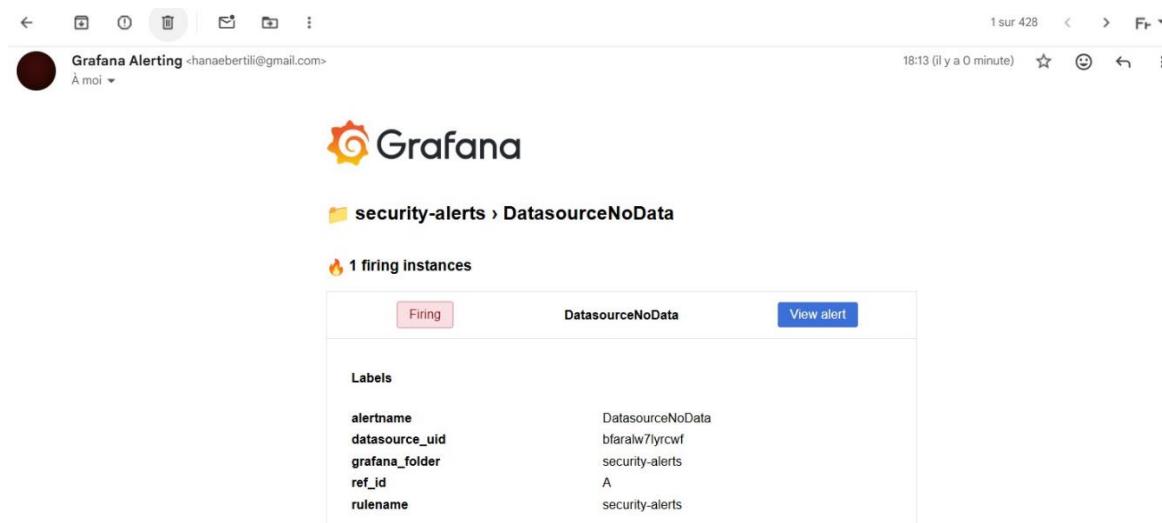


Figure 31 : Alerte Grafana – Absence de Données (DatasourceNoData) dans les Règles de Sécurité

## 2.Présentation des Interfaces Applicatives :

Dans cette section, nous exposons de manière structurée les interfaces principales de l'application web, en mettant en évidence leur rôle, leur organisation fonctionnelle, ainsi que les interactions offertes aux différents types d'utilisateurs pour assurer une expérience fluide et ergonomique.

### 2.1. Page d'accueil de la plateforme ROYELLAS HOTEL

La page d'accueil de la plateforme ROYELLAS HOTEL constitue un dispositif d'interaction numérique conçu selon les principes de l'architecture de l'information et de l'expérience utilisateur (UX) appliqués au secteur hôtelier de luxe. Elle intègre une hiérarchie visuelle et informationnelle stricte, avec une section héroïque à fort impact émotionnel positionnant immédiatement la marque dans l'imaginaire du luxe californien. La navigation persistante et la structuration modulaire du contenu (présentation des chambres, équipements, expériences) suivent un parcours utilisateur logique, optimisé pour la conversion. L'interface allie ainsi une esthétique premium — traduite par des visuels immersifs et une typographie soignée — à une fonctionnalité orientée action, où chaque élément vise à réduire la friction entre la découverte et la réservation, tout en renforçant la crédibilité et la désirabilité de l'offre.

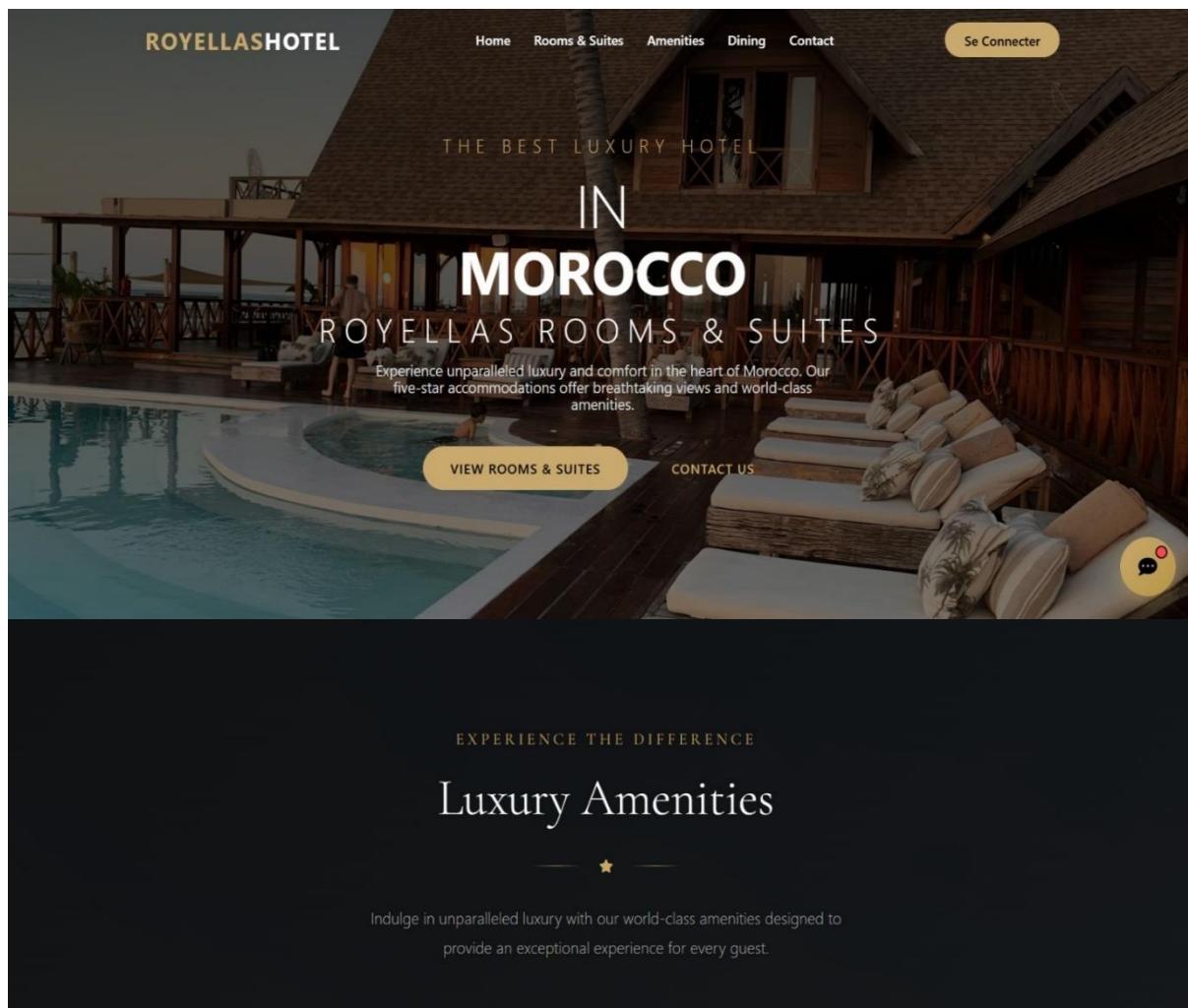


Figure 32 : Interface Web du Site ROYELLA SHOTEL

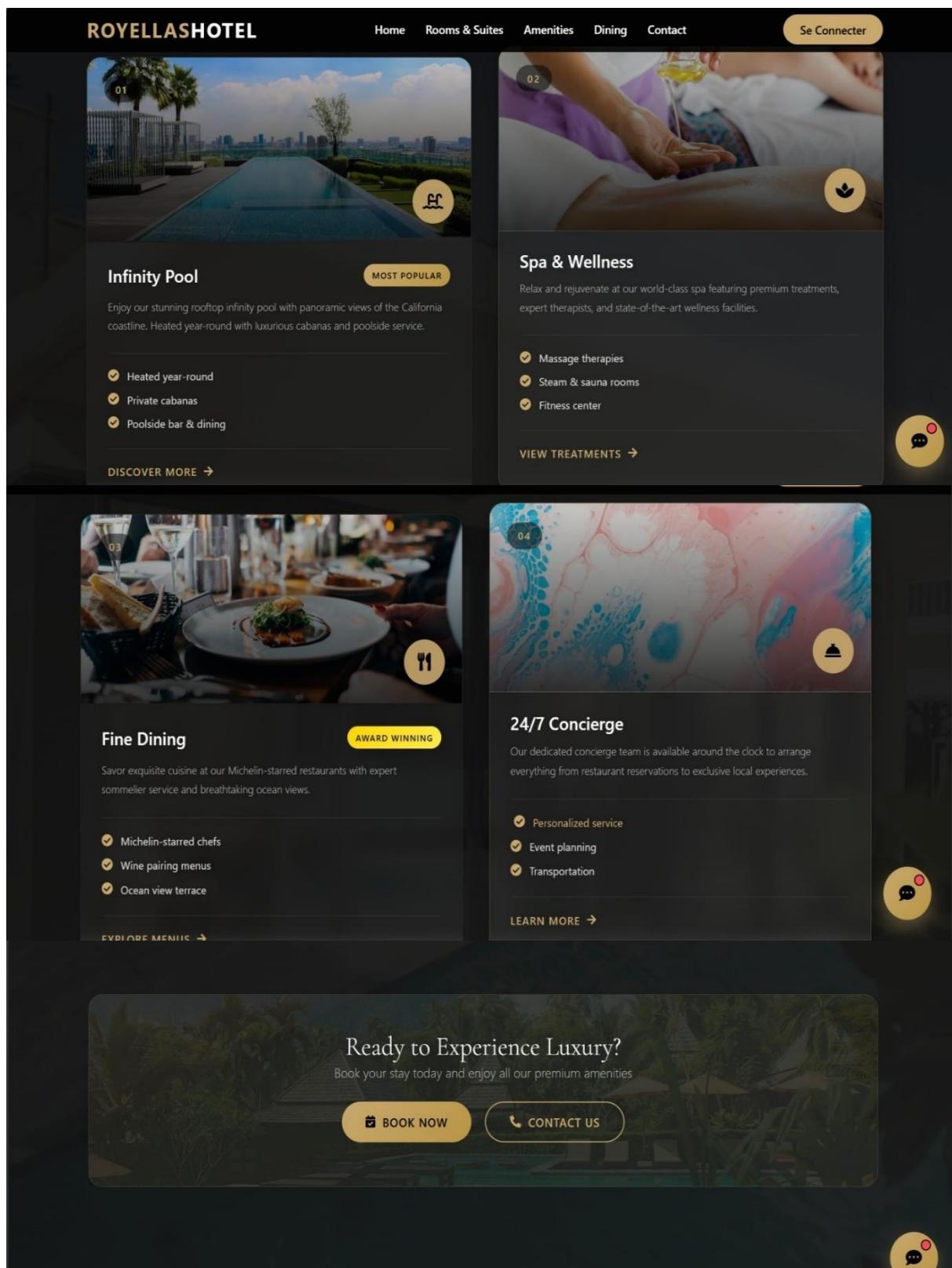


Figure 33 :Pages de Présentation et Services

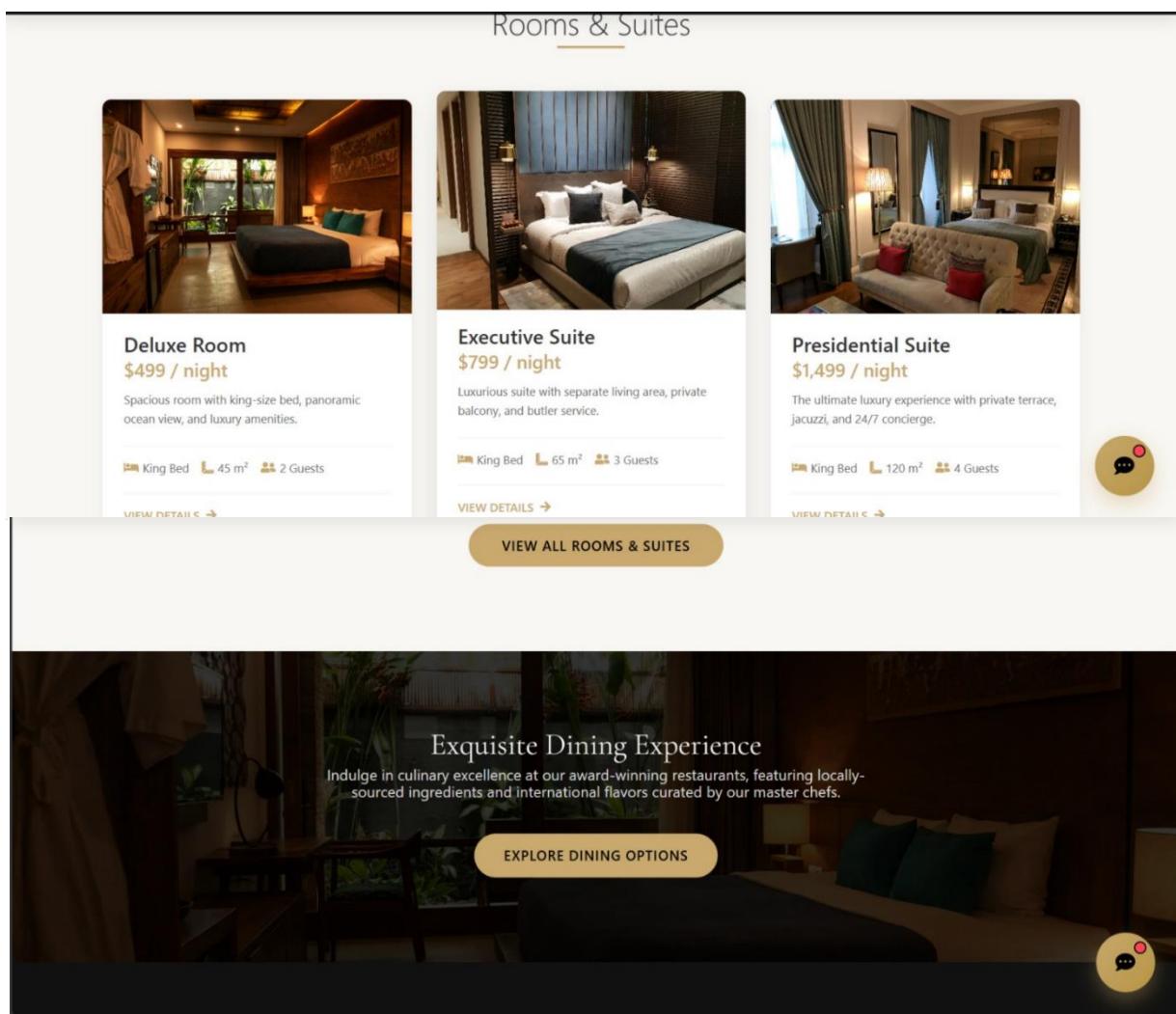


Figure 34 : Catalogue des Chambres et Présentation de la Restauration

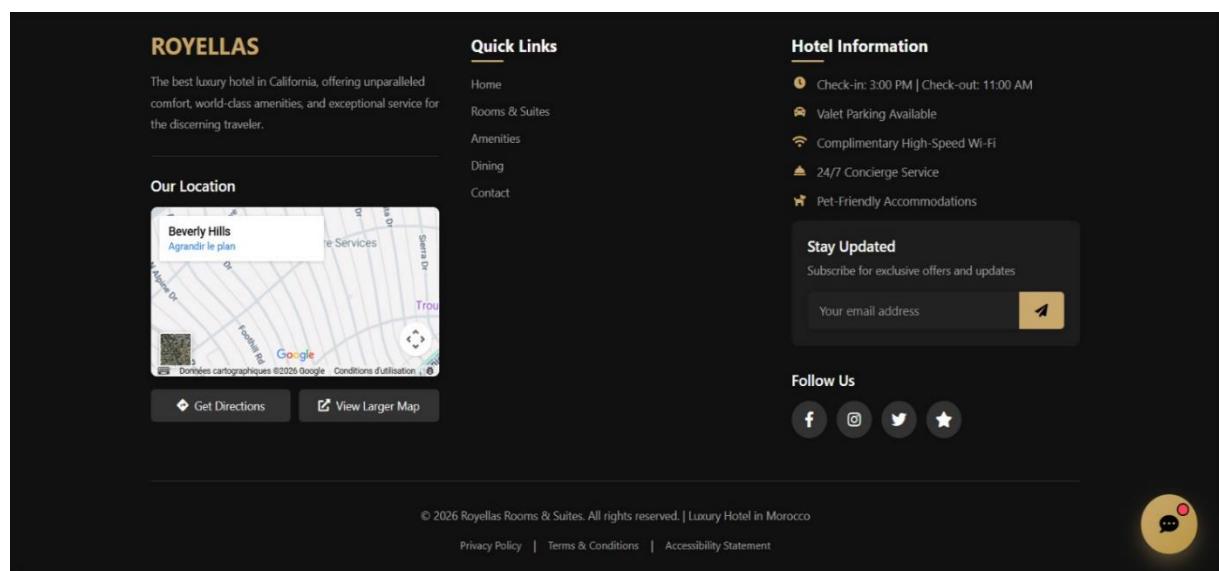


Figure 35 : Pied de Page (Footer) du Site ROYELLA SHOTEL

## 2.2. Page d'Inscription

La page d'inscription de l'hôtel Royellas se divise en deux parties : un formulaire d'inscription simplifié avec options de connexion sociale (à gauche) et une section promotionnelle présentant les avantages exclusifs de l'adhésion (à droite). Cette conception équilibre l'efficacité de conversion et le renforcement de l'image de luxe de l'établissement.

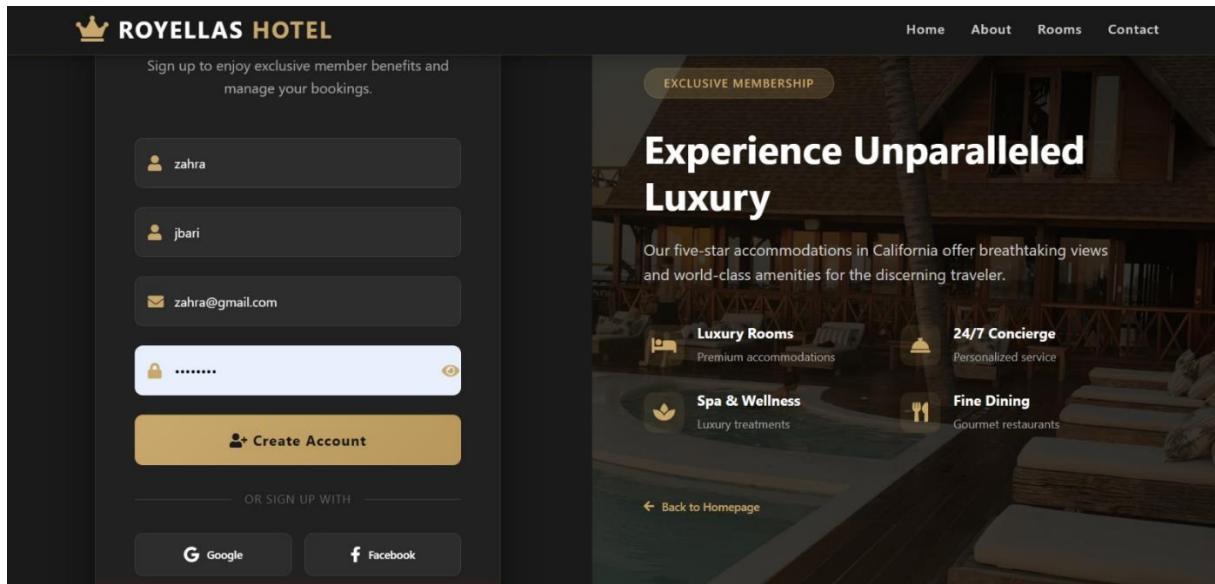


Figure 36 : Page d'Inscription

## 2.3. Interface d'authentification

L'interface de connexion présente une structure bipolaire qui équilibre la fonctionnalité d'accès et l'amélioration de l'expérience des membres. Sur la partie gauche se trouve un formulaire d'authentification succinct (email, mot de passe), tandis que la partie droite propose une section persuasive rappelant les avantages exclusifs (chambres de luxe, service de conciergerie 24h/24, spa, restaurants) afin de renforcer la valeur perçue de l'inscription. Cette conception bipartite garantit la cohérence de l'identité visuelle de la marque tout en facilitant le retour des utilisateurs enregistrés vers leur espace personnel.

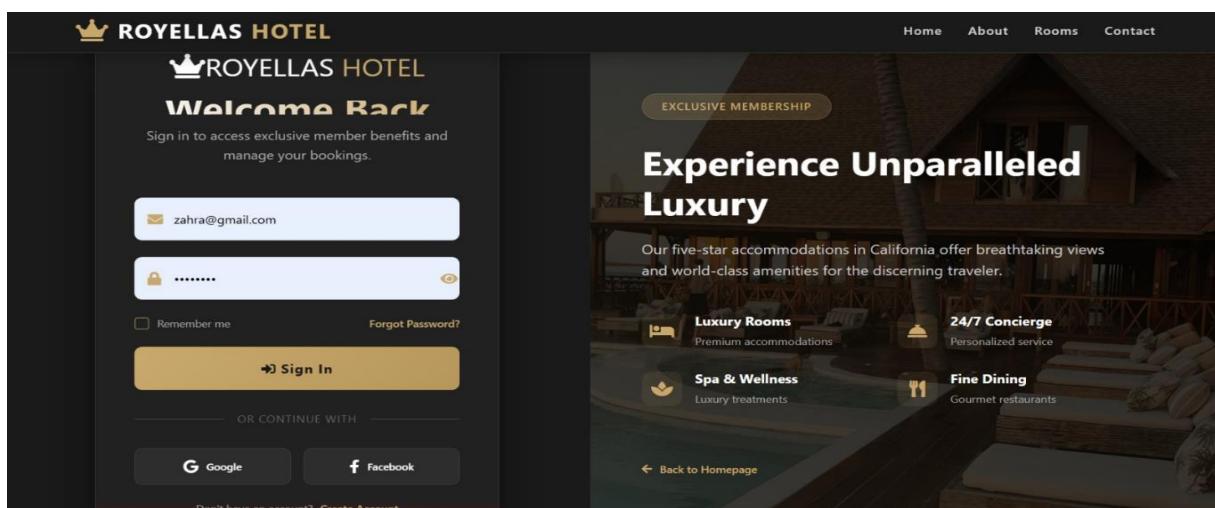


Figure 37 :Page d'authentification

## 2.4. Espace Client

### 2.4.1. Catalogue client

L'interface du catalogue client de ROYELLAS HOTEL est conçue comme un dispositif de décision assistée permettant aux utilisateurs de parcourir, filtrer et sélectionner leurs options d'hébergement. Elle combine une recherche textuelle ("Search rooms by number...") avec des filtres catégoriels ("Type Room", "Price Range") pour affiner la sélection. Chaque chambre est présentée sous forme de carte standardisée incluant : le nom, le prix par nuit, une description synthétique des équipements, et une indication visuelle du type de lit et des dimensions. Le design vise à équilibrer densité informationnelle et clarté visuelle, facilitant la comparaison rapide entre les options tout maintenant l'identité luxueuse de la marque.

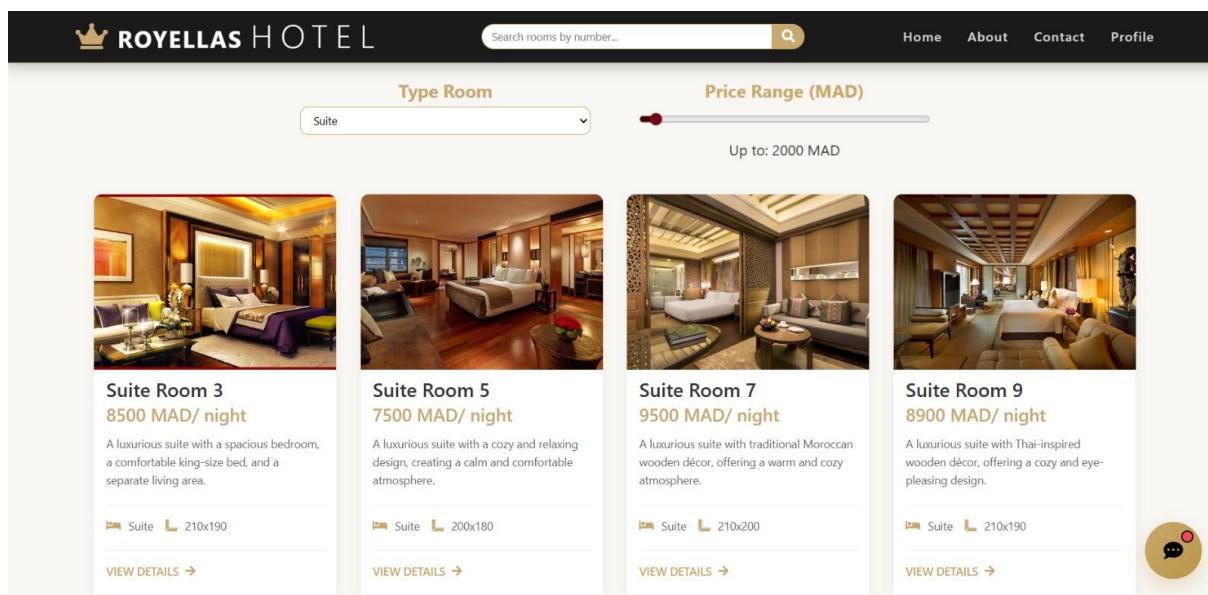


Figure 38 : Interface Client – Catalogue des Chambres avec Recherche et Filtrage

#### ➕ Recherche par numéro de chambre

Cette fonctionnalité permet aux utilisateurs de localiser rapidement une chambre spécifique en saisissant son numéro, en filtrant les résultats et en mettant en évidence la chambre correspondante, facilitant ainsi la décision tout en conservant la clarté de l'interface

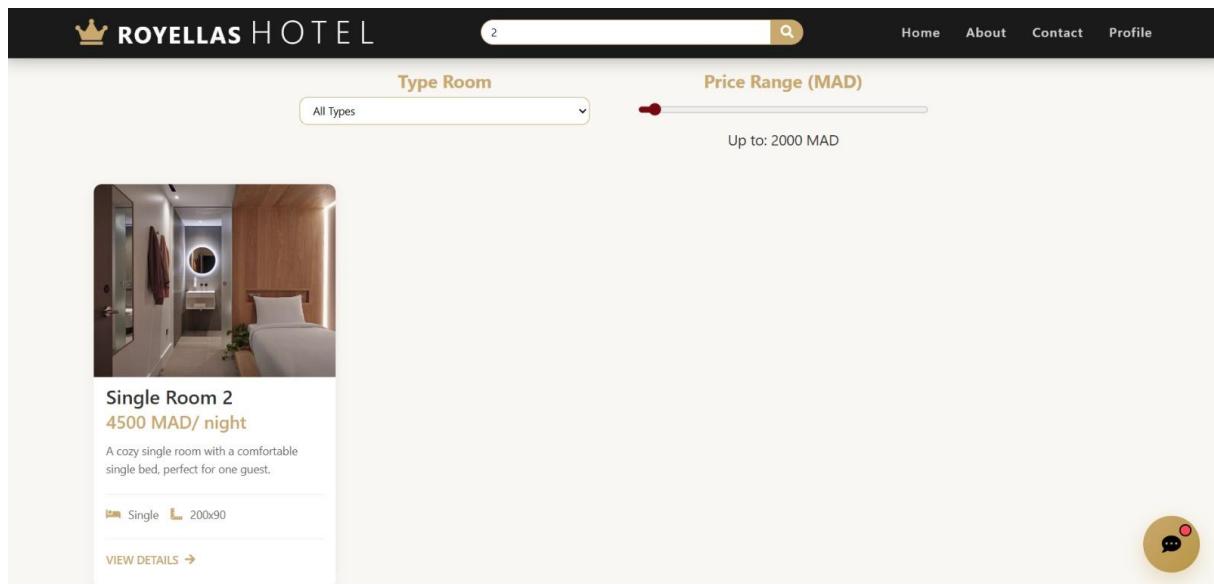


Figure 39 : Recherche par numéro de chambre

#### Recherche par type de chambre

Cette fonctionnalité permet de filtrer les chambres par type (Single, Double, Suite) pour faciliter la comparaison et accélérer le choix de l'utilisateur.

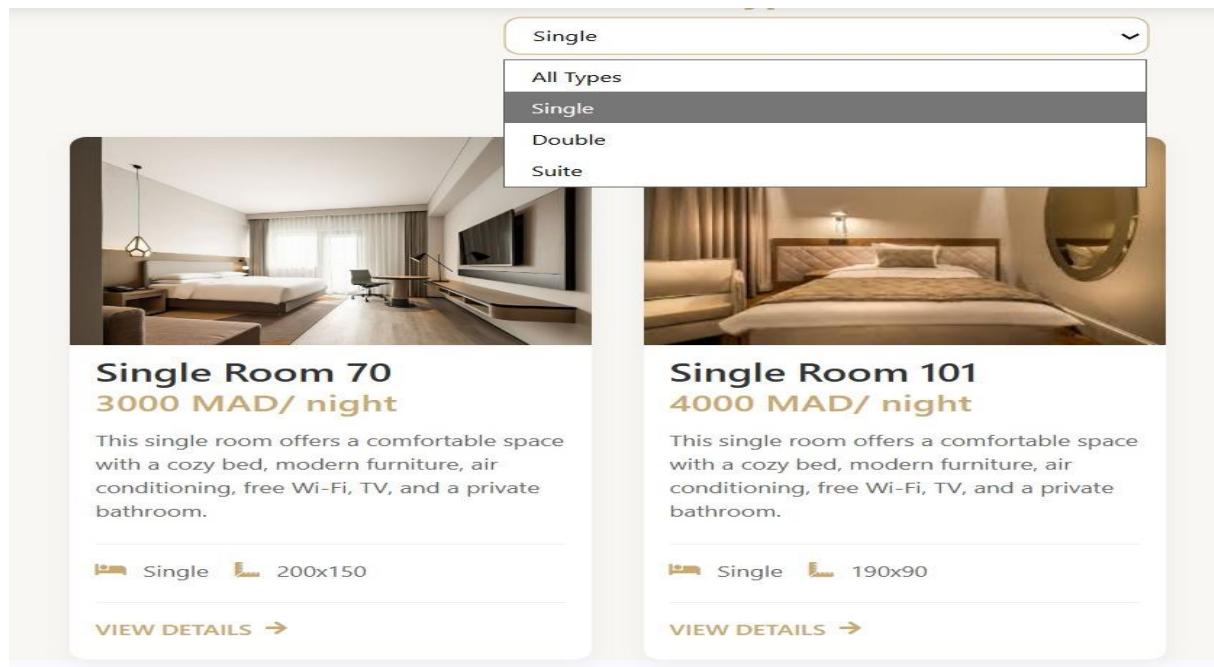


Figure 40 : Recherche par type de chambre

#### Recherche par prix

La fonctionnalité de **recherche par prix** permet aux utilisateurs de filtrer les chambres selon une fourchette tarifaire définie. Cela facilite l'identification rapide des options correspondant

au budget de chaque utilisateur, optimisant ainsi la prise de décision et améliorant l'expérience de navigation.

Figure 41 : Recherche par prix

### Page de détail de la chambre

La page de détail présente une chambre spécifique avec toutes ses informations : photos haute résolution, description complète des équipements, type et dimensions du lit, tarifs et disponibilités. Elle vise à fournir une vue exhaustive et immersive, facilitant la décision de réservation tout en maintenant l'identité luxueuse de la marque.

Figure 42 : Page de détail de la chambre

## ✚ Chambres similaires

Cette section propose des chambres alternatives présentant des caractéristiques proches de la chambre consultée (type, prix, équipements), afin de faciliter la comparaison et d'optimiser la prise de décision de l'utilisateur.



Figure 43 : Chambres similaires

### 2.4.2. Profile client

Cette interface permet à l'utilisateur de consulter et de modifier ses informations personnelles et paramètres de compte, assurant une gestion autonome, sécurisée et fluide de son profil.

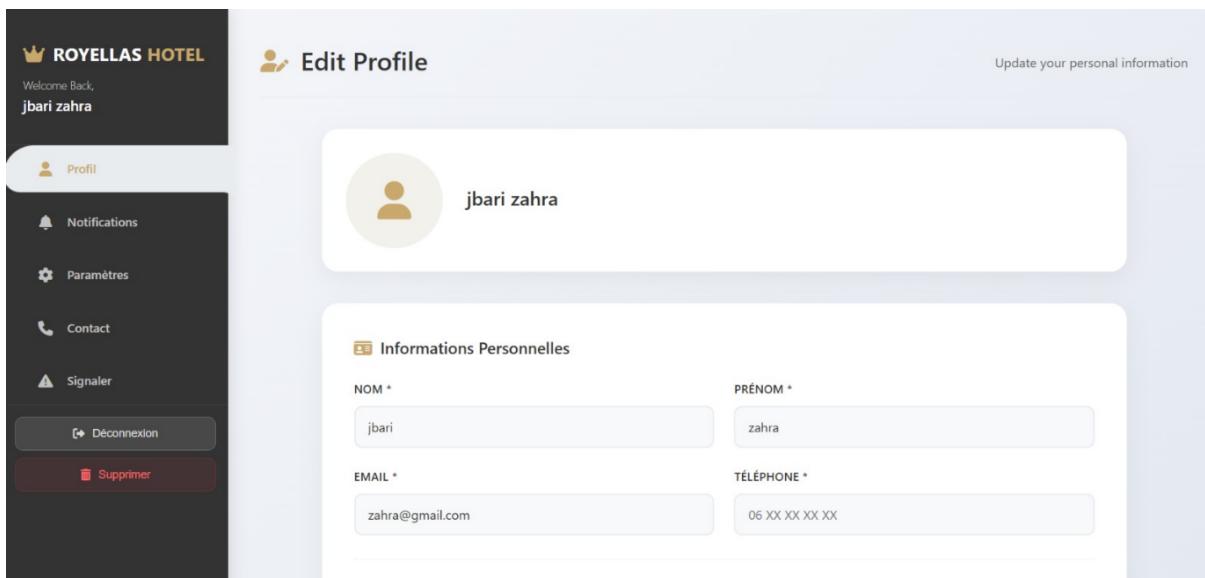


Figure 44 : Page Profile client

## ✚ Modifier profile

Dans l'interface de modification du profil, l'utilisateur peut mettre à jour ses informations personnelles, comme le **numéro de téléphone**. Par exemple, le numéro a été modifié de **06xx xx xx xx** à **0698562138** à des fins de test, tout en garantissant que la mise à jour s'effectue via une interface claire et sécurisée.

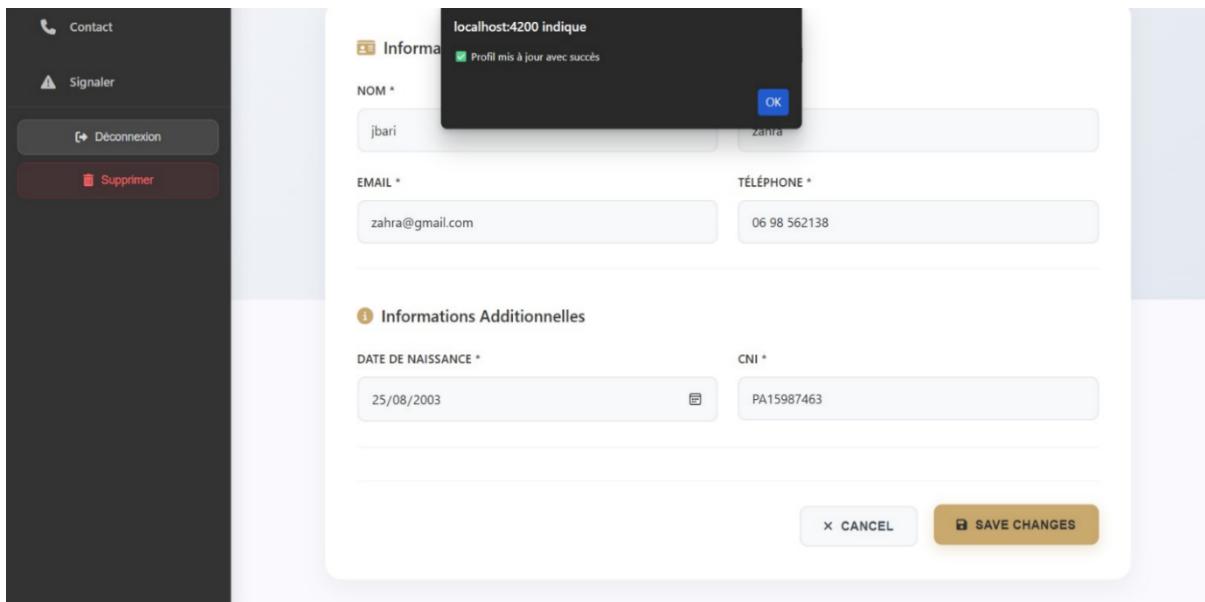


Figure 45 : Page pour modifier profile Client

#### 2.4.3. Réservation de chambre

L’interface de réservation présente une vue structurée de la chambre incluant son image, ses caractéristiques principales et son statut de disponibilité. Elle intègre un bouton d’action dédié à la réservation, visant à optimiser la clarté de l’information et à faciliter la prise de décision de l’utilisateur dans un parcours fluide et orienté conversion.

Les chambres peuvent se présenter sous trois statuts :

- **Occupée** : l’utilisateur ne peut pas effectuer de réservation.

**ROYELLAS HOTEL**

[← Back to Catalogue](#)

**Double Room 4**

Here : A peaceful room with a relaxing design, featuring a balcony that overlooks the sea.

Discover the epitome of luxury at ROYELLAS Hotel. Each room and suite is thoughtfully designed to blend elegance with comfort, featuring breathtaking views, plush bedding, and world-class amenities. From intimate retreats to expansive suites, every space promises an unforgettable stay tailored for discerning travelers. Experience the art of hospitality, where every detail is crafted to delight and every moment is designed to inspire.

Price: 6000 MAD/night

Double 200x190

THIS ROOM IS: OCCUPÉE

Reserve a Room

Figure 46 : Page de détail – Chambre Double 4 (Statut : Occupée)

- **En maintenance** : la chambre est temporairement indisponible et ne peut pas être réservée jusqu’à la fin de la maintenance.

**ROYELLAS HOTEL**

[← Back to Catalogue](#)

## Single Room 2



Here : A cozy single room with a comfortable single bed, perfect for one guest.

Discover the epitome of luxury at ROYELLAS Hotel. Each room and suite is thoughtfully designed to blend elegance with comfort, featuring breathtaking views, plush bedding, and world-class amenities. From intimate retreats to expansive suites, every space promises an unforgettable stay tailored for discerning travelers. Experience the art of hospitality, where every detail is crafted to delight and every moment is designed to inspire.

**Price: 4500 MAD/night**

 Single  200x90

[THIS ROOM IS: MAINTENANCE](#) [Reserve a Room](#)

Figure 47 : Page de détail – Chambre Single 2 (Statut : Maintenance)

- **Disponible** : la chambre peut être sélectionnée et réservée.

**ROYELLAS HOTEL**

[← Back to Catalogue](#)

## Suite Room 7



Here : A luxurious suite with traditional Moroccan wooden décor, offering a warm and cozy atmosphere.

Discover the epitome of luxury at ROYELLAS Hotel. Each room and suite is thoughtfully designed to blend elegance with comfort, featuring breathtaking views, plush bedding, and world-class amenities. From intimate retreats to expansive suites, every space promises an unforgettable stay tailored for discerning travelers. Experience the art of hospitality, where every detail is crafted to delight and every moment is designed to inspire.

**Price: 9500 MAD/night**

 Suite  210x200

[THIS ROOM IS: DISPONIBLE](#) [Reserve a Room](#)

Figure 48 : Page de détail – Suite Room 7 (Statut : Disponible)

## Formulaire de réservation

Le formulaire permet à l'utilisateur de saisir ses informations personnelles et les dates de séjour, facilitant la confirmation rapide et sécurisée de la réservation.

Figure 49 : Formulaire de Réservation en Ligne

#### 2.4.4. Suivi de réservation et notifications

##### Interface de confirmation de réservation

Après avoir complété le formulaire de réservation, l'utilisateur est redirigé vers son profil, où un message l'informe que sa réservation est en attente : **"Reservation pending. You will receive a confirmation or a rejection. Please check the notification section to stay informed about the status of your reservation."** Cette interface permet de rassurer l'utilisateur tout en l'incitant à consulter régulièrement les notifications pour suivre l'évolution de sa réservation.



Figure 51 : Espace Client – Profil Utilisateur avec Suivi de Réservation

Figure 50 : Profil Utilisateur avec Suivi de Réservation

## ❖ Interface des notifications

### ❖ Statut Pending

La section **Notifications** affiche le statut actuel de la réservation. Tant que la réception n'a pas confirmé la réservation, le statut reste **Pending**. Une fois validée ou rejetée, le message est mis à jour automatiquement, assurant un **suivi clair et en temps réel** de l'état de la réservation.

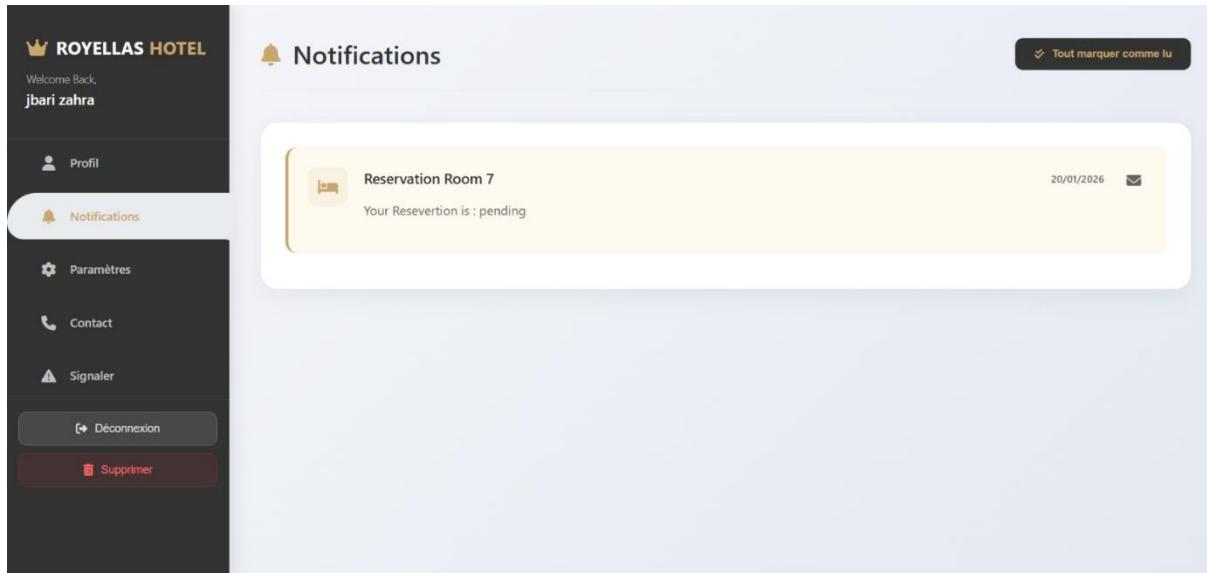


Figure 52 : Section Notifications avec Suivi de Réservation

### ❖ Statut Rejected

Réservation rejetée : le statut est mis à jour à **Rejected**, indiquant que la réservation n'a pas été validée.

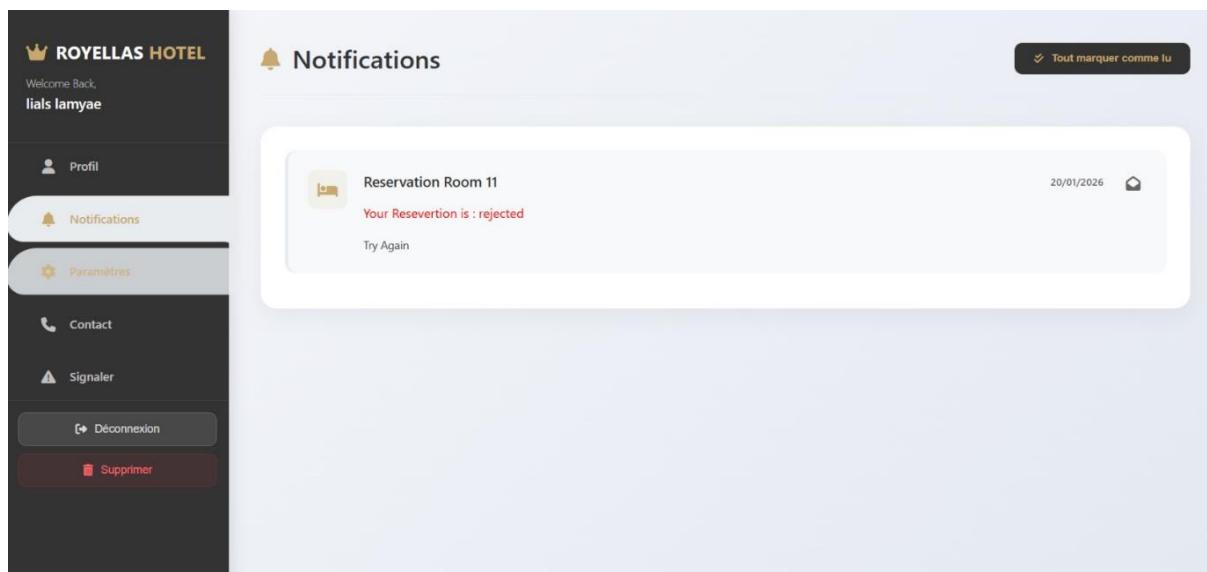


Figure 53 : Notification de Réservation Rejetée

### ❖ Statut Confirmed

Après validation par la réception, le statut de la réservation est mis à jour à « Confirmed », informant ainsi l'utilisateur que sa réservation est désormais confirmée et qu'il peut initier le processus de paiement.

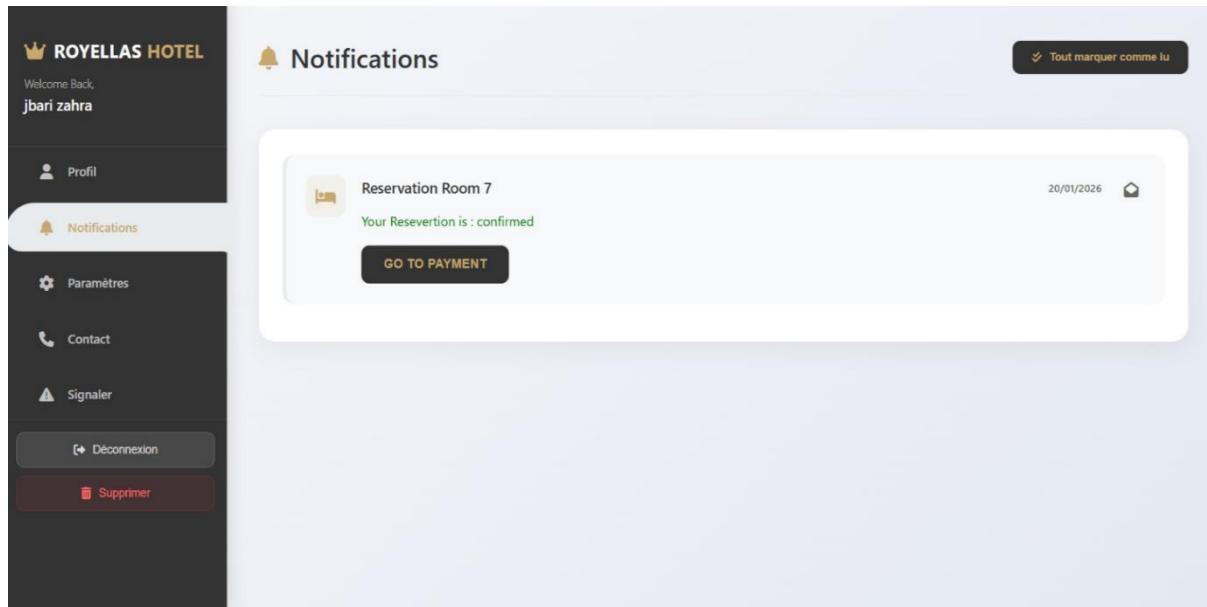


Figure 54 : Notification de Réservation Confirmée avec Accès au Paiement

#### 2.4.5. Interface de paiement

L'interface de paiement permet à l'utilisateur de finaliser sa réservation après confirmation par la réception, en présentant un récapitulatif de la réservation incluant la chambre sélectionnée, les dates de séjour et le montant total. L'interface prend en charge plusieurs méthodes de paiement : carte bancaire et PayPal, Cash offrant ainsi une flexibilité d'utilisation. Le design vise à garantir un parcours de paiement sécurisé, clair et fluide, assurant la finalisation de la transaction en toute confiance.

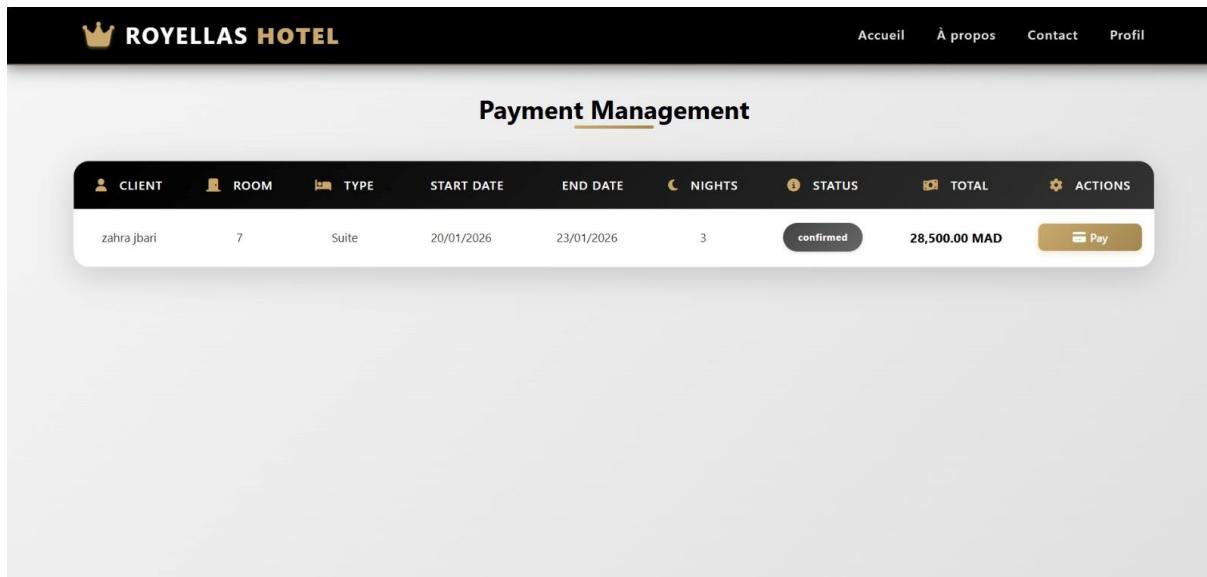


Figure 55 : Interface de gestion des paiements des réservations

**Payment - Reservation #60**

**Reservation Details**

- Client:** zahra jbari
- Room:** 7
- Period:** 20/01/2026 - 23/01/2026
- Nights:** 3
- Total:** 28,500.00 MAD

**Payment Method**

- Credit Card
- PayPal
- Cash

**Cancel** **✓ Confirm Payment**

Figure 56 : Interface de paiement

### ⊕ Paiement par carte bancaire

Lorsque l'utilisateur choisit l'option de paiement par Credit Card, l'interface affiche un formulaire sécurisé pour saisir les informations nécessaires : numéro de carte, date d'expiration, code CVV et nom du titulaire. Ce processus est conçu pour garantir la sécurité des données sensibles, tout en offrant une expérience rapide et intuitive jusqu'à la confirmation du paiement.

**Payment - Reservation #60**

**Bank Transfer**

**Card Information**

- Card Number:** 1234 5678 9012 3456
- Expiration Date:** zahra@gmail.com
- CVV:** ...

**VISA** **MasterCard** **AM EX**

**Cancel** **✓ Confirm Payment**

Figure 57 : Paiement par carte bancaire

## ⊕ Paiement via PayPal

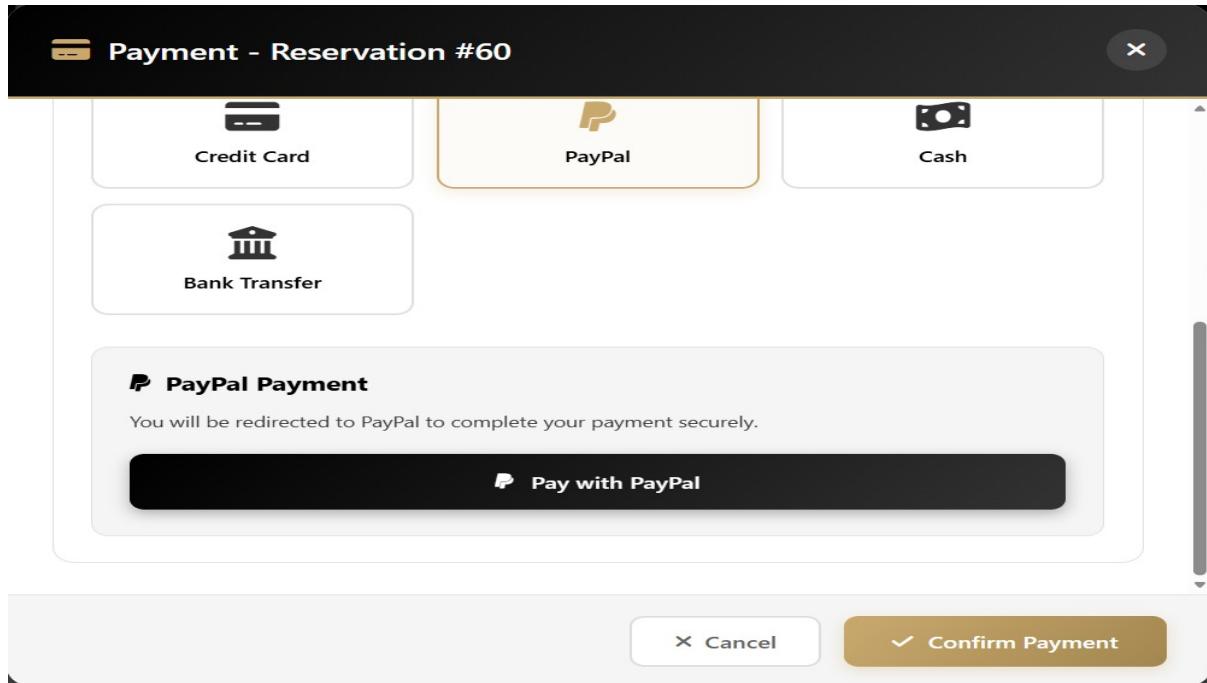


Figure 58 : Paiement via PayPal

## ⊕ Paiement par Cash

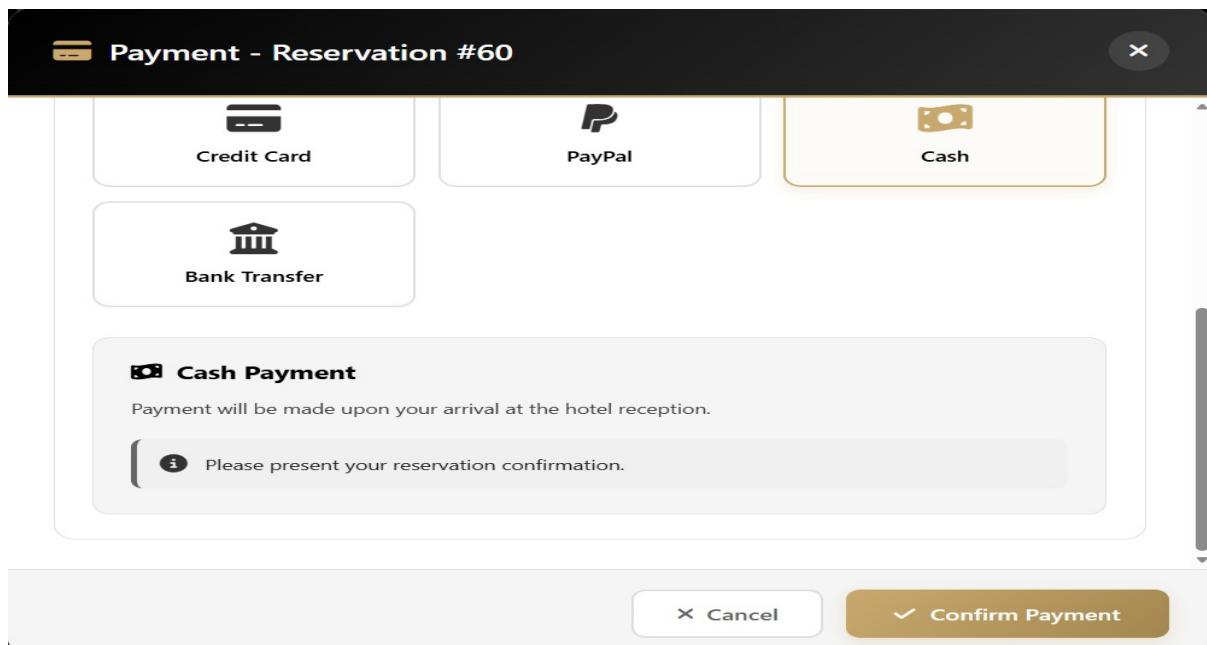
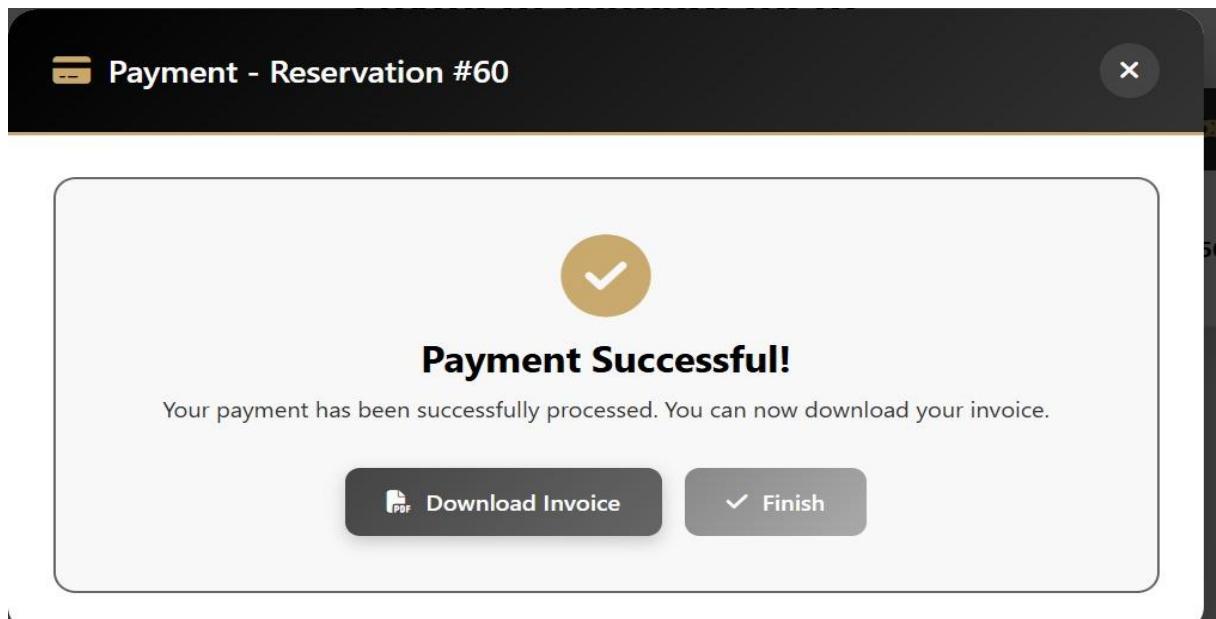


Figure 59 : Paiement Cash

## ⊕ Paiement réussi

Cette interface affiche un message confirmant le succès du paiement et permet à l'utilisateur de



*Figure 56 : Paiement réussi*

télécharger la facture pour la consulter ou la conserver ultérieurement.

 Facture

# ROYELLAS HOTEL

## OFFICIAL INVOICE

Invoice N°: INV-60-2026  
Invoice Date: 20/01/2026  
Due Date: 19/02/2026

**CLIENT INFORMATION**

Name: zahra jbari  
Email: zahra@gmail.com  
Reservation ID: #60

**ROYELLAS HOTEL**

Marrakech, Morocco  
+212 5 00 00 00 00  
contact@royellashotel.com  
[www.royellashotel.com](http://www.royellashotel.com)

**STAY DETAILS**

Room Number	7
Room Type	Suite
Check-in Date	20/01/2026
Check-out Date	23/01/2026
Nights	3
Payment Method	Virement bancaire

Thank you for your trust!

## 2.6. Espace réceptionniste

### 2.6.1. Profil du réceptionniste

Cette interface permet au réceptionniste de consulter et de modifier ses informations personnelles et paramètres de compte, assurant une gestion autonome, sécurisée et efficace de son profil professionnel.

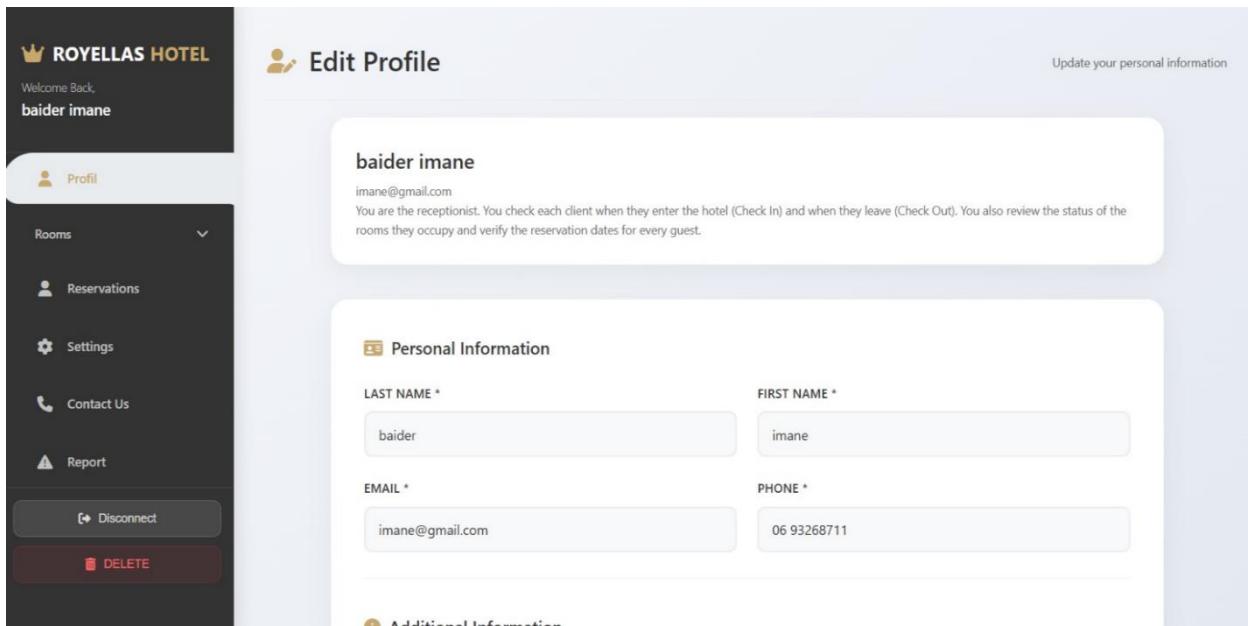


Figure 57 : Profil du réceptionniste

### Modifier profile

Dans l'interface de modification du profil, réceptionniste peut mettre à jour ses informations personnelles.

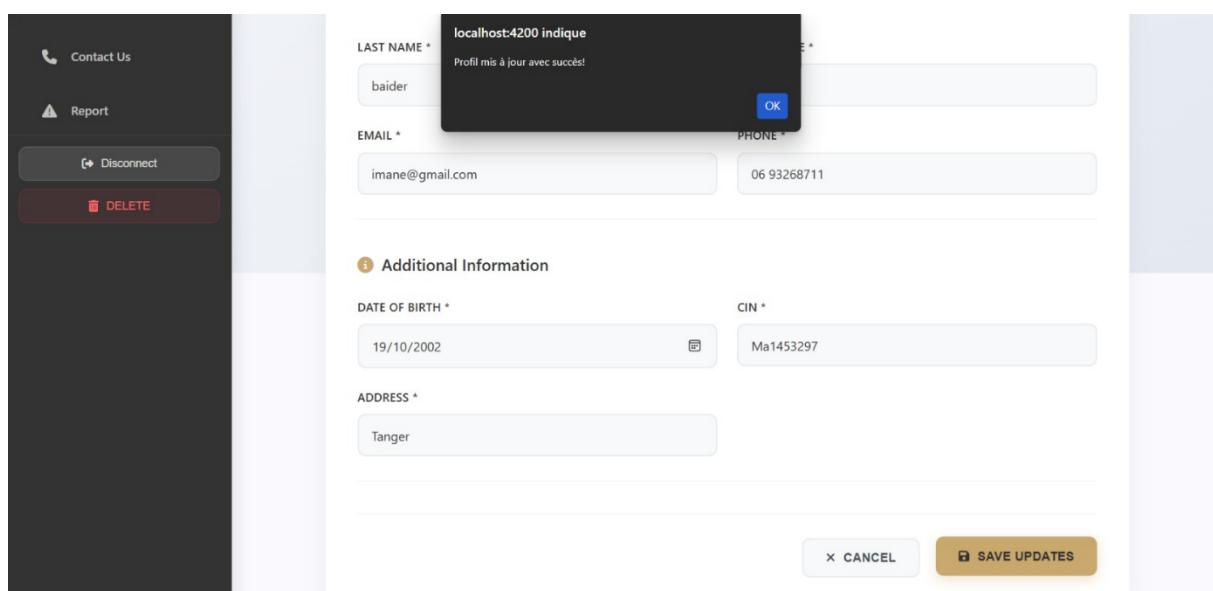


Figure 58 : Modifier profile Réceptionniste

## 2.6.2. Gestion des chambres

L'interface de l'espace réceptionniste permet de visualiser l'ensemble des chambres de l'hôtel avec leurs statuts respectifs (Disponible, Occupée, Maintenance). Elle fournit un tableau centralisé incluant les informations essentielles de chaque chambre : type, prix, et statut.

The screenshot shows a user interface for managing hotel rooms. On the left, there's a sidebar with navigation links: Profil, Rooms (selected), View Rooms, Reservations, Settings, Contact Us, Report, Disconnect, and a large red DELETE button. The main area is titled 'All Rooms' and displays a table of room details. The columns are: #, Picture, Number, Type, Description, Price, and Status. The data is as follows:

#	Picture	Number	Type	Description	Price	Status
1		2	Single	A cozy single room with a comfortable single bed, perfect for one guest.	4500 MAD/NIGHT	Maintenance
2		3	Suite	A luxurious suite with a spacious bedroom, a comfortable king-size bed, and a separate living area.	8500 MAD/NIGHT	Disponible
3		4	Double	A peaceful room with a relaxing design, featuring a balcony that overlooks the sea.	6000 MAD/NIGHT	Occupée
4		5	Suite	A luxurious suite with a cozy and relaxing design, creating a calm and comfortable atmosphere.	7500 MAD/NIGHT	Disponible
5		6	Single	A single room with a calm and relaxing atmosphere, perfect for a peaceful stay.	4500 MAD/NIGHT	Maintenance

Figure 59 : page chambres ( gérer les chambres)

Pour faciliter l'analyse et la surveillance en temps réel, l'interface propose également des diagrammes et graphiques :

Répartition des chambres par statut : un diagramme circulaire montre la proportion de chambres disponibles, occupées ou en maintenance.

Répartition des prix par type de chambre : un graphique indique les tarifs selon le type (Single, Double, Suite), facilitant la planification et la gestion commerciale.

Cette approche offre au réceptionniste une vision complète et intuitive de l'état de l'hôtel, permettant des décisions rapides et un suivi efficace des réservations et de la maintenance.

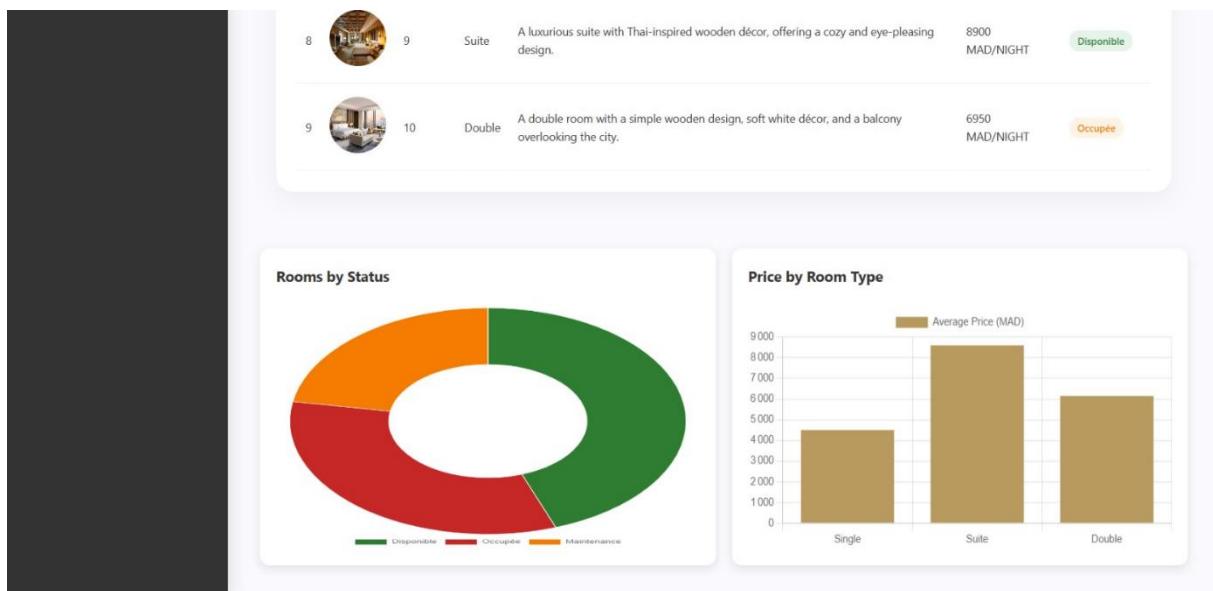


Figure 60 : suivi les Statistiques des chambres

### 2.6.3. Gestion des réservations

L’interface de gestion des réservations permet au réceptionniste de visualiser toutes les réservations effectuées par les clients dans un tableau structuré, incluant les informations essentielles telles que le nom du client, le type de chambre, les dates de séjour et le statut de la réservation.

Pour chaque réservation, le réceptionniste dispose de boutons d’action permettant de confirmer (Accepter) ou refuser (Rejeter) la réservation, facilitant ainsi la gestion en temps réel.

Afin de visualiser rapidement l’état des réservations, l’interface inclut également des diagrammes et graphiques :

- Diagramme circulaire montrant la répartition des réservations par chambre.
- Graphique des prix moyens par type de chambre, permettant d’analyser la performance tarifaire et de prendre des décisions commerciales éclairées.

Cette interface offre au réceptionniste une vision globale et claire des réservations, optimisant la réactivité et l'efficacité dans la gestion des clients et de l'occupation des chambres.

The screenshot shows the 'All Reservations' section of the hotel management system. On the left, a sidebar menu includes 'Profil', 'Rooms', 'View Rooms', 'Reservations', 'Settings', 'Contact Us', 'Report', 'Disconnect', and a red 'DELETE' button. The main area displays a table of reservations:

ID	Client Name	Client Email	CNI	Room Number	Room Type	Start Date	End Date	Status	Total Price	Actions
60	jbari zahra	zahra@gmail.com	PA15987463	7	Suite	20/01/2026	23/01/2026	<span style="color: orange;">pending</span>	28,500.00 MAD	<span style="color: green;">Confirm</span> <span style="color: red;">Reject</span>

Below the table are two charts: a donut chart titled 'Reservations by Room Type (%)' showing Suite at 100%, and a bar chart titled 'Average Price by Room Type (MAD)' showing Suite at 28,500.00 MAD.

Figure 61 : Page Réservations (gérer les réservations et suivre les statistiques)

### 2.3.6.1. Mise à jour automatique du statut de la chambre

Après qu'une réservation a été traitée par le réceptionniste :

- **Réservation acceptée** : le statut de la chambre passe à « Occupée » et est visible dans le tableau des chambres.

The screenshot shows a confirmation dialog box with the message 'localhost:4200 indique' and 'La chambre est maintenant OCCUPÉE' with an 'OK' button. Below the dialog is the same 'All Reservations' table as in Figure 61, but the status column for the row with ID 60 now shows 'confirmed' with a checkmark and the word 'Confirmed'.

Figure 62 : Mise à jour statut de la chambre

**Réservation rejetée** : le statut de la chambre revient à « **Disponible** », garantissant que la chambre peut être réservée par un autre client.

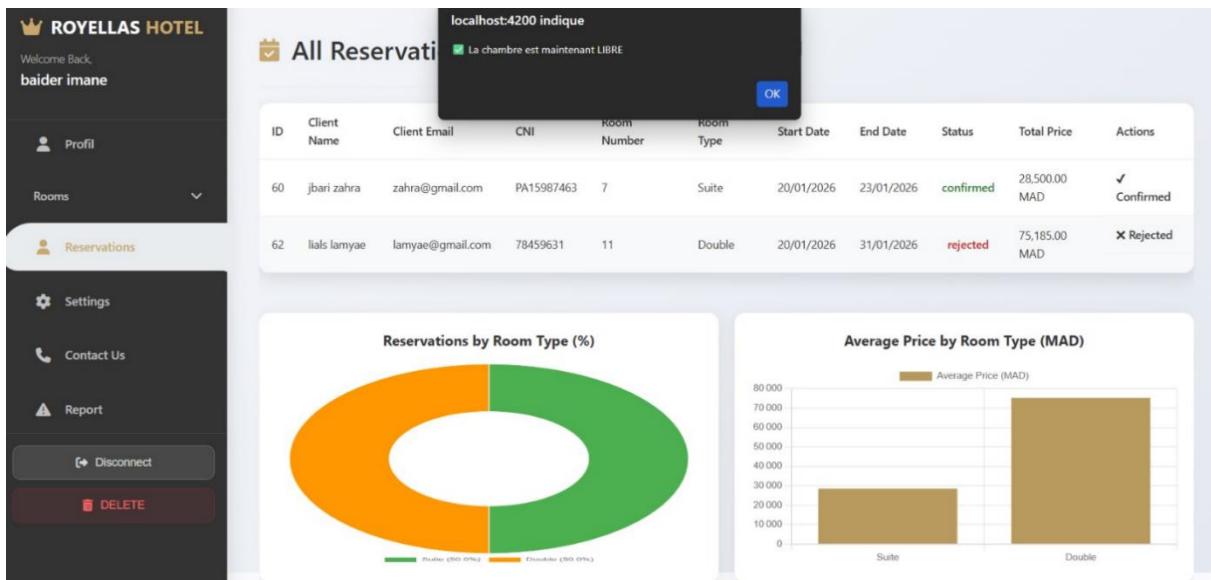


Figure 63 : modifier le statut d'une chambre

## 2.7. Espace Manager

### 2.7.2. Dashboard Manager

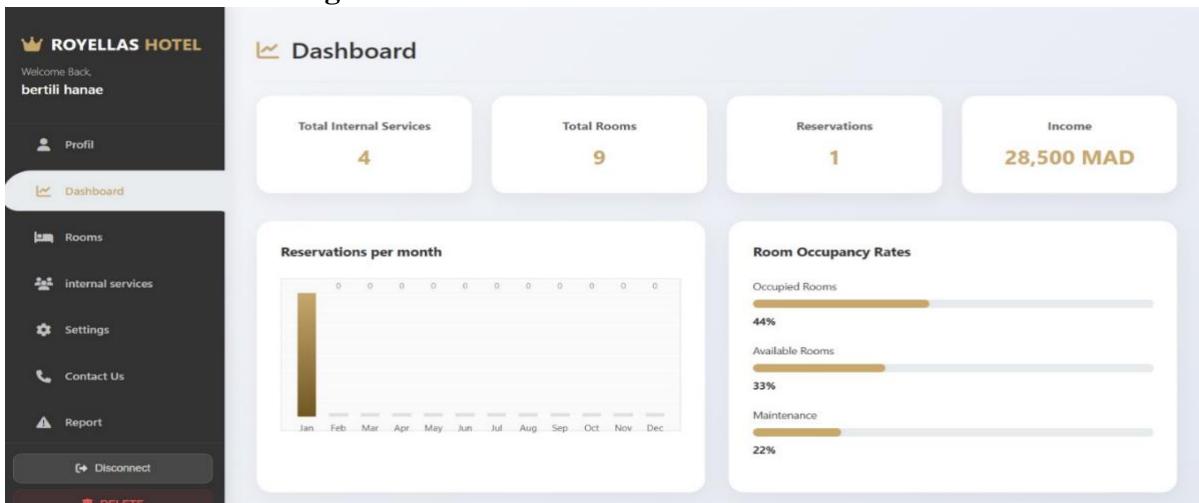


Figure 64 : Dashboard de Manager

### 2.7.3. Profile Manager

Cette interface permet au **manager** de consulter et de modifier ses informations personnelles et paramètres de compte, assurant une gestion **autonome, sécurisée et optimale** de son profil professionnel et de ses responsabilités de supervision.

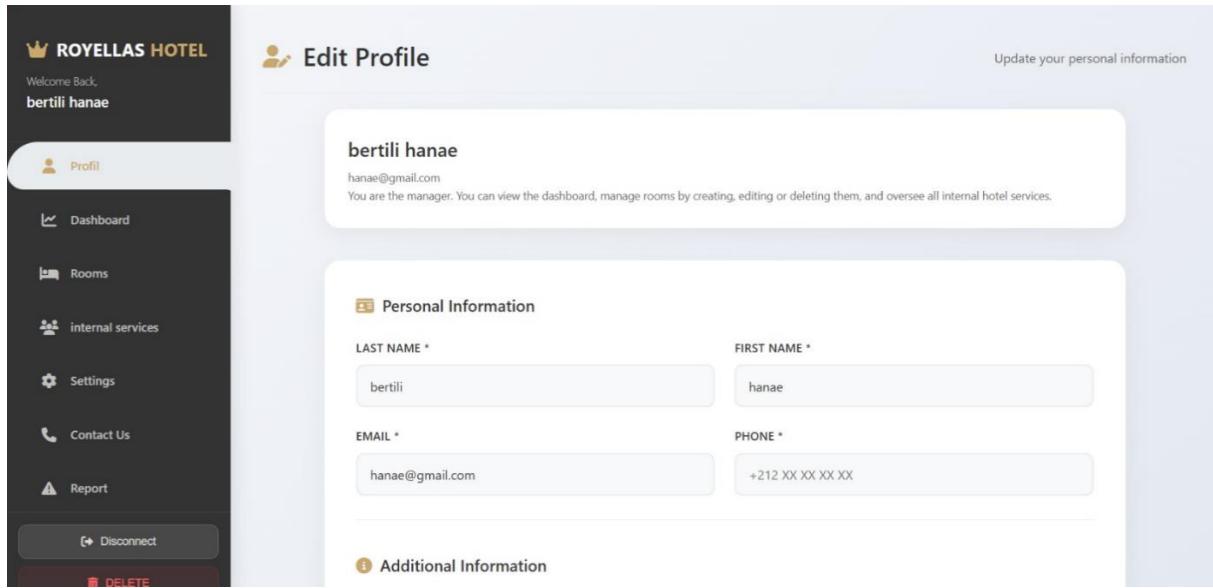


Figure 65 : Profile Manager

### Modifier le profil

Dans l’interface de modification du profil, le manager peut mettre à jour ses informations personnelles et paramètres de compte,

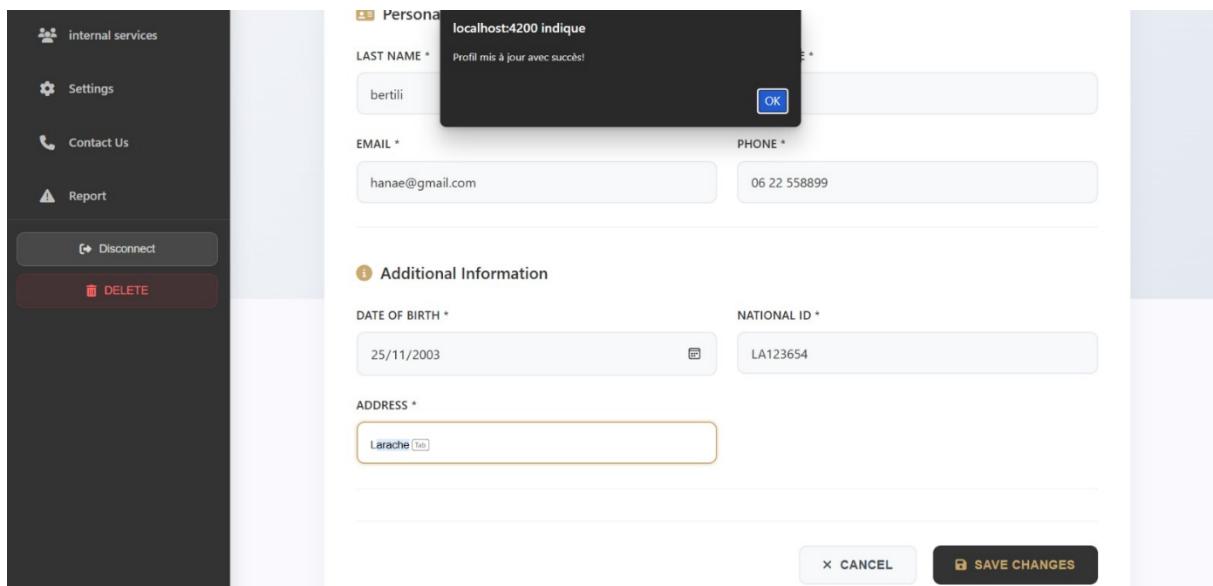


Figure 66 : Modifier le Profile Manager

#### 2.7.4. Gestion des Chambres

L’interface de gestion des chambres permet au personnel autorisé (manager) de superviser, créer et modifier les informations des chambres de l’hôtel à partir d’un seul espace centralisé.

##### 2.7.4.1. Liste des chambres :

Une interface présente l’ensemble des chambres avec leurs informations essentielles telles que le numéro, le type, le prix, le statut (Disponible, Occupée, Maintenance) et les actions possibles.

Number	Type	Price	Status	Description	Images	Rate	Bed (L x W)	Action
1	Double	5500 MAD	Disponible	A cozy double room with a comfortable bed.		80 %	200 x 180 cm	<button>Edit</button> <button>Delete</button>
2	Single	4500 MAD	Maintenance	A cozy single room with a comfortable single bed, perfect for one guest.		60 %	200 x 90 cm	<button>Edit</button> <button>Delete</button>
3	Suite	8500 MAD	Disponible	A luxurious suite with a spacious bedroom, a comfortable king-size bed, and a separate living area.		90 %	210 x 190 cm	<button>Edit</button> <button>Delete</button>
4	Double	6000 MAD	Occupée	A peaceful room with a relaxing design, featuring a balcony that overlooks the sea.		80 %	200 x 190 cm	<button>Edit</button> <button>Delete</button>

Figure 67 : Liste des chambres

#### 2.7.4.2. Crédation d'une nouvelle chambre :

Un formulaire permet d'ajouter une chambre au système, en renseignant le type, le prix, les caractéristiques et l'image de la chambre.

Figure 68 : formulaire pour Crée une nouvelle chambre

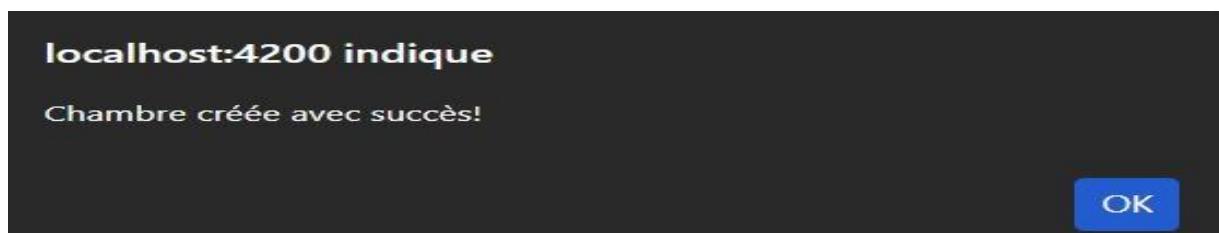


Figure 69 : chambre crée avec succès

#### **2.7.4.3. Modification d'une chambre existante :**

L'interface offre la possibilité de mettre à jour les informations d'une chambre existante, incluant le type, le prix, le statut et les équipements.

The screenshot shows a modal dialog box titled "localhost:4200 indique". Inside, a message says "Chambre modifiée avec succès!" with an "OK" button. The form fields include:

- Type: Suite
- Prix: 8900
- Statut: Disponible
- Description: A luxurious suite with Thai-inspired wooden décor, offering a cozy and eye-pleasing design.
- Surface: 90
- Nombre de lits: 210
- Nombre de personnes: 190
- Image: Room Image (button to choose file, currently empty)
- Preview: Room Image (empty placeholder)
- Action buttons: Update (yellow)

Figure 70 : formulaire pour modifier une chambre existante

#### **2.7.5. Gestion des services internes**

L'interface de gestion des services internes permet au manager de superviser et contrôler l'activité du personnel en temps réel. Chaque membre du personnel dispose d'un statut indiquant sa disponibilité :

- On Work : le personnel est actif et disponible pour accomplir ses tâches.
- Out Work : le personnel est inactif ou déconnecté, indiquant qu'il n'est pas disponible pour intervenir.

L'interface présente cette information sous forme de tableau clair avec les noms des employés, leur service et leur statut actuel. Cette visualisation permet au manager de planifier efficacement les interventions, de répartir les tâches et d'assurer un suivi précis de la disponibilité du personnel.

The screenshot shows the 'Internal Services' section of the application. At the top, it says 'Internal Services' and 'List of employees and their status (on\_work or out\_work)'. Below is a table with columns: Last Name, First Name, Email, Role, and Status. The data is as follows:

Last Name	First Name	Email	Role	Status
bertili	hanae	hanae@gmail.com	MANAGER	ON_WORK
baider	imane	imane@gmail.com	RECEPTIONNISTE	ON_WORK
el ghaylani	salwa	salwa@gmail.com	HOUSEKEEPING	ON_WORK
bonam	basma	basma@gmail.com	RECEPTIONNISTE	OUT_WORK

Figure 71 : L'interface de gestion des services internes

## 2.8. Espace Housekeeping

L'espace Housekeeping est conçu pour permettre au personnel de nettoyage de gérer efficacement les chambres et de suivre leur état. Il comprend plusieurs composants principaux

### 2.8.1. Page Profil

Le personnel peut consulter et mettre à jour ses informations personnelles et paramètres de compte, garantissant une gestion autonome et sécurisée de son profil professionnel.

The screenshot shows the 'Edit Profile' page for 'el ghaylani salwa'. At the top, it says 'Edit Profile' and 'Update your personal information'. Below is a summary box with the name 'el ghaylani salwa' and 'Housekeeping Staff'. The main form is divided into two sections: 'Personal Information' and 'Additional Information'.

**Personal Information:**

- LAST NAME \*: el ghaylani
- FIRST NAME \*: salwa
- EMAIL \*: salwa@gmail.com
- PHONE \*: 06 8965423

**Additional Information:**

- DATE OF BIRTH \*: 14/01/2002
- CIN \*: CA59874132

Figure 72 : L'interface profile Housekeeping

### Modifier profile

Dans l'interface de modification du profil, le Housekeeping peut mettre à jour ses informations personnelles et paramètres de compte,

Figure 73 : l'interface pour modifier profile Housekeeping

### 2.8.2. Dashboard :

Le tableau de bord fournit une vue globale de l'état des chambres, incluant les chambres disponibles, occupées, en maintenance ou nécessitant un nettoyage.



Figure 74 : Dashboard de Housekeeping

### 2.8.3. Chambres :

Chaque chambre est présentée avec ses informations essentielles (numéro, type, statut) et un bouton d'action « Clean », permettant au personnel de marquer la chambre comme nettoyée. Le statut de la chambre est automatiquement mis à jour dans le système, assurant une coordination fluide entre l'équipe Housekeeping et le reste du personnel hôtelier.

Cette interface permet au personnel de planifier et suivre efficacement les opérations de nettoyage, d'assurer la propreté des chambres et de maintenir un standard élevé de service client.

NUMBER	TYPE	STATUS	PRICE (MAD)	ACTIONS
#2	Single	Maintenance	4500 MAD	<button>Details</button>
#3	Suite	DISPONIBLE	8500 MAD	<button>Details</button>
#4	Double	OCCUPÉE	6000 MAD	<button>Clean</button> <button>Details</button>
#5	Suite	DISPONIBLE	7500 MAD	<button>Details</button>
#6	Single	Maintenance	4500 MAD	<button>Details</button>
#7	Suite	OCCUPÉE	9500 MAD	<button>Clean</button> <button>Details</button>
#8	Double	OCCUPÉE	5500 MAD	<button>Clean</button> <button>Details</button>
#9	Suite	DISPONIBLE	8900 MAD	<button>Details</button>

Figure 75 : L'interface chambre pour Housekeeping

## 2.9. Espace Admin

L'Espace Admin est conçu pour offrir une gestion complète et centralisée de l'ensemble des opérations hôtelières, permettant à l'administrateur de superviser l'activité quotidienne, de gérer les utilisateurs internes, et de contrôler les réservations et l'état des chambres.

### 2.9.1. Profil Admin

L'administrateur peut **consulter ses informations personnelles** (nom, coordonnées, etc.) afin de connaître son compte et son niveau d'accès dans le système.

Figure 76: L'interface de Profile Administrateur

### 2.9.2. Gestion des utilisateurs internes

Ce module permet de gérer les comptes des utilisateurs internes, tels que les managers et le personnel Housekeeping. Chaque compte inclut les informations personnelles de l'utilisateur, son rôle actuel et son état de connexion. L'administrateur peut également modifier le statut de

l'utilisateur entre « Actif » et « Inactif », permettant de contrôler l'accès au système selon les besoins.

ID	Email	Nom complet	Rôles	Statut	Actions
1	admin@example.com	Admin Admin	ADMIN	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
22	hanae@gmail.com	hanae bertili	MANAGER	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
23	zahra@gmail.com	zahra jbari	CLIENT	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
24	imane@gmail.com	imane baider	RÉCEPTIONNISTE	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
25	salwa@gmail.com	salwa el ghaylani	HOUSEKEEPING	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
26	basma@gmail.com	basma bonam	RÉCEPTIONNISTE	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
27	lamyae@gmail.com	lamyae lials	CLIENT	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>

Figure 77 : L'interface pour gérer les utilisateurs internes

## ✚ Mise à jour du statut

Dans le cadre de la gestion des utilisateurs internes, l'administrateur peut contrôler la disponibilité du personnel en modifiant le statut de chaque employé. Par exemple, le membre du personnel Salwa El Gailani a été mis en statut « Inactif », ce qui indique qu'elle n'est pas disponible pour effectuer ses tâches au sein de l'hôtel.

ID	Email	Nom complet	Rôles	Statut	Actions
1	admin@example.com	Admin Admin	ADMIN	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
22	hanae@gmail.com	hanae bertili	MANAGER	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
23	zahra@gmail.com	zahra jbari	CLIENT	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
24	imane@gmail.com	imane baider	RÉCEPTIONNISTE	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
25	salwa@gmail.com	salwa el ghaylani	HOUSEKEEPING	<span>✗ INACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
26	basma@gmail.com	basma bonam	RÉCEPTIONNISTE	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>
27	lamyae@gmail.com	lamyae lials	CLIENT	<span>✓ ACTIF</span>	<span>edit</span> <span>lock</span> <span>trash</span>

Figure 78 : Mise à jour le statut d'une utilisateur

Après le passage du statut de Salwa El Gailani à « Inactif », elle ne peut plus se connecter au système ni accéder à ses fonctionnalités. Cette mesure contribue à renforcer la sécurité de la

plateforme et garantit que seuls les utilisateurs actifs peuvent effectuer des opérations, assurant ainsi une gestion efficace des ressources humaines au sein de l'hôtel.

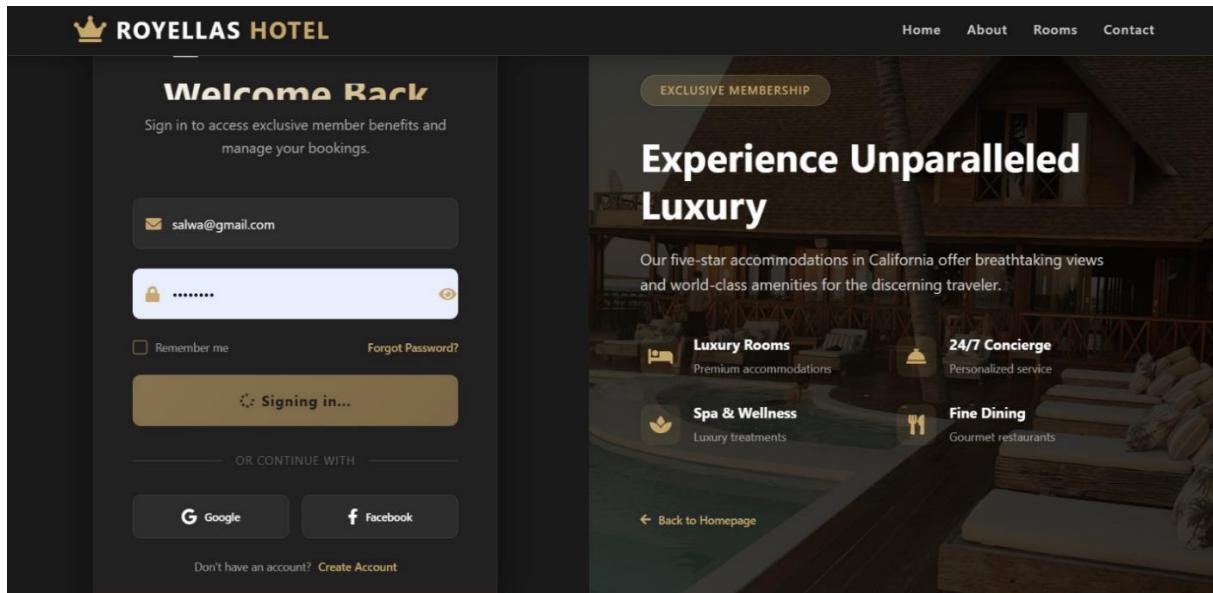


Figure 79 : seule une connexion active permet l'accès au système

### 2.9.3. Gestion des rôles et des permissions

L'interface de gestion des rôles et des permissions permet à l'administrateur de définir et de structurer les droits d'accès au sein du système hôtelier. Chaque rôle (Admin, Manager, Réceptionniste, Housekeeping, etc.) est associé à un ensemble précis de permissions déterminant les fonctionnalités accessibles.

L'administrateur peut créer, modifier ou supprimer des rôles, ainsi qu'attribuer ou révoquer des permissions spécifiques, garantissant ainsi une séparation claire des responsabilités, une sécurisation des accès, et une gestion flexible des profils utilisateurs selon les besoins organisationnels.

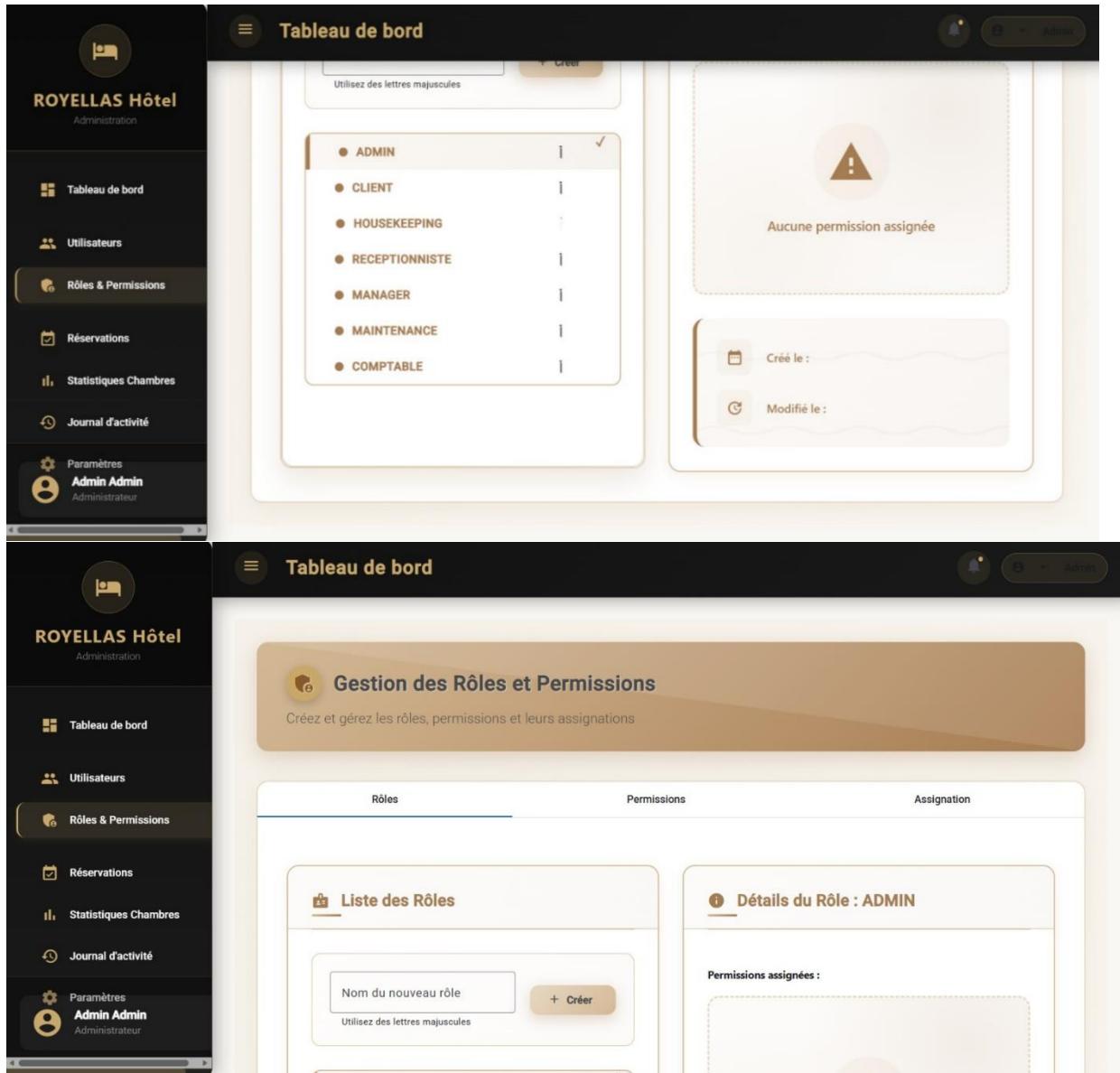


Figure 80 : L'interface pour gérer les rôles et les permissions

### ✚ Création d'un rôle : « Gatekeeper »

Dans l'interface de gestion des rôles et des permissions, l'administrateur peut créer de nouveaux rôles afin de structurer l'accès aux fonctionnalités du système. À titre d'exemple, un rôle nommé « Gatekeeper » a été créé pour gérer des permissions spécifiques liées à la supervision des entrées et sorties dans l'hôtel.

Une fois le rôle créé, le système affiche un résultat confirmant la création réussie, incluant le nom du rôle et les permissions associées. Cette fonctionnalité permet de formaliser la hiérarchie des accès et d'assurer une gestion sécurisée et claire des responsabilités.

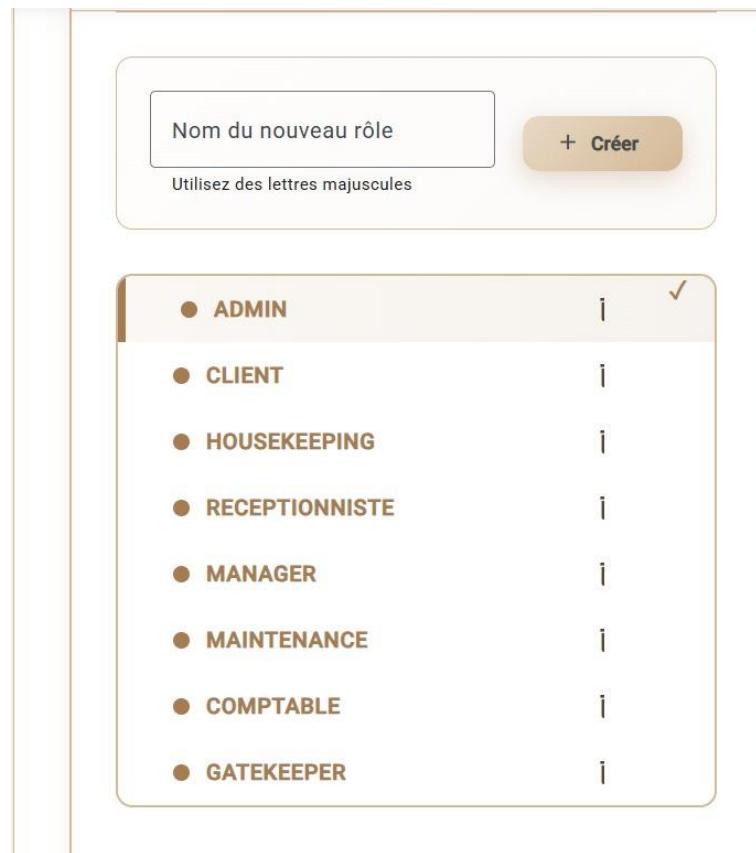


Figure 81 : Crédit d'un rôle

#### ✚ Création d'une permission : « Open Door »

Dans l'interface de gestion des permissions, l'administrateur peut créer de nouvelles permissions correspondant à des actions spécifiques au sein de l'hôtel. Par exemple, la permission « Open Door » a été créée pour permettre aux utilisateurs assignés d'ouvrir certaines portes ou accès sécurisés.

Figure 82 : Crédit d'une permission

## Attribution de la permission

Une fois la permission « Open Door » créée, l'administrateur peut l'associer à un rôle spécifique afin de définir clairement les actions autorisées pour les utilisateurs qui possèdent ce rôle.

Dans ce cas, la permission « Open Door » a été attribuée au rôle « Gatekeeper », permettant ainsi aux utilisateurs ayant ce rôle d'accéder aux fonctionnalités sécurisées liées à l'ouverture des portes.

Le système affiche un résultat confirmant l'attribution réussie, indiquant le rôle concerné et la permission associée. Cette étape garantit une gestion sécurisée et contrôlée des droits d'accès, en respectant la hiérarchie des rôles et des responsabilités au sein de l'hôtel.

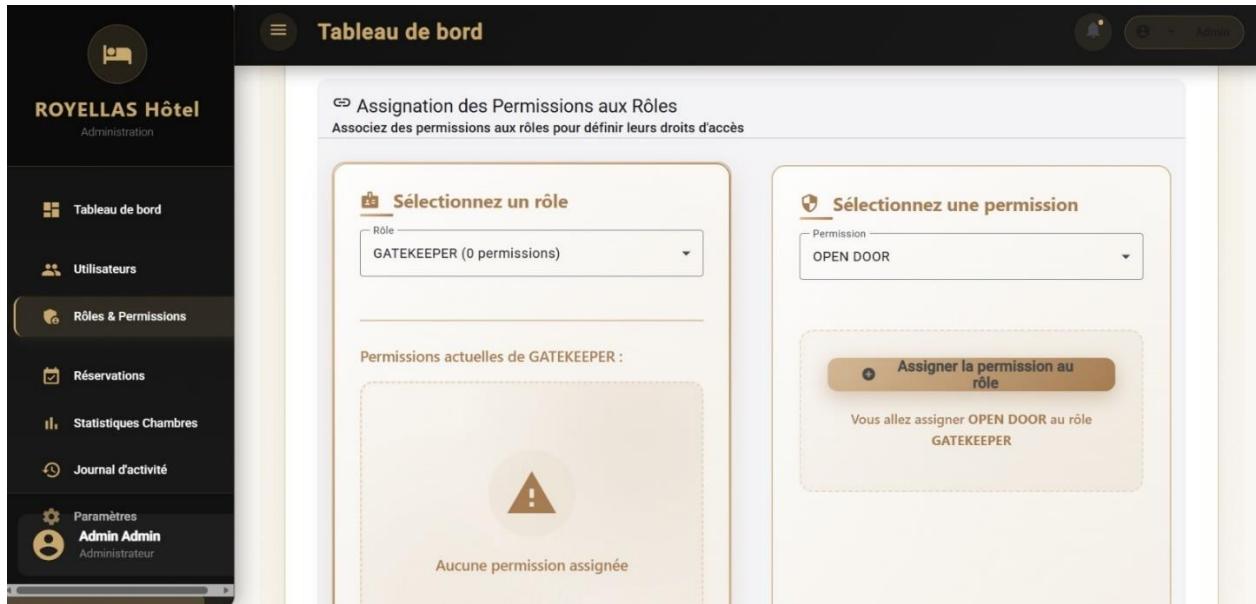


Figure 83 : Étape d'Assignment d'une Permission à un Rôle (GATEKEEPER)

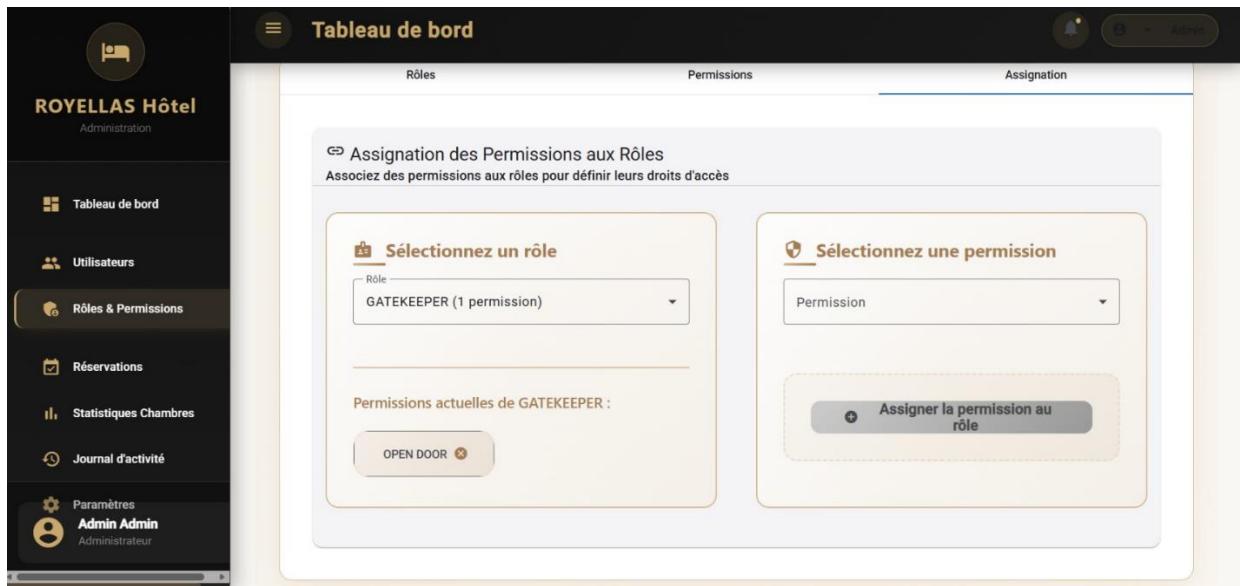


Figure 84 : Confirmation de l'Assignment – Permission 'OPEN DOOR' au Rôle GATEKEEPER

#### 2.9.4. Visualisation globale des réservations

L'interface de l'Espace Admin permet à l'administrateur d'avoir une vision complète de toutes les réservations effectuées dans l'hôtel.

- Liste des réservations : Toutes les réservations sont affichées dans un tableau structuré, comprenant les informations suivantes : nom du client, type de chambre, dates de séjour, statut de la réservation (Pending, Confirmed, Rejected), et montant total.
- Diagrammes et visualisations graphiques : L'interface propose des graphes et diagrammes permettant de visualiser rapidement la répartition des réservations par

statut et par type de chambre, ainsi que l'évolution de l'occupation au fil du temps. Ces outils facilitent l'analyse des tendances, la planification des ressources et l'optimisation de l'occupation de l'hôtel.

Cette fonctionnalité offre à l'administrateur une gestion stratégique des réservations, en combinant données détaillées et visualisations graphiques pour soutenir la prise de décision et assurer un suivi efficace du service client.

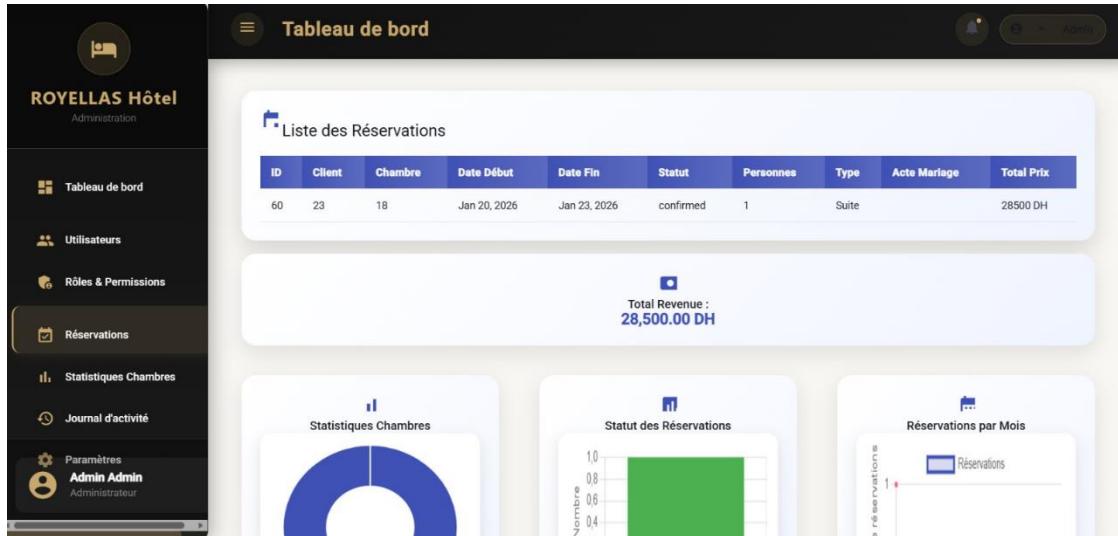


Figure 85 : Tableau de Bord Administrateur – Liste et Statistiques des Réservations

## 2.9.5. Visualisation globale des chambres

L'interface de l'Espace Admin permet à l'administrateur d'avoir une vision complète et statistique de toutes les chambres de l'hôtel.

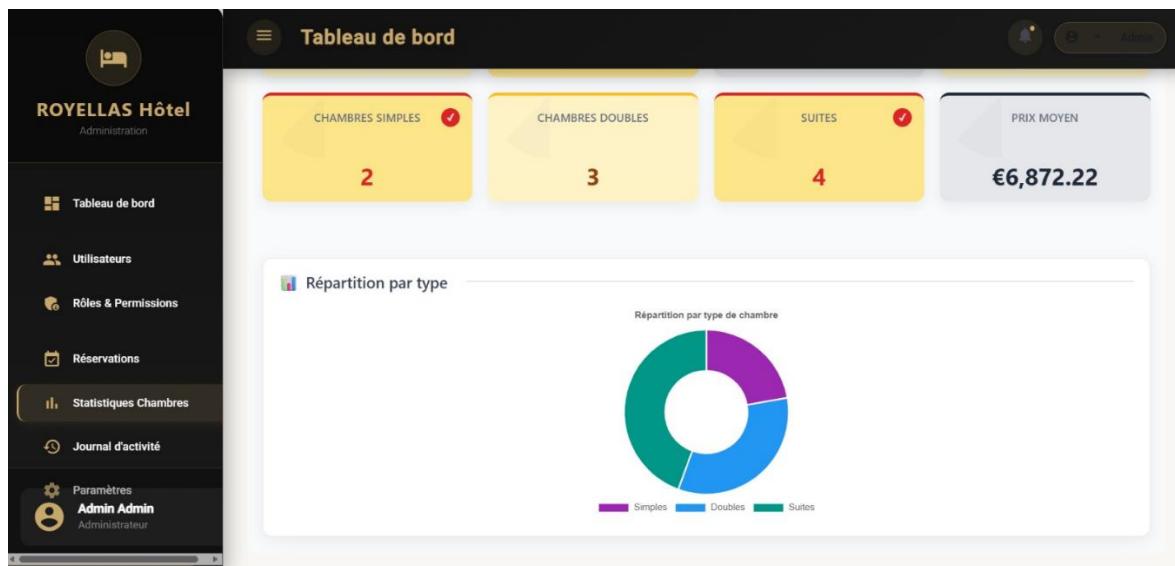


Figure 86 : Tableau de Bord Administrateur – Liste et Statistiques des Réservations

# Conclusion

Au terme de ce projet, nous pouvons affirmer que le système ROYELLA SHOTEL constitue une solution complète, moderne et performante pour la gestion hôtelière. Ce travail nous a permis de mettre en pratique l'ensemble des concepts théoriques étudiés durant notre formation en Master DevOps et Cloud Computing, tout en répondant à une problématique réelle et concrète du secteur hôtelier.

À travers la conception et la réalisation de cette application, nous avons adopté une architecture orientée microservices, permettant une meilleure modularité, une forte évolutivité et une maintenance facilitée. L'utilisation des principes du Domain-Driven Design (DDD) a joué un rôle clé dans la structuration des domaines métiers, en assurant une séparation claire des responsabilités et une cohérence fonctionnelle entre les différents services du système.

Sur le plan technique, le choix des technologies telles que Spring Boot, Flask, Node.js, Angular, ainsi que l'intégration des outils DevOps comme Docker, Kubernetes, Jenkins, SonarQube et les pipelines CI/CD, nous a permis de mettre en place une plateforme robuste, sécurisée et scalable. Ces choix garantissent une haute disponibilité du système, une automatisation des déploiements et une amélioration continue de la qualité du code.

Le système développé offre une gestion efficace des réservations, des chambres, des paiements et des services internes, tout en améliorant la coordination entre les différents acteurs de l'hôtel (réceptionnistes, manager, housekeeping, maintenance, comptabilité et administration). De plus, l'interface utilisateur intuitive permet aux clients de gérer leurs réservations et paiements de manière autonome, contribuant ainsi à une meilleure expérience utilisateur.

Ce projet nous a également permis de renforcer nos compétences techniques, organisationnelles et collaboratives. Le travail en équipe, la gestion des tâches via des outils de planification, ainsi que la résolution des problèmes techniques rencontrés tout au long du développement, ont constitué une expérience enrichissante et formatrice.

En conclusion, le projet ROYELLA SHOTEL représente une application aboutie, alignée avec les standards actuels du génie logiciel et du DevOps, et constitue une base solide pour des développements futurs dans le domaine des systèmes de gestion hôtelière modernes.

# Perspective

Dans la continuité de ce travail, plusieurs perspectives d'évolution peuvent être envisagées afin d'améliorer davantage le système ROYELLA SHOTEL, tout en restant cohérentes avec l'architecture microservices et l'approche DevOps adoptées.

Tout d'abord, une amélioration possible concerne le renforcement des mécanismes de monitoring et de logging des microservices. L'intégration d'outils de supervision permettrait de mieux suivre l'état des services, d'analyser les performances et de détecter rapidement les anomalies, ce qui est essentiel dans une architecture distribuée.

Ensuite, l'optimisation des pipelines CI/CD représente une perspective importante. Il serait possible d'enrichir les pipelines existants en intégrant davantage de tests automatisés (tests unitaires et tests d'intégration), afin d'assurer une meilleure fiabilité du code avant chaque déploiement et de réduire les risques en environnement de production.

Une autre perspective concerne l'amélioration de la gestion de la configuration et des variables d'environnement des microservices. L'utilisation de solutions centralisées permettrait de simplifier la gestion des paramètres de configuration et d'assurer une meilleure cohérence entre les différents environnements (développement, test et production).

Par ailleurs, le système pourrait être renforcé au niveau de la sécurité, notamment par une gestion plus avancée de l'authentification et de l'autorisation entre les services, afin de sécuriser davantage les échanges internes et de garantir un meilleur contrôle des accès.

De plus, l'intégration de l'intelligence artificielle pour le développement d'un chatbot intelligent destiné à la fois aux clients et aux services internes constitue une perspective majeure. Ce chatbot permettrait d'améliorer l'expérience utilisateur en fournissant des réponses instantanées, en facilitant les réservations et en assistant le personnel dans l'accès rapide aux informations et aux services internes.

Enfin, une perspective à long terme consiste à améliorer la scalabilité du système en exploitant davantage les capacités offertes par l'orchestration des conteneurs. Cela permettrait d'adapter automatiquement les ressources en fonction de la charge et d'assurer une meilleure disponibilité des services.

Ces perspectives s'inscrivent dans une logique d'amélioration continue et visent à consolider les acquis du projet, tout en ouvrant la voie à des évolutions futures réalisables et adaptées à une architecture microservices orientée DevOps.

# Webographie :

**Spring Boot – Documentation officielle**

<https://spring.io/projects/spring-boot>

**Spring Cloud – Microservices & Discovery**

<https://spring.io/projects/spring-cloud>

**Docker – Documentation officielle**

<https://docs.docker.com>

**Kubernetes – Documentation officielle**

<https://kubernetes.io/docs>

**Jenkins – Documentation officielle**

<https://www.jenkins.io/doc>

**SonarQube – Documentation officielle**

<https://docs.sonarsource.com>

**Angular – Documentation officielle**

<https://angular.io/docs>

**Node.js – Documentation officielle**

<https://nodejs.org/en/docs>

**Flask – Documentation officielle**

<https://flask.palletsprojects.com>

**Microservices Architecture – Martin Fowler**

<https://martinfowler.com/microservices>

**Domain-Driven Design (DDD) – Martin Fowler**

<https://martinfowler.com/tags/domain%20driven%20design.html>

**DevOps Concepts and Practices – Red Hat**

<https://www.redhat.com/en/topics/devops>

**CI/CD Concepts – Atlassian**

<https://www.atlassian.com/continuous-delivery>

**REST API Best Practices**

<https://restfulapi.net>

**Git & Version Control – Git Documentation**

<https://git-scm.com/doc>

Newman, S.

*Building Microservices: Designing Fine-Grained Systems.*

O'Reilly Media, 2015.

Evans, E.

*Domain-Driven Design: Tackling Complexity in the Heart of Software.*

Addison-Wesley, 2003