REPUBLIQUE DU CAMEROU
****
PAIX - TRAVAIL – PATRIE
****
UNIVERSITÉ DE YAOUNDÉ
Faculté des Sciences
Département d'Informatiqu
B.P. 812 Yaoundé

REPUBLIC OF CAMEROON
****
PEACE-WORK-FATHERLAND
****
UNIVERSITY OF YAOUNDÉ I
Faculty of Sciences
Department of Computer Science
P.O. Box 812 Yaoundé

MEMOIRE PROFESSIONNEL

Présenté en vue de l'obtention du diplôme de Maîtrise professionnelle

Filière : **Système d'Information et Génie Logiciel**

THEME :

MISE EN PLACE D'UN SYSTEME DE GESTION DES SERVICES TOURISTIQUES POUR PARC NATIONAL ZAKOUMA

Rédigé et soutenu par :

MOUSSA ABAKAR ABBAZENE : 23V2834

Sous l'encadrement de :

Dr AMINOU HALIDOU

Année Académique : 2024-2025

Table des matières

Chapitre 1 : Introduction

0.0.1 Le Parc National de Zakouma

Chapitre 2 : Cadre institutionnel du stage

Le Parc National de Zakouma

Le Parc National de Zakouma, situé dans le sud-est du Tchad, est l'un des joyaux de la conservation de la biodiversité en Afrique. Créé en 1963, ce parc couvre une superficie d'environ 3 000 km² et est administré par l'organisation African Parks depuis 2010. Il se distingue par une faune riche et variée, comprenant notamment des éléphants, des girafes, des lions et de nombreuses espèces d'oiseaux. En plus de sa mission de protection de la biodiversité, le parc joue un rôle économique et social majeur grâce à ses activités touristiques, qui attirent des visiteurs du monde entier.

Le département touristique du parc, où s'est déroulé mon stage, est chargé de l'accueil des touristes, de la gestion des hébergements, des excursions et des services liés à la restauration. Ce département est un pilier important de la structure, car il contribue à la sensibilisation sur les enjeux environnementaux tout en générant des revenus pour la gestion durable du parc.

2.2 Historique du Parc National de Zakouma

Depuis sa création, le Parc National de Zakouma a connu plusieurs phases marquantes. Les premières décennies ont été consacrées à l'aménagement et à la mise en place de politiques de conservation. Cependant, des défis tels que le braconnage et la déforestation ont menacé l'écosystème du parc, entraînant une diminution alarmante de certaines espèces.

L'arrivée d'African Parks en 2010 a marqué un tournant dans la gestion du parc. Cette organisation a instauré des mesures rigoureuses de protection, réhabilitant ainsi les populations animales et rétablissant la sécurité dans la région. Aujourd'hui, le parc est un exemple de réussite en matière de conservation et de développement touristique durable, offrant des infrastructures modernes pour accueillir les visiteurs tout en préservant l'écosystème.

2.3 Rappel de l'objectif

L'objectif principal de mon stage était de contribuer à la conception et à la mise en œuvre d'un **système de gestion numérique des services touristiques** pour améliorer l'efficacité des opérations du département touristique. Ce système vise à centraliser la gestion des réservations, la gestion des touristes, les commandes pour le restaurant et la publication des actualités concernant la faune et la flore.

Ce projet a pour finalité d'apporter une solution innovante aux défis rencontrés dans la gestion quotidienne, tels que la dispersion des informations, la lenteur des processus manuels et le besoin de mieux interagir avec les touristes et les visiteurs potentiels.

2.4 Bilan des activités réalisées

Au cours de mon stage, plusieurs résultats ont été obtenus, parmi lesquels :

— **Analyse des besoins** : Une étude approfondie a été réalisée pour identifier les besoins fonctionnels et techniques du système.

— **Conception de l'application** : Une architecture du système a été élaborée, incluant des modules pour la gestion des réservations, la gestion des commandes et un espace dédié aux actualités.

— **Développement partiel du système** : Des fonctionnalités telles que la gestion des réservations et l'interface utilisateur pour les touristes ont été développées et testées.

— **Documentation technique** : Une documentation complète a été rédigée pour faciliter la maintenance et l'évolution du système.

Ces résultats constituent une base solide pour la finalisation et la mise en production de l'application.

2.5 Perspectives

Pour l'avenir, plusieurs perspectives peuvent être envisagées :

— **Finalisation du développement** : Terminer les modules en cours et procéder au déploiement du système.

— **Formation du personnel** : Assurer une formation des équipes sur l'utilisation de l'application afin de garantir une adoption optimale.

— **Améliorations continues** : Intégrer des fonctionnalités supplémentaires, telles que des outils analytiques pour le suivi des performances et des retours clients.

— **Extension des services** : Étendre l'utilisation du système à d'autres domaines, comme la gestion des hébergements ou l'organisation des excursions.

Ces perspectives permettront au Parc National de Zakouma de se positionner comme un leader dans la gestion numérique des services touristiques en Afrique, tout en renforçant son impact positif sur la conservation et le développement local.

2.6 Localisation

Le Parc National de Zakouma est situé dans la région du Salamat, dans le sud-est du Tchad, à environ 800 km de la capitale N'Djamena. Il s'étend sur une superficie de 3 000 km², au cœur d'un écosystème riche qui inclut des plaines, des savanes et des zones humides saisonnières. Le parc est accessible par voie terrestre et aérienne, avec une piste d'atterrissage permettant de recevoir des vols touristiques.

The administrative seat of the park is located in the same location in Zakouma, and the tourist infrastructure (lodges, restaurant, center of information) are located nearby to facilitate visitor arrival.


Tamanrasset
تامنراست
Niger
Chad
Sudan
Khartoum
الخرطوم
Al Fashir
الفاشر
N'Djamena
إنجامينا
Kano
Maiduguri
Zakouma National Park
Nigeria
South Sudan
Central
African
Republic
Cameroon
Port Harcourt
Douala
Yaoundé
Bangui
Juba
Google
Map data ©2025 Mapa GISrael, Google

2.7 Mission du Parc National de Zakouma

Le Parc National de Zakouma, situé au Tchad, est l'un des plus grands et plus anciens parcs nationaux d'Afrique centrale. Il joue un rôle essentiel dans la protection de la biodiversité, le développement durable et la valorisation touristique du pays.

2.7.1 Protection de la faune et de la flore

L'une des principales missions du parc est de protéger les espèces animales menacées, comme les éléphants, les girafes, les antilopes, les lions et de nombreux oiseaux. Grâce à des efforts de surveillance renforcés et à la lutte contre le braconnage, Zakouma est devenu un refuge sécurisé pour ces animaux. La flore, composée de savanes, forêts sèches et plaines, est également protégée pour maintenir l'équilibre écologique.

2.7.2 Conservation de l'environnement

Le parc œuvre pour la préservation des écosystèmes naturels. Cela passe par la gestion durable des ressources, la surveillance des zones sensibles et l'implication des communautés locales dans les projets de restauration des habitats. Le parc lutte aussi contre les feux de brousse et la déforestation.

2.7.3 Développement local et éducation

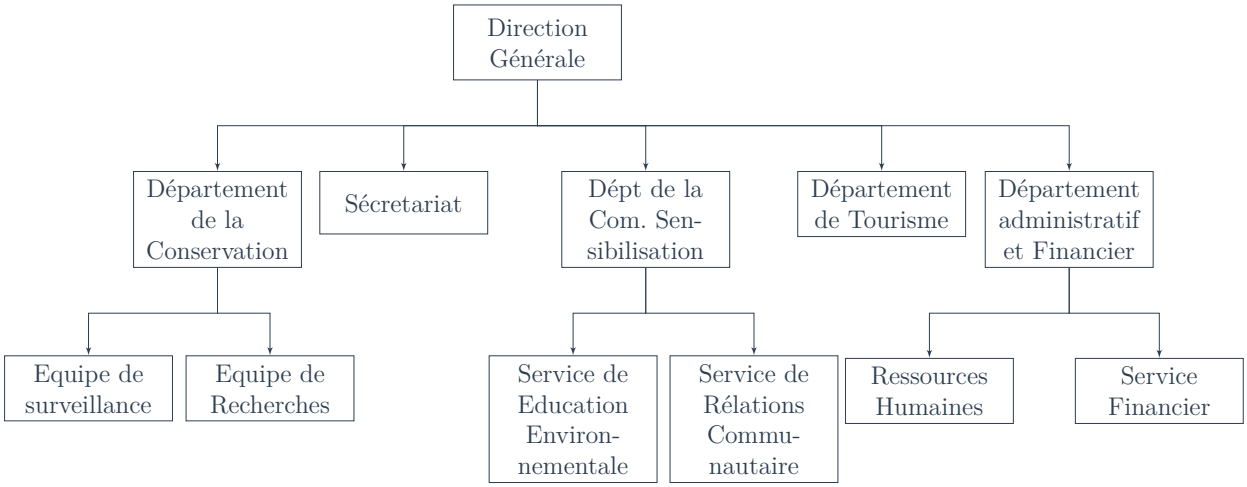
Zakouma soutient les communautés locales en créant des emplois, en construisant des écoles, et en favorisant des projets agricoles durables. Des programmes d'éducation environnementale sont aussi organisés pour sensibiliser les jeunes et les adultes à l'importance de la nature.

2.7.4 Accueil des touristes et valorisation touristique

Le parc joue un rôle clé dans la promotion du tourisme écologique. Chaque année, il attire des touristes du monde entier, passionnés par la nature et les safaris. Les visiteurs peuvent découvrir des paysages magnifiques, observer des animaux sauvages dans leur milieu naturel, et apprendre sur la biodiversité africaine. Le tourisme apporte des ressources financières importantes, qui sont réinvesties dans la gestion du parc et dans le développement des villages voisins.

2.8 Organigramme du Parc National de Zakouma

Le Parc National de Zakouma est organisé en plusieurs départements, chacun ayant des responsabilités spécifiques pour assurer la bonne gestion du parc. Voici l'organigramme présentant la hièrarchie de l'Institution :



Mon stage a été réalisé au sein du **Service Touristique**, en collaboration avec les autres départements pour collecter les informations nécessaires à la conception du système.

2.9 Déroulement du stage et tâches effectuées

Mon stage s'est déroulé sur une période de xxxxxxx, selon les étapes suivantes :

Phase 1 : Prise de contact et découverte de la structure

— Participation à des réunions d'introduction pour comprendre le fonctionnement global du parc et ses défis touristiques.

— Visites sur le terrain pour observer les interactions entre les équipes et les touristes.

Phase 2 : Analyse des besoins et conception du système

— Réalisation d'une étude des besoins fonctionnels à travers des entretiens avec les responsables du service touristique.

— Rédaction des spécifications fonctionnelles et techniques de l'application.

— Création de maquettes de l'interface utilisateur pour les différents modules (réservations, gestion des commandes, actualités).

Phase 3 : Développement et tests

— Développement des fonctionnalités principales de l'application :

    - Gestion des Utilisateurs ;
    - Gestion des chambres ;
    - Gestion des réservations ;
    - Gestion des check-in et check-out
    - Gestion des paiements et facturation en ligne
    - Gestion de Service Restauration
    - Gestion des chatBot
    - Publication des actualités.
- Tests unitaires et validation des modules développés.

Phase 4 : Documentation et formation

- Rédaction d'un manuel d'utilisation pour le personnel.
- Organisation d'ateliers pour présenter l'application et former les utilisateurs.

Ce déroulement m'a permis de contribuer activement à un projet concret tout en acquérant des compétences techniques et organisationnelles liées à mon domaine d'études.

Chapitre 3 : Tourisme et technologies de gestion : fondements théoriques et travaux existants

Introduction

Le présent chapitre s'inscrit dans la continuité du chapitre précédent, qui a présenté l'architecture de la structure d'accueil du Parc National de Zakouma. Il a pour objectif d'explorer l'état de l'art des principaux services numériques intervenant dans la gestion touristique : la réservation en ligne, le paiement en ligne, les chatbots à intelligence artificielle, les services de restauration numérique, le reporting et la gestion des utilisateurs. À travers cette revue de la littérature, nous identifierons les concepts clés, les évolutions historiques et les définitions opérationnelles nécessaires à la conception de notre système.

Ce panorama permettra de situer chaque technologie dans son contexte applicatif et historique, fournir un vocabulaire commun (définitions), mettre en évidence les apports et limites des travaux existants, préparer la synthèse critique qui guidera le choix d'architectures et de méthodes pour le système de gestion des services touristiques de Zakouma.

3.1 Contextualisation du Tourisme et des Technologies de Gestion

Dans cette section, nous clarifions d'abord les notions fondamentales relatives à notre thème, avant de retracer leur évolution historique.

3.1.1 Concepts Clés du Tourisme et des Technologies

Système de gestion des services touristiques : Le système de gestion des services touristiques constitue une plateforme intégrée regroupant l'ensemble des modules logiciels nécessaires pour faciliter les interactions entre les touristes et les prestataires du parc. Il centralise les fonctionnalités essentielles telles que la réservation, le paiement, la restauration et l'assistance, offrant ainsi une expérience utilisateur fluide et cohérente. En automatisant ces processus, le système permet d'optimiser la gestion des ressources tout en améliorant la satisfaction des visiteurs. Son architecture modulaire assure une grande flexibilité, facilitant l'ajout de nouvelles fonctionnalités ou l'adaptation à des besoins spécifiques.

Réservation en ligne : La réservation en ligne représente un processus entièrement dématérialisé permettant aux touristes d'enregistrer et de confirmer leurs prestations, qu'il s'agisse d'hébergement, d'activités ou de location de véhicules. Accessible via une interface web ou mobile, cette fonctionnalité offre un gain de temps significatif en évitant les démarches physiques ou les échanges téléphoniques. Les utilisateurs peuvent consulter les disponibilités en temps réel, choisir leurs options et recevoir une confirmation immédiate. Cette automatisation réduit également les risques d'erreurs humaines et améliore l'efficacité opérationnelle du parc.

Paiement en ligne : Le paiement en ligne repose sur un échange sécurisé de données financières, incluant les cartes bancaires ou les porte-monnaies électroniques. Les transactions sont protégées par des protocoles de chiffrement avancés, garantissant l'authenticité, la confidentialité et l'intégrité des données. Ce système permet aux touristes de régler leurs prestations en toute confiance, sans crainte de fraude ou de fuite d'informations. De plus, il simplifie la gestion des remboursements et des annulations, tout en offrant une traçabilité complète des opérations financières pour les administrateurs du parc.

Chatbot à IA : Le chatbot à intelligence artificielle est un agent conversationnel automatisé conçu pour interagir avec les utilisateurs de manière naturelle et intuitive. Grâce à des algorithmes de traitement du langage naturel (NLP) et de génération de texte, il est capable de comprendre les requêtes des touristes et d'y répondre avec précision. Que ce soit pour fournir des informations sur les activités du parc, aider à une réservation ou répondre à des questions fréquentes, le chatbot améliore l'expérience utilisateur tout en réduisant la charge de travail des équipes d'assistance. En cas de demande complexe, il peut rediriger vers un agent humain, assurant ainsi une prise en charge complète.

Service de restauration numérique : Le service de restauration numérique est une solution logicielle dédiée à la prise de commande et à la facturation des repas. Intégré à un système de point de vente (POS) ou à une application mobile/tablette, il permet aux visiteurs de passer leurs commandes de manière autonome, réduisant ainsi les temps d'attente. Les menus sont mis à jour en temps

réel, et les options de paiement sont directement liées au système de paiement en ligne. Ce service optimise également la gestion des stocks et des préférences alimentaires, offrant une expérience personnalisée aux clients tout en simplifiant les opérations pour le personnel.

Service de reporting : Le service de reporting est un composant clé pour l'analyse et la restitution des données opérationnelles. Il génère des rapports détaillés, des tableaux de bord interactifs et des indicateurs clés de performance (KPI), permettant aux gestionnaires du parc de suivre l'évolution des activités et de prendre des décisions éclairées. Ces outils fournissent des insights sur les tendances de réservation, les préférences des visiteurs ou les performances financières, facilitant ainsi une gestion proactive et stratégique des ressources.

Service de gestion des utilisateurs : Le service de gestion des utilisateurs est un module central pour l'administration des comptes. Il prend en charge l'inscription, l'authentification, l'autorisation et la gestion des rôles et droits d'accès. Grâce à ce système, les profils des touristes et des administrateurs sont sécurisés, et les accès aux différentes fonctionnalités sont strictement contrôlés. Ce module permet également de personnaliser l'expérience utilisateur en sauvegardant les préférences et l'historique des interactions, tout en assurant la conformité aux réglementations en matière de protection des données.

3.1.2 Évolution Historique des Technologies dans le Secteur Touristique

Les années 1990 ont marqué l'émergence des premières plateformes de réservation en ligne, avec des acteurs pionniers comme Bookings.com (fondé en 1996). Ces solutions ont révolutionné le secteur touristique en permettant aux utilisateurs d'effectuer des réservations d'hôtels directement via Internet, remplaçant progressivement les méthodes traditionnelles basées sur le téléphone ou les agences physiques. Cette transition vers des interfaces web a posé les bases de la digitalisation du secteur, offrant une plus grande transparence et accessibilité aux voyageurs.

À la fin des années 1990, le paiement en ligne a connu un essor significatif avec l'apparition de solutions innovantes telles que PayPal (1998) et le développement de protocoles de sécurité comme le 3D Secure (2001). Ces avancées ont permis de standardiser les transactions financières sur le Web, en garantissant des échanges sécurisés et fiables. L'adoption généralisée de ces technologies a joué un rôle clé dans la croissance du commerce électronique et des services en ligne, en renforçant la confiance des consommateurs.

L'histoire des chatbots remonte aux années 1960, avec des prototypes pionniers comme ELIZA (1966), un programme simulant une conversation humaine basée sur des règles prédéfinies. Au fil des décennies, les chatbots ont évolué, passant de

systèmes simples à des solutions plus sophistiquées intégrant l'intelligence artificielle. À partir de 2016, l'avènement des modèles de type "encodeur-décodeur" et des architectures Transformer (comme GPT) a marqué un tournant, permettant des interactions plus naturelles et contextuelles, et ouvrant la voie aux assistants virtuels modernes.

Les années 2000 ont vu le déploiement des premiers systèmes de point de vente (POS) informatisés dans le secteur de la restauration. Cette décennie a également été témoin de l'arrivée des applications mobiles de commande après 2010, ainsi que des bornes interactives (self-ordering kiosks). Ces innovations ont transformé l'expérience client en réduisant les temps d'attente et en offrant une plus grande autonomie, tout en optimisant la gestion des commandes pour les établissements.

Durant les années 2000 et 2010, les pratiques de reporting ont évolué des rapports papier vers des solutions de Business Intelligence (BI) centralisées, comme SAP BI (2005). Ces outils ont introduit l'automatisation des tableaux de bord et des analyses en temps réel, permettant aux entreprises d'exploiter plus efficacement leurs données. Cette transition a facilité la prise de décision stratégique en fournissant des insights précis et actualisés.

Entre les années 1990 et 2020, la gestion des utilisateurs a connu une transformation majeure, passant des annuaires LDAP aux systèmes modernes d'IAM (Identity and Access Management). Ces solutions intègrent désormais des protocoles avancés comme OAuth 2.0, OpenID Connect et la fédération d'identité, offrant une sécurité renforcée et une gestion simplifiée des accès. Cette évolution reflète l'importance croissante de la protection des données et de l'expérience utilisateur dans un monde de plus en plus connecté.

Cette mise au point pose les bases conceptuelles et historiques indispensables pour aborder, dans les sections suivantes, l'analyse détaillée de chaque service et la synthèse critique de la littérature.

3.2 Réservation en ligne

La réservation en ligne constitue l'un des piliers des systèmes modernes de gestion touristique. Comme le démontre l'étude de Geert-Jan Bruinsma (2007), les plateformes centralisées permettent de traiter des volumes importants (100M réservations/an) grâce à des interfaces conviviales et une large gamme d'hébergements. Cependant, cette approche présente une dépendance critique aux avis clients et des commissions élevées.

Les travaux d'Ankor et al. (2019) sur les portails web synchronisés révèlent une augmentation de 35% du taux de réservation dans l'hôtellerie, avec une visibilité accrue pour les établissements. Leur modèle, testé sur 1 000 réservations pilotes, souligne néanmoins la nécessité de former le personnel à ces outils.

L'étude de Barua & Kaiser (2024) introduit une innovation majeure : l'intégration de l'IA pour la prédiction de demande (92% de précision). Bien que scalable, cette solution complexifie l'architecture système en raison de son recours aux microservices – un défi technique pertinent pour le Parc National Zakouma, où la scalabilité est essentielle pour gérer les variations saisonnières.

Si la réservation en ligne optimise l'accès aux services, son efficacité repose sur des mécanismes de paiement sécurisés, comme l'illustrent les développements récents en fintech touristique.

3.3 Paiement en ligne

L'évolution des solutions de paiement en ligne a été marquée par deux approches complémentaires. L'étude de Michael Kenny et Robert Rosenstein sur la tarification dynamique démontre que l'optimisation des flux financiers réduit de 15% le délai de réservation, tout en offrant des interfaces multilingues adaptées au tourisme international.

Les recherches de Do et al. (2023) mettent en lumière l'essor des e-wallets dans les zones reculées, avec un taux de réussite de 98% sur 120 transactions testées. Leur solution, bien que nécessitant une couverture réseau stable (3s/tx en moyenne), répond aux défis des paiements sécurisés en contexte africain, où les infrastructures bancaires traditionnelles sont limitées.

Ces travaux convergent vers un impératif : l'intégration de protocoles de chiffrement (comme le 3D Secure cité dans l'historique) pour garantir à la fois rapidité et sécurité, comme le préconise Dubois & Martin (2019) dans leur analyse des applications mobiles touristiques.

Au-delà des transactions financières, l'automatisation des interactions avec les touristes passe par des outils conversationnels avancés, où l'IA joue un rôle croissant.

3.4 Chatbot avec IA

Les chatbots intelligents représentent une révolution dans la relation client touristique. Shrestha et al. (2024) ont démontré l'efficacité des modèles hybrides (IA collaborative + analyse de contenu), atteignant 82% de précision dans la recommandation d'activités grâce à l'analyse de 50 000 avis TripAdvisor.

Cette approche rejoint les conclusions de Barua & Kaiser (2024) sur la nécessité de traiter des volumes massifs de données pour des réponses en temps réel – un défi technique que le Parc Zakouma devra résoudre via une base vectorielle robuste, comme suggéré dans l'architecture du chapitre 4.

Cependant, l'étude de Nkosana et al. (2016) rappelle une limite cruciale : la dépendance à la connectivité internet. Dans un parc national où le réseau peut être intermittent, une solution hybride (couplant IA et relais humains) s'impose, comme le préconise le cadre théorique de Dubois & Martin (2019) pour les zones rurales.

Ces technologies d'interaction s'inscrivent dans un écosystème plus vaste incluant la gestion numérique des services annexes, tels que la restauration.

3.5 Services de Restauration Numérique

Les systèmes de restauration numérique ont transformé la gestion des commandes dans le secteur touristique. L'étude de Nkosana et al. (2016) sur les applications mobiles de type PMS (Property Management System) révèle une amélioration de 25% de l'efficacité opérationnelle dans les petits restaurants ruraux d'Afrique du Sud. Cependant, cette recherche souligne un défi majeur : la dépendance à une connexion internet stable, particulièrement critique dans des zones comme le Parc National Zakouma où l'infrastructure réseau peut être intermittente.

Les solutions modernes intègrent désormais des fonctionnalités avancées telles que :

— Bornes interactives (self-ordering kiosks) pour réduire les files d'attente

— Synchronisation en temps réel avec les systèmes de gestion des stocks

— Intégration aux plateformes de paiement comme l'illustre l'étude de Do et al. (2023)

Ces innovations s'accompagnent d'un impératif de formation du personnel, comme le note Ankor et al. (2019) dans son analyse des portails web – un aspect crucial pour le parc, où la simplicité d'utilisation doit concilier besoins locaux et attentes des touristes internationaux.

La digitalisation des services sur site génère des flux de données qui, lorsqu'exploités via des outils analytiques, permettent d'optimiser la gestion quotidienne et stratégique.

3.6 Service de Reporting

Le reporting automatisé est un levier essentiel pour la prise de décision dans les écosystèmes touristiques. Les travaux de Barua & Kaiser (2024) démontrent comment l'IA peut prédire la demande avec 92% de précision, réduisant ainsi les goulets d'étranglement de 20%. Cette approche s'appuie sur :

— Tableaux de bord interactifs (ex : Grafana cité dans l'architecture)

— Indicateurs clés (KPI) : taux d'occupation, satisfaction client, délais moyens

— Intégration avec les microservices, bien que complexe comme le souligne l'étude

L'expérience de Michael Kenny et Robert Rosenstein dans l'analyse de 500 000 réservations met en lumière l'impact de la tarification dynamique sur les revenus – une piste pertinente pour le parc, où la saisonnalité influence fortement la fréquentation.

Cependant, Shrestha et al. (2024) rappellent la nécessité de disposer de volumes de données suffisants (50 000 avis dans leur cas) pour des analyses fiables – un défi pour une structure émergente comme Zakouma, qui pourrait compenser par des partenariats avec des plateformes tierces (ex : Tripadvisor).

Ces outils analytiques reposent sur une gestion rigoureuse des identités et accès, garantissant sécurité et personnalisation des services.

0.1 Synthèse et Techniques existantes et leurs limites

TABLE 1 – Comparaison des approches technologiques dans le tourisme

Auteurs	**Approches**	**Résultats**	**Avantages**	**Inconvénients**
Geert-Jan Bruinsma (2007)	Plateforme web de réservation centralisée	–	Interface très conviviale et large choix d'hébergements	Absence de certains hôtels dans la plateforme
Nkosana et al. (2016)	Restauration numérique	Automatisation des commandes (+25%)	Gestion des commandes automatisée	Absence de l'IA pour guider les utilisateurs
Dubois et Martin (2019)	Application mobile de réservation et suivi des services touristiques	–	Interface conviviale, accessibilité accrue	Dépendance à la connectivité, faible adoption en zones rurales
Viverit et al. (2023)	Clustering des courbes de réservation historique (pickup avancé)	MAPE d'environ 11,5% (prévision à 7j) ∼16,9% pour le pickup classique	Meilleure précision prédictive (MAPE plus faible)	Étude sur seulement 3 hôtels réels
Barua & Kaiser (2024)	Reporting	Prédiction IA (92% précision)	Recommandations en temps réel	Besoin en données historiques

Bien que ces techniques offrent des solutions intéressantes, comme celles proposées par **Ankor *et al.* (2019)** et **Nkosana *et al.* (2016)**, elles présentent néanmoins des défis dans le contexte particulier des parcs nationaux tels que **Zakouma**, en raison d'infrastructures limitées, d'un manque de compétences, ainsi que de coûts exorbitants.

Ajoutons également que la solution de **Booking.com (2007)** représente une alternative évidente, mais ses commissions sont trop élevées pour les touristes. C'est pourquoi le parc n'est pas partenaire de cette plateforme.

La solution de **Do *et al.* (2023)** constitue une piste intéressante, mais elle reste limitée à une application dans le contexte du Vietnam. Bien que ces approches offrent des avancées notables, comme celles développées par NKOSANA ET AL. (2016) avec la restauration numérique (+25% d'automatisation) ou VIVERIT ET AL. (2023) grâce au clustering des données (MAPE de 11,5%), elles révèlent des

limites criantes dans le contexte des parcs nationaux africains tels que Zakouma. Les infrastructures précaires, le manque d'expertise locale en IA, et les coûts de déploiement prohibitifs rendent ces solutions difficilement transposables.

La plateforme centralisée de Geert-Jan Bruinsma (2007), bien qu'intuitive, exclut déjà certains établissements – un écueil majeur pour Zakouma où l'inclusion des acteurs locaux (guides, lodges familiaux) est essentielle. Quant à l'application mobile de Dubois et Martin (2019), son exigence de connectivité permanente la disqualifie dans les zones reculées du parc, où le réseau est intermittent.

L'approche de Barua & Kaiser (2024), bien que prometteuse (92% de précision), bute sur l'absence de données historiques suffisantes – un problème récurrent dans les réserves peu digitalisées. Ces limitations soulignent la nécessité d'innover *avec* les contraintes du terrain, plutôt que de dupliquer des modèles conçus pour des écosystèmes technologiquement matures.

Ces défis incluent :

- L'inadéquation des outils standards avec les spécificités locales ;
- Le manque de ressources techniques pour le déploiement et la maintenance des systèmes ;
- Les contraintes environnementales, telles que l'accès limité à l'électricité ou à Internet.

Ces limites justifient le besoin de développer une **solution adaptée**, spécifiquement conçue pour répondre aux besoins du **Parc National de Zakouma**. Une telle solution devra **intégrer les avantages des techniques existantes** tout en **tenant compte des réalités locales**, afin d'assurer une **gestion efficace et durable des services touristiques**.

Chapitre 4 : Architecture et Solution proposée pour le système touristique du Parc de Zakouma

0.2 Introduction

Après avoir étudié la revue de la littérature qui a permis d'identifier les approches existantes et les choix technologiques pertinents, ce chapitre se consacre à la conception de notre système. Il expose l'architecture générale de l'application, les modèles de données, ainsi que les différents composants et leurs interactions. Cette phase de conception vise à assurer la cohérence, la robustesse et l'évolutivité de la solution proposée. Elle constitue ainsi une étape essentielle avant la mise en œuvre technique.

0.2.1 Architecture globale du système de gestion touristiques pour le Parc National de Zakouma

La solution proposée pour la gestion des services touristiques du Parc National Zakouma repose sur une architecture micro-services, qui permettra de décomposer l'application en plusieurs services indépendants, chacun responsable d'une fonctionnalité spécifique.

Cette architecture présente plusieurs avantages, notamment la flexibilité, la scalabilité et la facilité de maintenance. Chaque microservice sera autonome, avec sa propre base de données et son propre domaine de gestion.

L'utilisation d'une architecture micro-services facilitera également l'intégration de nouvelles fonctionnalités à l'avenir, comme l'ajout de nouveaux services ou l'intégration de nouveaux systèmes.

Cette architecture offre une solution flexible et scalable pour répondre aux besoins d'application moderne. En adoptant des pratiques de développement et de déploiement agiles, ce système est conçu pour évoluer avec les exigences du marché, tout en garantissant une haute disponibilité et une maintenance simplifiée. Cette

approche innovante positionne le système comme une solution robuste et adaptable face aux défis technologiques actuels.

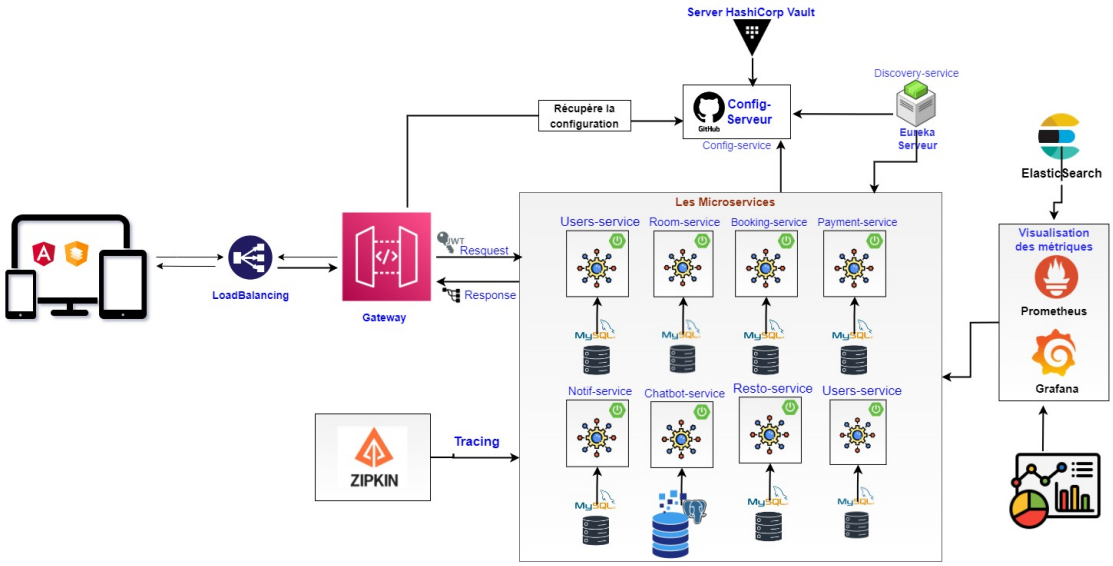


FIGURE 1 – Architecture générale de la Solution

0.3 Les micro-services :

0.3.1 Gestion des utilisateurs : users-service

Le microservice users-service est un composant essentiel de l'architecture globale du système, dédié à la gestion des utilisateurs. Développé en Spring Boot, il offre une interface robuste pour l'enregistrement, l'authentification et la gestion des comptes utilisateurs. Ce service est conçu pour garantir une expérience utilisateur fluide, tout en intégrant des mesures de sécurité avancées.

0.3.1.1 Fonctionnalités Principales

Enregistrement des Utilisateurs Lorsqu'un nouvel utilisateur s'inscrit, le microservice envoie un e-mail de confirmation contenant un code à six chiffres. Ce code permet de valider l'adresse e-mail de l'utilisateur et d'assurer que l'inscription est légitime. Une fois l'e-mail confirmé, l'utilisateur est redirigé vers la page de connexion, facilitant ainsi l'accès à l'application.

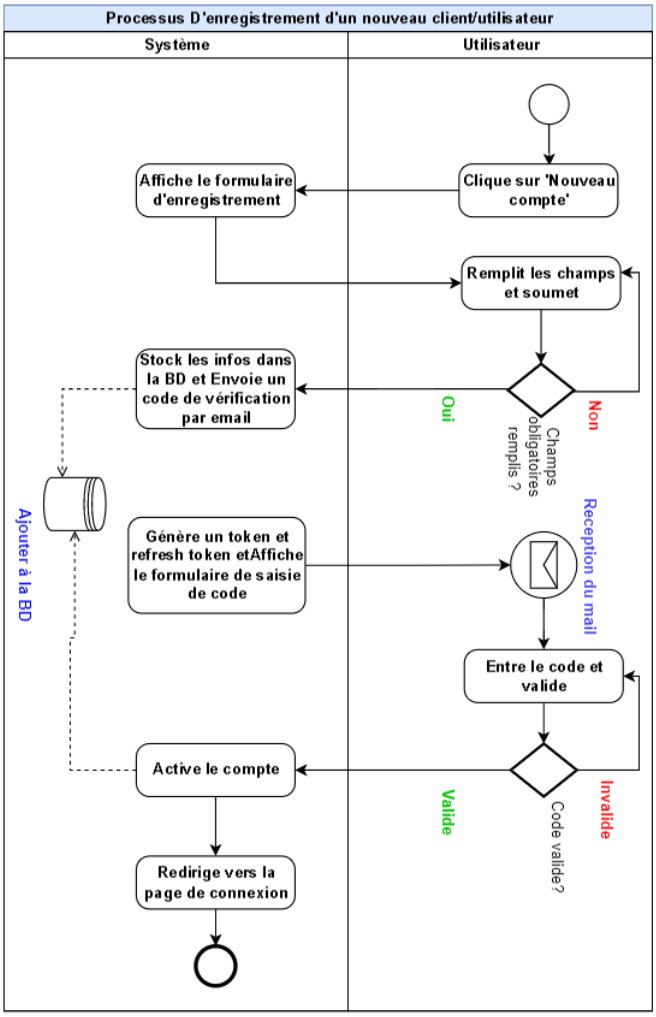


FIGURE 2 – Diagramme de processus d'enregistrement

Authentification et Gestion des Sessions : Le microservice utilise des tokens JWT (JSON Web Tokens) pour gérer l'authentification. Lorsqu'un utilisateur se connecte avec succès, un access token et un refresh token sont générés. L'access token permet d'accéder aux ressources protégées de l'application, tandis que le refresh token permet de renouveler l'access token sans nécessiter une nouvelle connexion, améliorant ainsi l'expérience utilisateur.

Gestion du Mot de Passe : Le service offre des fonctionnalités pour changer le mot de passe et récupérer un mot de passe oublié. En cas d'oubli, l'utilisateur peut

initier un processus de réinitialisation, qui implique l'envoi d'un e-mail contenant un lien sécurisé pour définir un nouveau mot de passe.
L'architecture du users-service repose sur une API REST, permettant une communication efficace entre le frontend et le backend. Les principales interactions incluent :

Endpoints d'Inscription : Un endpoint pour enregistrer de nouveaux utilisateurs et envoyer des e-mails de confirmation. Endpoints de Connexion : Un endpoint pour authentifier les utilisateurs et fournir des tokens JWT. Endpoints de Gestion du Mot de Passe : Endpoints dédiés pour changer le mot de passe et réinitialiser un mot de passe oublié.

Sécurité et Validation Pour garantir la sécurité des données des utilisateurs, le service applique les meilleures pratiques en matière de sécurité, notamment :

Validation des Données : Les entrées utilisateur sont validées pour prévenir les attaques par injection et garantir l'intégrité des données. Chiffrement des Mots de Passe : Les mots de passe sont stockés de manière sécurisée en utilisant des algorithmes de hachage robustes. Gestion des Tokens : Les tokens JWT sont signés et expirent après une période définie, réduisant ainsi le risque d'accès non autorisé.
Le microservice users-service constitue un élément clé de l'architecture du système, offrant une gestion complète des utilisateurs tout en garantissant la sécurité et la facilité d'utilisation. Grâce à l'intégration de technologies modernes comme Spring Boot et Angular, ce service est conçu pour s'adapter aux besoins des utilisateurs, tout en permettant une évolutivité et une maintenance aisées. En fournissant des fonctionnalités telles que l'enregistrement, l'authentification et la gestion des mots de passe, le users-service joue un rôle central dans l'expérience globale de l'application.

0.3.2 Gestion des chambres : room-service

Le microservice room-service est un composant clé de l'architecture globale du système, dédié à la gestion des chambres. Ce service permet d'enregistrer, de mettre à jour et de gérer les informations relatives aux chambres disponibles dans l'application. Développé en Spring Boot, room-service assure une gestion efficace des données, tout en offrant une interface conviviale pour les utilisateurs.

0.3.2.1 Fonctionnalités Principales

Enregistrement des Chambres

Le microservice permet l'enregistrement de nouvelles chambres dans le système. Chaque chambre peut être associée à des attributs tels que le numéro de chambre,

le type (simple, double, suite), le prix, et d'autres caractéristiques pertinentes. Lors de l'enregistrement, des validations sont effectuées pour garantir que les données saisies sont correctes et conformes aux exigences.

Mise à Jour des Informations

room-service permet également de mettre à jour les informations des chambres existantes. Cela inclut la modification des prix, des caractéristiques ou de l'état de disponibilité des chambres. Les utilisateurs peuvent accéder à une interface dédiée pour effectuer ces mises à jour, assurant ainsi que les informations restent à jour.

Consultation des Chambres

Les utilisateurs peuvent consulter la liste des chambres disponibles à tout moment. Ce service fournit des endpoints pour récupérer les informations sur les chambres, facilitant ainsi la recherche et la réservation. Des filtres peuvent être appliqués pour affiner les résultats selon des critères spécifiques, tels que le type de chambre ou le prix.

Le microservice room-service joue un rôle essentiel dans la gestion des chambres au sein de l'application. En offrant des fonctionnalités complètes pour l'enregistrement, la mise à jour et la consultation des informations sur les chambres, ce service contribue à une expérience utilisateur optimale. Grâce à l'utilisation de technologies modernes comme Spring Boot, room-service est conçu pour être performant, évolutif et facile à maintenir, répondant ainsi aux besoins dynamiques du système.

0.3.3 Gestion des réservations : reservation-service

Le microservice reservation-service est un élément central de l'architecture du système, chargé de la gestion des réservations. Ce service permet aux utilisateurs de réserver des chambres tout en interagissant efficacement avec d'autres microservices, tels que room-service et users-service. Développé en Spring Boot, reservation-service utilise OpenFeign et Kafka pour assurer une communication fluide entre les différents composants du système.

0.3.3.1 Fonctionnalités Principales

Création de Réservations

Le microservice permet aux utilisateurs de créer des réservations pour des chambres disponibles. Lorsqu'une réservation est effectuée, les détails, tels que l'identifiant de l'utilisateur, le numéro de chambre, les dates de début et de fin, sont enregistrés dans la base de données. Des validations sont mises en place pour s'assurer que les chambres sélectionnées sont disponibles aux dates demandées.

Consultation des Réservations

Les utilisateurs peuvent consulter leurs réservations existantes via une interface dédiée. Cela inclut la possibilité de voir les détails de chaque réservation, tels que les dates, le statut et les informations sur la chambre. Des filtres peuvent être appliqués pour faciliter la recherche de réservations spécifiques.

Annulation de Réservations

Le service offre également la possibilité d'annuler des réservations. Les utilisateurs peuvent initier une annulation, qui est ensuite traitée par le microservice, mettant à jour le statut de la réservation et libérant la chambre pour de futures réservations.

Le microservice reservation-service joue un rôle crucial dans la gestion des réservations au sein de l'application. En permettant la création, la consultation et l'annulation des réservations, tout en interagissant efficacement avec room-service et users-service via OpenFeign et Kafka, ce service contribue à une expérience utilisateur optimale. Grâce à son architecture robuste et à l'utilisation de technologies modernes, reservation-service est conçu pour être performant, évolutif et facile à maintenir, répondant ainsi aux besoins dynamiques du système.

0.3.4 Gestion des paiements : paiement-service

Le microservice payment-service est un composant essentiel de l'architecture du système, dédié à la gestion des paiements associés aux réservations. Ce service permet de traiter les transactions de manière sécurisée, en intégrant des solutions de paiement modernes telles que Stripe pour les paiements en ligne, ainsi que des options de paiement sur place. Développé en Spring Boot, payment-service garantit une expérience utilisateur fluide tout en respectant les normes de sécurité des données financières.

0.3.4.1 Fonctionnalités Principales

Traitement des Paiements en Ligne Le microservice permet aux utilisateurs de régler leurs réservations en ligne via Stripe. Lorsqu'une réservation est confirmée, l'utilisateur peut effectuer un paiement sécurisé en entrant ses informations de carte de crédit. payment-service interagit avec l'API de Stripe pour créer des transactions, gérer les paiements et traiter les éventuels remboursements. Les détails de la transaction sont enregistrés pour assurer un suivi et une gestion appropriés.

Options de Paiement sur Place En plus des paiements en ligne, le service prend en charge les paiements sur place. Les utilisateurs peuvent choisir de régler leur réservation directement à l'arrivée, en utilisant des moyens de paiement tels que les cartes de crédit, les espèces ou d'autres méthodes acceptées. payment-service

gère également la validation et l'enregistrement de ces paiements, garantissant que toutes les transactions sont correctement suivies.

Gestion des Remboursements Le microservice offre des fonctionnalités pour traiter les demandes de remboursement. Si un utilisateur annule une réservation, payment-service peut initier un remboursement via Stripe pour les paiements effectués en ligne, tout en assurant que les paiements sur place sont gérés de manière appropriée.

Sécurité et Conformité Pour garantir la sécurité des transactions financières, plusieurs mesures sont mises en place :

Sécurisation des Données : Toutes les informations de paiement sont traitées conformément aux normes PCI DSS (Payment Card Industry Data Security Standard), garantissant la sécurité des données des utilisateurs. **Validation des Transactions** : Chaque transaction est validée pour prévenir les fraudes et garantir que seules les transactions légitimes sont traitées.
Le microservice payment-service joue un rôle crucial dans la gestion des paiements au sein de l'application. En offrant des options de paiement en ligne via Stripe et des paiements sur place, ce service assure une expérience utilisateur optimale tout en garantissant la sécurité des transactions. Grâce à son architecture robuste et à l'utilisation de technologies modernes, payment-service est conçu pour être performant, évolutif et facile à maintenir, répondant ainsi aux besoins dynamiques du système.

0.3.5 Gestion des commandes : commande-service

Le microservice commande-service est un composant essentiel de l'architecture du système, dédié à la gestion des commandes de restauration pour les touristes. Une fois qu'une réservation est confirmée, les clients peuvent passer des commandes de repas en fonction de leur calendrier, intégrant ainsi une expérience culinaire personnalisée. Développé en Spring Boot, commande-service utilise l'API de Google Calendar pour synchroniser les commandes avec les horaires des clients.

0.3.5.1 Fonctionnalités Principales

Affichage des Commandes Disponibles Après la confirmation d'une réservation, commande-service génère une interface utilisateur où les clients peuvent consulter les options de repas disponibles. Cela inclut des plats variés, des menus spéciaux et des recommandations basées sur les préférences des utilisateurs. Les clients peuvent visualiser les commandes disponibles en fonction de leur calendrier,

ce qui leur permet de planifier leurs repas à des moments qui leur conviennent.

Passation de Commandes

Les clients peuvent passer des commandes directement via l'interface fournie par commande-service. Une fois qu'une commande est sélectionnée, les utilisateurs peuvent choisir l'heure de livraison ou de service, qui est synchronisée avec leur calendrier Google. Le service enregistre les détails de chaque commande, y compris les éléments commandés, l'heure de livraison souhaitée et les informations de contact.

Le microservice commande-service joue un rôle crucial dans la gestion des commandes de restauration au sein de l'application. En permettant aux clients de passer des commandes personnalisées en fonction de leur calendrier, et en intégrant l'API de Google Calendar pour une synchronisation fluide, ce service contribue à une expérience utilisateur enrichissante. Grâce à son architecture moderne et à son utilisation de technologies avancées, commande-service est conçu pour être performant, évolutif et facile à maintenir, répondant ainsi aux besoins variés des clients.

0.3.6 ChatBotAI : chatBot-service

Le microservice chatBot-service est un composant innovant de l'architecture du système, permettant aux clients (utilisateurs) de poser des questions concernant l'institution et d'obtenir des réponses précises grâce à une intelligence artificielle (IA). Que ce soit des questions sur les réservations, les paiements, les commandes ou des sujets spécifiques à l'entreprise, chat-service offre une assistance instantanée. De plus, en tant que parc, il permet également aux scientifiques de poser des questions sur les animaux, leur mode de vie, et d'autres sujets pertinents.

0.3.6.1 Fonctionnalités Principales

Assistance Clientèle Les utilisateurs peuvent poser des questions sur divers sujets, notamment les réservations, les paiements et les commandes. Le système fournit des réponses claires et précises et est disponible 24H/24, améliorant ainsi l'expérience utilisateur. Les réponses sont générées en temps réel par l'IA, garantissant que les utilisateurs reçoivent des informations actualisées et pertinentes.

Questions Spécifiques aux Animaux du parc Les scientifiques et les passionnés de la nature peuvent poser des questions spécifiques sur les animaux présents dans le parc, leur comportement, leur habitat et leur mode de vie. Cela enrichit l'expérience éducative des visiteurs.

Système de Récupération Augmentée Générée (RAG) Le système est développé en utilisant le modèle RAG (Retrieval-Augmented Generation). Les informations sont fournies sous forme de fichiers PDF, qui sont ensuite découpés en "chunks" (morceaux) et enregistrés dans un vectorStore basé sur PostgreSQL. Lorsqu'une question est posée, l'agent IA interroge le vectorStore pour récupérer les morceaux d'information pertinents, puis génère une réponse contextuelle basée sur ces données.
Le microservice chat-service représente une avancée significative dans la gestion des interactions avec les utilisateurs. En permettant aux clients de poser des questions sur divers sujets et en intégrant un système de récupération augmentée générée, ce service offre une expérience d'assistance enrichissante et informative. Grâce à son architecture moderne et à l'utilisation de technologies avancées, chat-service est conçu pour être performant, évolutif et facile à maintenir, répondant ainsi aux besoins variés des utilisateurs.

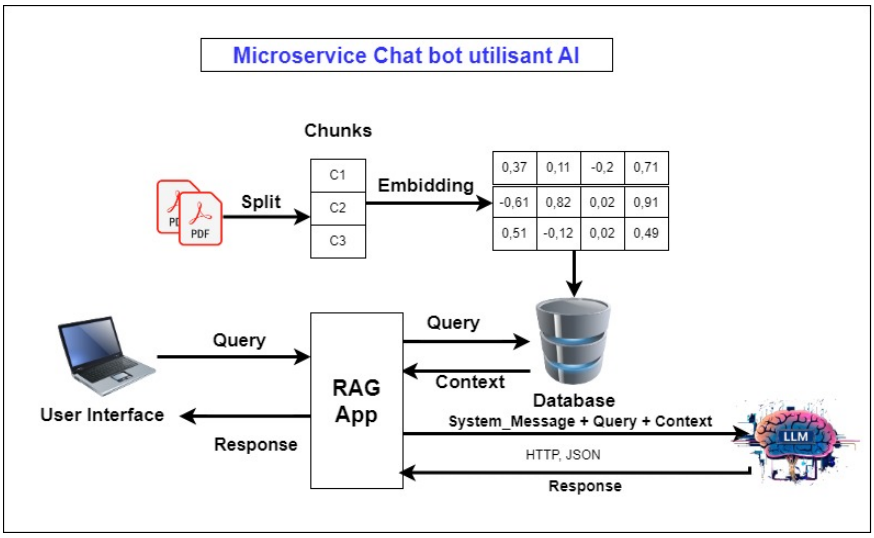


FIGURE 3 – Schéma du service de chatbot IA

0.3.7 Prédication : predicat-service :

Le microservice prédicat-service est un composant crucial de l'architecture du système, dédié à la prédiction des réservations futures et des utilisateurs potentiels. En s'appuyant sur la bibliothèque Prophet de Facebook, ce service utilise des techniques de machine learning pour analyser les tendances des réservations et fournir des prévisions précises. Développé en partie avec Python (Flask) et en partie avec Spring Boot, prédicat-service combine le meilleur des deux mondes pour offrir des performances optimales.

0.3.7.1 Fonctionnalités Principales

Prédiction des Réservations Futures Le microservice analyse les données historiques des réservations pour identifier des tendances et des modèles. Grâce à l'algorithme de Prophet, il peut prédire les fluctuations des réservations à venir, permettant ainsi à l'institution de mieux planifier ses ressources. Les prévisions incluent des informations sur le nombre attendu de réservations, les périodes de forte affluence, et d'autres métriques pertinentes qui aident à la prise de décision.

Prédiction du Nombre d'Utilisateurs Futurs : Le microservice analyse les données historiques des utilisateurs pour identifier des tendances et des modèles. Grâce à l'algorithme de Prophet, il peut prédire le nombre d'utilisateurs attendus dans le futur, permettant ainsi à l'institution de mieux planifier ses ressources. Les prévisions incluent des informations sur les périodes de forte affluence et d'autres métriques pertinentes, facilitant ainsi la prise de décision stratégique.
Le microservice prédicat-service joue un rôle essentiel dans la gestion des réservations en fournissant des prévisions basées sur des analyses de données avancées. En intégrant la puissance de Prophet de Facebook et en utilisant une architecture hybride avec Flask et Spring Boot, ce service est capable de prédire les réservations futures et d'identifier les utilisateurs potentiels de manière efficace. Grâce à son approche moderne et à l'utilisation de technologies de pointe, prédicat-service est conçu pour être performant, évolutif et capable de répondre aux besoins dynamiques de l'institution.

0.3.8 Reporting-service :

Le microservice reporting-service est un composant essentiel de l'architecture du système, dédié à la gestion et à l'automatisation des rapports hebdomadaires sur le fonctionnement de l'application. Ce service permet d'envoyer à l'administrateur des rapports détaillés chaque semaine, fournissant des informations précieuses sur le comportement de l'application, le nombre d'utilisateurs, et d'autres métriques clés. De plus, il génère des fichiers PDF hebdomadaires disponibles pour téléchargement.

0.3.8.1 Fonctionnalités Principales

Génération de Rapports Hebdomadaires Le microservice compile des données sur le fonctionnement de l'application, y compris :

— Nombre d'utilisateurs actifs

— Nombre de chambres occupées et disponibles

— Statistiques sur les réservations : confirmées et annulées

— Le nombre des commandes passées ;

Ces rapports sont générés automatiquement à la fin de chaque semaine, garantissant que l'administrateur dispose toujours d'informations actualisées.

Envoi de Rapports à l'Administrateur Les rapports hebdomadaires sont envoyés par e-mail à l'administrateur, permettant une surveillance facile et rapide des performances de l'application. Les notifications peuvent être configurées pour inclure des résumés et des alertes sur des métriques critiques.

Affichage et Téléchargement de PDF Les rapports générés sont convertis en fichiers PDF, qui sont ensuite affichés dans l'interface de l'application. Les administrateurs peuvent facilement télécharger ces PDF pour une consultation ultérieure ou pour des besoins de documentation.
Le microservice reporting-service joue un rôle crucial dans la gestion et l'analyse des performances de l'application. En automatisant la génération et l'envoi de rapports hebdomadaires, il permet à l'administrateur de rester informé des tendances et des comportements clés. Grâce à son architecture moderne et à l'utilisation de technologies adaptées, reporting-service est conçu pour être performant, évolutif et capable de répondre aux besoins d'analyse de l'institution.

0.4 Gateway : gateway-service

Le microservice gateway-service est le point d'entrée central de l'application, jouant un rôle crucial dans le routage des requêtes, la gestion de la sécurité et l'orchestration des interactions entre les différents microservices. En utilisant le JSON Web Token (JWT) pour sécuriser les communications, gateway-service assure que toutes les requêtes passent par elle avant d'être redirigées vers d'autres services tels que users-service, room-service, reservation-service, reporting-service, et bien d'autres.

0.4.1 Fonctionnalités Principales

Routage des Requêtes : gateway-service agit comme un routeur intelligent, dirigeant les requêtes des clients vers les microservices appropriés. Cela simplifie la gestion des requêtes et améliore l'efficacité du système. Les règles de routage peuvent être configurées pour diriger les requêtes en fonction de différents critères, tels que le type de service demandé ou les paramètres de la requête.

Sécurité avec JWT : La sécurité de l'application est renforcée grâce à l'utilisation de JWT. Chaque requête doit inclure un token valide, garantissant que seules les utilisateurs authentifiés peuvent accéder aux ressources. gateway-service est responsable de la validation des tokens JWT avant de permettre l'accès aux microservices, assurant ainsi la protection des données sensibles. Pour assurer la sécurité et la performance de gateway-service, plusieurs mesures sont mises en place :

Contrôle d'Accès : Mise en place de politiques de contrôle d'accès basées sur les rôles des utilisateurs, garantissant que seules les actions autorisées peuvent être effectuées.

Optimisation des Performances : Le microservice est conçu pour gérer un grand volume de requêtes simultanées, optimisant ainsi la latence et la réactivité de l'application.

Le microservice gateway-service est un élément fondamental de l'architecture de l'application, agissant comme un point d'entrée sécurisé et efficace pour toutes les requêtes. Grâce à son rôle de routage et à l'utilisation de JWT pour la sécurité, il garantit que les interactions entre les clients et les microservices sont fluides et sécurisées. Avec une architecture moderne et une approche axée sur la performance, gateway-service est essentiel pour le bon fonctionnement de l'ensemble du système.

0.5 Discovery-service

Le service-discovery, implémenté avec Eureka, joue un rôle essentiel dans les architectures basées sur les microservices. Il permet aux différents services de se découvrir et de communiquer entre eux de manière dynamique. Grâce à Eureka, chaque microservice peut s'enregistrer auprès d'un serveur de découverte lors de son démarrage, ce qui facilite la gestion des instances de services en temps réel.

Eureka permet également aux services de récupérer des informations sur d'autres services disponibles dans l'écosystème. Lorsqu'un service souhaite interagir avec un autre, il peut interroger le serveur Eureka pour obtenir l'adresse et le port du service cible, ce qui simplifie considérablement le processus de communication entre les microservices. Cela est particulièrement utile dans des environnements où les instances de services peuvent être ajoutées ou supprimées fréquemment, comme dans des architectures cloud.

De plus, Eureka offre des fonctionnalités de résilience en permettant aux services de continuer à fonctionner même si certaines instances deviennent inaccessibles. En cas de défaillance d'une instance, le service client peut automatiquement se connecter à une autre instance disponible, assurant ainsi une continuité de ser-

vice. Cela contribue à améliorer la robustesse et la disponibilité de l'application dans son ensemble.

0.6 Le service de configuration : config-service

Le service de configuration est un composant crucial dans les architectures de microservices, conçu pour gérer les configurations de manière centralisée. Ce service permet aux différents microservices de récupérer leurs paramètres de configuration à partir d'un dépôt GitHub, garantissant ainsi que tous les services utilisent des configurations cohérentes et à jour.

L'utilisation d'un dépôt GitHub comme source de configuration offre plusieurs avantages. Tout d'abord, cela permet une gestion versionnée des configurations, facilitant le suivi des modifications et le retour à des versions antérieures si nécessaire. De plus, cela permet aux équipes de développement de collaborer efficacement sur les configurations, en utilisant les outils de contrôle de version standard.

Pour assurer la sécurité des données sensibles contenues dans les configurations, le service de configuration est sécurisé par HashiCorp Vault. Ce système de gestion des secrets permet de stocker et de gérer les informations sensibles, telles que les mots de passe, les clés API et d'autres données critiques, de manière sécurisée. Grâce à cette intégration, les microservices peuvent accéder aux configurations nécessaires sans exposer les informations sensibles, garantissant ainsi la sécurité de l'ensemble du système.

0.7 Observabilité et Monitoring

L'observabilité et le monitoring sont des aspects essentiels pour assurer le bon fonctionnement et la performance des architectures basées sur des microservices. En utilisant des outils comme ELK, Zipkin, Prometheus et Grafana, les équipes peuvent obtenir une visibilité complète sur l'état et le comportement de leurs applications.

0.7.1 Rôle de l'observabilité et Monitoring

Visibilité des Systèmes : L'observabilité permet d'obtenir une vue d'ensemble sur le fonctionnement des microservices. En collectant des données sur les performances, les erreurs et les comportements des services, les équipes peuvent com-

prendre comment les différents composants interagissent et identifier rapidement les anomalies.

Détection des Anomalies : Le monitoring proactif aide à détecter les problèmes avant qu'ils n'affectent les utilisateurs finaux. Grâce à des alertes configurées sur des seuils spécifiques, les équipes peuvent être informées immédiatement de toute défaillance ou dégradation des performances, ce qui leur permet d'agir rapidement pour résoudre les problèmes.

Analyse des Performances : Les outils d'observabilité fournissent des métriques et des logs détaillés, permettant aux équipes d'analyser les performances des microservices. Cela inclut l'évaluation des temps de réponse, l'utilisation des ressources et les taux d'erreur. Ces analyses aident à identifier les goulets d'étranglement et à optimiser les performances globales du système.

Traçabilité des Transactions : L'observabilité permet de suivre le parcours des requêtes à travers les différents microservices. Cela aide à comprendre comment les demandes sont traitées et où des retards peuvent survenir. La traçabilité est essentielle pour diagnostiquer les problèmes de latence et améliorer l'efficacité des services.

Amélioration Continue : En analysant les données collectées, les équipes peuvent identifier les tendances et les opportunités d'amélioration. Cela permet de mettre en place des modifications et des optimisations basées sur des données réelles, contribuant ainsi à l'évolution continue des microservices

Support à la Décision : L'observabilité fournit des informations précieuses qui aident les équipes à prendre des décisions éclairées concernant l'architecture, la mise à l'échelle et la gestion des ressources. Ces décisions sont basées sur des données concrètes plutôt que sur des suppositions, ce qui améliore la fiabilité du système.

0.8 La communication entre les microservices

Dans une architecture de microservices, une communication efficace et fiable entre les services est essentielle. Cela repose sur plusieurs composants et mécanismes qui garantissent la performance, la résilience et la sécurité des interactions. Les principaux éléments impliqués : **Échange Synchrone avec OpenFeign** La communication entre les microservices peut se faire de manière synchrones grâce à OpenFeign. Cet outil facilite la création de clients HTTP en permettant aux développeurs de définir des interfaces Java qui se traduisent automatiquement en appels HTTP. Cela rend l'intégration entre les services plus simple et rapide.

Échange Asynchrone avec Kafka Pour les communications asynchrones, Kafka est souvent utilisé. Ce système de messagerie permet aux microservices d'échanger des messages de manière décentralisée et sans couplage direct. Les services peuvent publier des événements sur des topics, et d'autres services peuvent s'abonner pour recevoir ces messages. Cela améliore la résilience et la scalabilité de l'architecture.

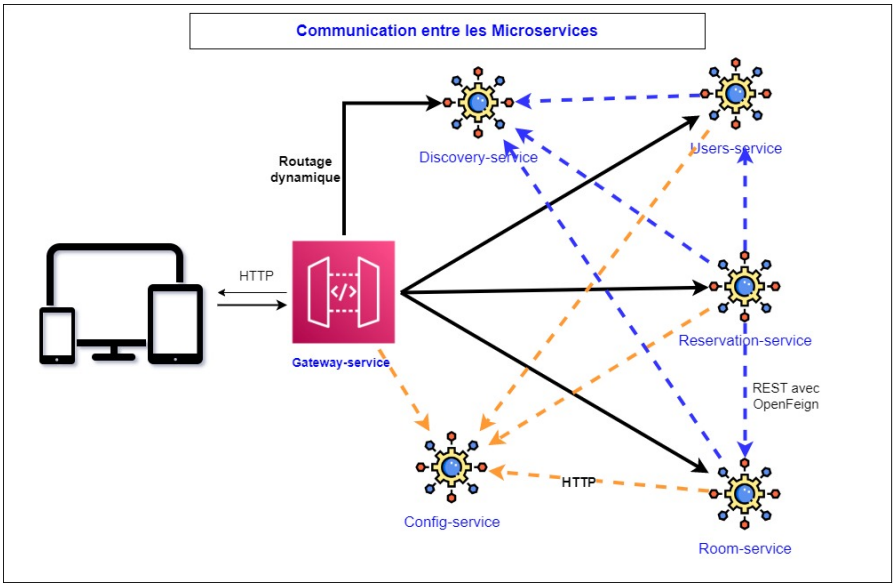


FIGURE 4 – Schéma du service de gestion des réservations

0.9 Les Bases de données

Dans une architecture microservices, chaque service est conçu pour être autonome et indépendant, ce qui inclut la gestion de sa propre base de données. Ce modèle repose sur le principe de bounded context, qui définit les limites d'un modèle de domaine spécifique, permettant ainsi à chaque microservice de gérer ses propres données de manière isolée.

0.9.1 Principe de Bounded Context :

Le principe de bounded context stipule que chaque microservice doit avoir son propre modèle de données et sa propre base de données. Cela signifie que :

Indépendance : Chaque microservice peut évoluer et être déployé indépendamment des autres, ce qui facilite les mises à jour et les modifications. Modèles de

Données Spécifiques : Les modèles de données peuvent être adaptés aux besoins spécifiques de chaque service, ce qui permet une optimisation des performances.

0.9.1.1 Diagramme de déploiement de la solution

Un diagramme de déploiement (en UML) est une représentation visuelle de la façon dont les artefacts logiciels (services, exécutables, conteneurs, scripts, etc.) sont mappés sur des noeuds physiques ou virtuels (serveurs, machines virtuelles, conteneurs Docker, edge devices, etc.). Il décrit les noeuds (machines, VM, clusters, conteneurs) et leurs relations physiques (réseau, topologie), les artefacts déployés sur chaque nœud (microservice JAR/WAR, image Docker), les connecteurs (protocole HTTP, MQ, gRPC) qui définissent comment les nœuds communiquent. Voici le diagramme de déploiement de la solution :

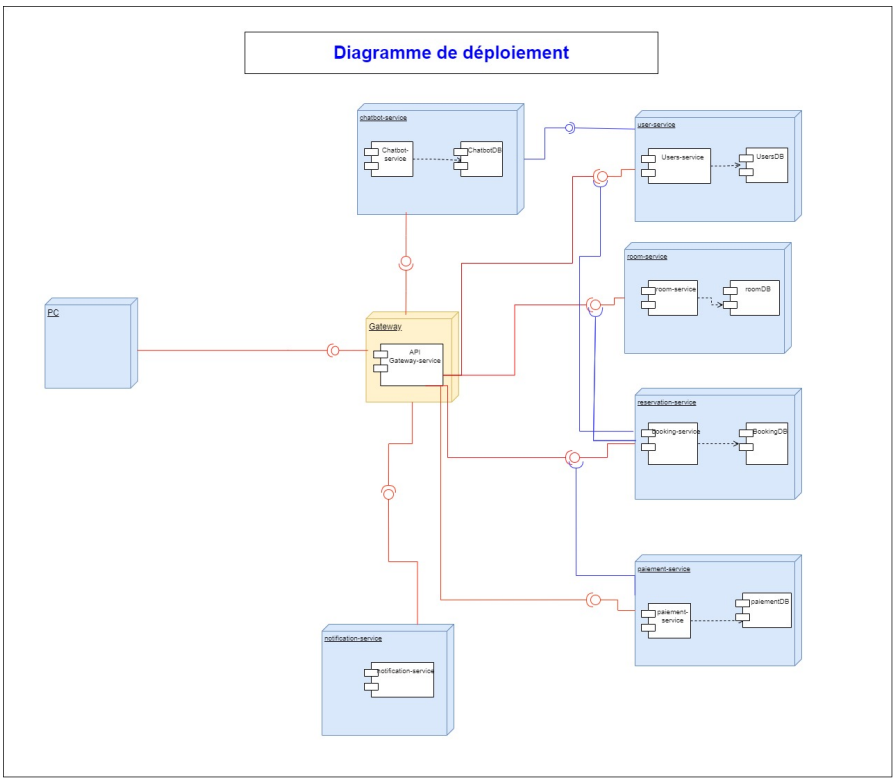


FIGURE 5 – Diagramme de déploiement

Chaque artefact sera dans une machine virtuelle conteneurisé par Docker et les touts managé par Kubernetes. En résumé, cette architecture de solution, reposant sur une approche micro-services, permettra de répondre efficacement aux besoins

du Parc National Zakouma, tout en garantissant la scalabilité, la sécurité et la flexibilité de la solution à long terme.

Chapitre 5 : Implémentation de la Solution

0.10 Introduction

Dans le chapitre précédent consacré à la conception de notre solution, nous avons défini l'architecture technique détaillée de notre système, spécifié les modèles de données et les interfaces entre les différents composants, et validé nos choix technologiques à travers des schémas UML et des protocoles d'échange. Cette phase de conception nous a permis d'établir un cadre théorique solide et des spécifications fonctionnelles précises.

Nous passons maintenant à la phase concrète de l'implémentation, où nous allons donner vie à cette conception à travers le développement effectif de l'application. Ce chapitre marque ainsi la transition entre la réflexion architecturale et la matérialisation opérationnelle de notre solution.

0.11 Les Outils Utilisés

Dans le cadre de notre projet, plusieurs outils ont été employés pour garantir une mise en œuvre efficace et robuste. Ces outils comprennent des langages de programmation, des frameworks, ainsi que des environnements de développement intégrés (IDE) qui facilitent le processus de développement. Chaque outil a été sélectionné en fonction de ses capacités à répondre aux exigences spécifiques du projet.

0.11.1 La Technologie backend

Le backend est la partie du système qui gère la logique métier, l'interaction avec la base de données et la communication avec le frontend. Il joue un rôle crucial dans le traitement des requêtes des utilisateurs et la gestion des données. Dans notre projet, nous avons opté pour une architecture basée sur des microservices, ce qui

permet une scalabilité et une flexibilité accrues. Les langages et frameworks choisis pour le développement backend sont Java et Python, associés respectivement à Spring Boot et Flask.

0.11.1.1 Les Langages

Java : est un langage de programmation orienté objet, largement utilisé pour le développement d'applications d'entreprise. Sa robustesse, sa portabilité et sa grande communauté de développeurs en font un choix privilégié pour les applications backend. Java est également connu pour sa gestion de la mémoire et sa sécurité, ce qui en fait un excellent choix pour des projets nécessitant une haute disponibilité. **Python :** est un langage de programmation interprété qui se distingue par sa simplicité et sa lisibilité. Il est particulièrement apprécié pour le développement rapide d'applications et son utilisation dans le domaine de la science des données et de l'intelligence artificielle. Grâce à sa vaste bibliothèque de modules, Python permet de développer des fonctionnalités complexes avec un minimum de code.

0.11.1.2 Les Frameworks

Spring Boot : est un framework Java qui simplifie le processus de développement d'applications en fournissant des configurations par défaut et des fonctionnalités prêtes à l'emploi. Il permet de créer des applications autonomes et de faciliter le déploiement. Spring Boot est également connu pour sa capacité à intégrer facilement des bases de données et d'autres services.

Comparaison du framework Spring boot avec les autres frameworks Java

Critère	Quarkus	Micronaut	Spring Boot
Écosystème	Extensions officielles (Kafka, Hibernate)	Modules légers et personnalisables	Vaste écosystème (Spring Data, Security)
Performance	Démarrage ultra-rapide	Faible empreinte mémoire, injection à la compilation	Démarrage moyen, dépendances runtime
Cloud/Serverless	Conçu pour Kubernetes et serverless	Support natif AWS Lambda, Azure Functions	Intégrable via Spring Cloud, mais optimisé
Facilité d'apprentissage	Courbe modérée (nécessite des connaissances cloud)	Similaire à Spring, mais plus orienté modularité	Documentation abondante, communauté large
Cas d'usage	Microservices cloud-native, applications natives	Applications légères, serverless, IoT	Applications d'entreprise, monolithes modulaires

Table 2 – Comparaison entre Quarkus, Micronaut et Spring Boot

Flask : est un micro-framework pour Python qui permet de créer des applications web légères. Il est minimaliste, ce qui permet aux développeurs de choisir les composants qu'ils souhaitent intégrer. Flask est idéal pour des projets de petite à moyenne envergure, où la rapidité de développement est essentielle.

Table 3 – Comparaison entre Flask et Django

Critère	Flask	Django
Philosophie	Microframework minimaliste	Framework "batteries included"
Flexibilité	Hautement configurable	Structure plus rigide
Interface Admin	Doit être construite manuellement	Générée automatiquement
Courbe d'apprentissage	Plus facile pour débuter	Plus complexe mais documentée
Performance	Plus léger et rapide	Plus complet mais plus lourd
Écosystème	Nombreuses extensions optionnelles	Tout inclus mais moins flexible

0.11.2 La Technologie frontend

0.11.2.1 Les langages et style

Le développement frontend constitue la partie visible et interactive d'une application web, directement accessible aux utilisateurs. Il repose sur trois langages fondamentaux et un style. **HTML : (HyperText Markup Language)** Langage de structuration des contenus web, HTML définit le squelette d'une page à travers un système de balises. Il organise les éléments textuels, multimédias et les liens hypertextes. **CSS (Cascading Style Sheets)** Langage de stylisation, CSS sépare la présentation du contenu HTML. Il permet de contrôler l'aspect visuel (couleurs, polices, mise en page) et les effets graphiques, y compris les designs responsifs adaptatifs. **JavaScript (JS) :**Langage de programmation dynamique qui ajoute l'interactivité aux pages web. Exécuté côté navigateur, il permet de manipuler le DOM, gérer les événements utilisateurs et communiquer avec les backends. **TypeScript (TS)** Surcouche typée de JavaScript développée par Microsoft, TypeScript apporte une syntaxe plus rigoureuse avec un système de types statiques, facilitant le développement d'applications complexes tout en étant transpilé en JS standard.

0.11.2.2 Frameworks et Plugins : Outils Clés pour des Interfaces Dynamiques

Angular et Angular Material : Framework complet pour le développement d'applications web complexes, Angular offre une structure modulaire et une gestion optimisée des données. Couplé à Angular Material, il intègre des composants UI prêts à l'emploi (boutons, menus, grilles) alignés sur les principes du Material Design, idéal pour des interfaces cohérentes et accessibles . Comparaison entre Angular et React

Table 4 – Comparaison entre Angular+Material et React

Critère	Angular	React
Type	Framework complet	Bibliothèque UI
Architecture	MVC structuré	Composants isolés
UI Components	Angular Material intégré	Choix libre (MUI, etc.)
Courbe d'apprentissage	Raide (TypeScript, RxJS)	Plus douce
Data Binding	Bidirectionnel par défaut	Unidirectionnel
Gestion d'état	Services + RxJS	Context/Redux/etc.
Performances	Optimisé (Change Detection)	Virtuel DOM
SSR	Angular Universal	Next.js
Mobile	Ionic/Capacitor	React Native
Écosystème	Intégré (CLI, etc.)	Modular (choix libres)
Cas typiques	Applications d'entreprise	SPAs, applications modulaires

Highcharts : Bibliothèque spécialisée dans la création de graphiques (lignes, barres, camemberts) avec une interactivité fluide (zooms, tooltips). Son atout réside dans sa simplicité d'intégration et sa compatibilité multiplateforme, adaptée aux tableaux de bord analytiques. **amCharts :** Solution moderne pour des visualisations complexes (cartes géographiques, diagrammes animés). Optimisé pour les jeux de données volumineux, il mise sur des animations fluides et des options de personnalisation avancées, parfait pour des applications financières ou scientifiques. **SweetAlert :** plugin léger pour remplacer les alertes natives du navigateur par des modaux esthétiques et configurables (icônes, boutons personnalisés, chargements). Il améliore l'expérience utilisateur grâce à des transitions élégantes et une ergonomie intuitive.

0.11.3 La Technologie des Bases de données

0.11.3.1 MySQL :

Système de gestion de base de données relationnelle (SGBDR) open source, MySQL est réputé pour sa fiabilité, sa rapidité et sa simplicité d'utilisation. Il est largement adopté pour les applications web grâce à :

Modèle relationnel : Stockage structuré avec tables, clés primaires et étrangères.

Transactions ACID : Garantie de cohérence des données (atomicité, isolation, etc.).

Réplication : Synchronisation multi-serveurs pour la haute disponibilité.
Communauté active : Documentation abondante et support technique étendu.

Comparaison entre MySQL et PostGreSQL

Table 5 – Comparaison entre MySQL et PostgreSQL

Critère	**MySQL**	**PostgreSQL**
Modèle de Données	Relationnel strict	Relationnel + JSON, XML, etc.
Scalabilité	Verticale (monolithique)	Horizontale (sharding)
Transactions	ACID (InnoDB)	ACID + MVCC (gestion fine)
Extensions	Limitées	Extensions riches (pgvector...)
Cas d'Usage	Applications web simples	Systèmes complexes, analytique

0.11.3.2 pgvector (Base de Données Vectorielle)

Extension de PostgreSQL, pgvector ajoute des fonctionnalités de stockage et de recherche de vecteurs (représentations numériques de données complexes comme du texte ou des images). Idéal pour l'IA et le Machine Learning, il permet : **Similarité vectorielle** : Recherche de proximité (ex : recommandations personnalisées) ; **Intégration native** : Compatibilité avec les fonctionnalités existantes de PostgreSQL (requêtes SQL, jointures). **Performance optimisée** : Indexation avancée (IVFFlat, HNSW) pour des requêtes rapides sur de grands jeux de données. Le Tableau suivant nous englobe une comparaison entre Pgvector Store de PostgreSQL avec une BD NoSQL MongoDB

Table 6 – Comparaison entre pgvector et NoSQL Vectoriel

Critère	**pgvector (PostgreSQL)**	**NoSQL Vectoriel (ex : MongoDB)**
Structure	Hybride (relationnel + vecteurs)	Orienté documents/vecteurs
Scalabilité	Verticale (limités en volume)	Horizontale (big data)
Requêtes	SQL + opérateurs de similarité	Langage propriétaire (MQL...)
Intégration IA	Via extensions (ML libraries)	Native (pipelines de ML)
Cas d'Usage	Applications hybrides (SQL + IA)	Systèmes d'IA à grande échelle

0.11.4 Les Outils Serveurs pour le Monitoring et la Gestion Distribuée

Les architectures modernes, en particulier celles basées sur des microservices, nécessitent des outils spécialisés pour le monitoring/Observabilité, la gestion des configurations, la traçabilité et la sécurité. Ainsi, dans notre application, les serveurs utilisés sont :

0.11.4.1 Prometheus

Système de monitoring et d'alerte open-source conçu pour les environnements dynamiques. Il collecte et stocke des métriques sous forme de séries temporelles, avec un modèle de pull basé sur HTTP. Idéal pour Kubernetes et les architectures cloud-native.

0.11.4.2 Grafana

Plateforme de visualisation qui s'intègre parfaitement avec Prometheus et d'autres sources de données. Permet de créer des tableaux de bord interactifs pour analyser les performances système en temps réel.

0.11.4.3 Traçabilité des Requêtes : Zipkin

Outil de distributed tracing qui aide à identifier les goulots d'étranglement dans les architectures microservices. Compatible avec Spring Cloud Sleuth, il permet de suivre une requête à travers plusieurs services.

0.11.4.4 Gestion des Messages et Événements : Kafka

Plateforme de streaming d'événements distribuée, utilisée pour la communication asynchrone entre services, le traitement de flux en temps réel et l'intégration de systèmes hétérogènes.

0.11.4.5 Découverte de Services : Eureka (Netflix OSS)

Serveur de service discovery pour architectures microservices. Permet aux services de s'enregistrer et de découvrir d'autres services dynamiquement, éliminant le besoin de configuration statique.

0.11.5 Les models AI et frameworks basé sur Machine Learning

0.11.5.1 Assistant OpenAI

Comme il est définit, l'application a un service dédié au chatbot intélligent qui est assisté par une IA OpenAI. Ce dernier représente l'état de l'art en matière de modèles de langage (LLM). Ses caractéristiques principales incluent : Compréhension contextuelle avancée, Génération de texte cohérent et structuré. Ce assistant a pour mission typique de : Chatbots intelligents, génération de contenu, Analyse de documents.

0.11.5.2 Prophet de Facebook basé sur la Machine Learning

C'est un Framework spécialisé dans les séries temporelles offrant une modélisation automatique des tendances aussi une prise en compte des saisonnalités et une robustesse aux données manquantes ainsi qu'une interface simple avec Python/R. Cet outils est simple d'utilisation et très pratique car il utilise de Modèle additif généralisé et sa précision RMSE est de 10 - 15% par rapport aux autres frameworks comme statsmodels (8-12%) et Kats qui est 5 à 18% comme on peut le voir dans le tableau comparatif ci-dessous :

TABLE 7 – Comparaison des outils d'analyse temporelle (ML implémentés et performances)

Critère	Prophet (Meta)	statsmodels (Python)	Kats (Meta)
Algorithmes ML	Modèle additif généralisé (GAM) Régression bayésienne Changepoint detection (ML)	SARIMA (statistique) VAR/VECM Holt-Winters (ETS)	Modèles LSTM (via TensorFlow) Kernel Regression Detection de motifs (k-NN)
Précision	RMSE : 10-15% sur données saisonnières Meilleur sur court/moyen terme	RMSE : 8-12% (si paramétrage optimal) Stable en long terme	RMSE : 5-18% (selon modèle) Optimal pour séries complexes
Temps d'entraînement	2-5 min (100K points) Scalable avec Stan	1-30 min (selon modèle) VAR lent sur big data	3-20 min (LSTM gourmand) Parallélisation possible
Features clés ML	Auto-sélection des saisonnalités Robustesse aux outliers	Tests ADF/KPSS automatisés Confidence intervals	Feature engineering temporel Meta-learning pour modèle optimal
Limites ML	Faible adaptabilité aux tendences non-linéaires complexes	Requiert expertise statistique	Overfitting possible sur petits datasets

0.12 Présentation De L'environnement De Travail

0.12.1 Environnement logiciel

0.12.1.1 Système d'exploitation

Le système d'exploitation utilisé pour effectué notre travail est le Windows 11 qui est un système.

0.12.1.2 Environnement de développement (IDE)

L'environnement de développement regroupe l'ensemble des outils et logiciels utilisés pour concevoir, coder, tester et déployer une application. Il permet au développeur de travailler dans un cadre structuré et cohérent, en facilitant l'intégration

des différentes technologies nécessaires au bon déroulement du projet.

IntelliJ : IntelliJ est un environnement de développement intégré (IDE) très puissant, particulièrement adapté aux projets Java. Il offre de nombreuses fonctionnalités comme l'autocomplétion intelligente, le débogage, l'intégration avec les frameworks populaires et une excellente gestion des projets Maven ou Gradle.

VS Code : Visual Studio Code, ou VS Code, est un éditeur de code source léger, multiplateforme et très extensible. Il prend en charge de nombreux langages de programmation et propose une vaste bibliothèque d'extensions permettant de personnaliser l'environnement de développement selon les besoins du projet.

Un tableau comparatif des ces IDE avec Eclispe

Table 8 – Comparaison des IDE de développement

Critère	**IntelliJ IDEA**	**VS Code**	**Eclipse**
Type	IDE complet	Éditeur léger + extensions	IDE modulaire
Langages	Java, Kotlin, Python (Ultimate)	Multi-langages (extensions)	Java, C++, Python, etc.
Performances	Lourd mais puissant	Léger et rapide	Moyenne, peut être lent
Extensibilité	Plugins officiels	Marketplace très riche	Large écosystème de plugins
Debugging	Outils avancés intégrés	Bon via extensions	Completx mais complet
Intégration Git	Excellente	Excellente (extension)	Bonne (EGit)
Licence	Commercial (gratuit Community)	Gratuit et open-source	Gratuit et open-source
Cas d'usage	Développement Java/Android	Développement web/scripting	Développement Java/EE

Git Git est un système de gestion de version distribué qui permet de suivre l'évolution du code source, de collaborer efficacement en équipe et de gérer les différentes branches du développement. Il est essentiel pour organiser le travail sur des projets logiciels complexes et assurer la traçabilité des modifications.

Docker-Desktop Docker Desktop est une application qui permet de créer, déployer et gérer des conteneurs sur un poste de travail. Elle est utilisée pour simuler des environnements d'exécution isolés et reproductibles, ce qui facilite le développement, les tests et le déploiement des applications dans divers contextes.

Navigateur (Firefox Mozilla, Edge, Chrome) Les navigateurs comme Firefox Mozilla, Microsoft Edge et Google Chrome sont utilisés pour tester et

visualiser les applications web. Ils permettent de vérifier le rendu graphique, la compatibilité entre navigateurs et d'utiliser les outils de développement intégrés pour analyser les performances, le code HTML/CSS et les requêtes réseau.

0.12.2 Environnement matériel

Les matériels utilisés pour la réalisation de cette application sont :

Ordinateurs portables HP

- **HP EliteBook** avec les spécifications suivantes :
  - Processeur : Intel Core i5-8365U CPU @ 1.60GHz 1.90 GHz
  - Mémoire RAM : 16 Go
  - Connectique : 2 ports USB 3.0 + 1 port USB 3.0 Type C
  - Stockage : Disque dur 256 Go
- **HP Pavilion 15** avec les spécifications suivantes :
  - Processeur : Intel Core i3-1005G1 CPU @ 1.60GHz
  - Mémoire RAM : 12 Go
  - Connectique : 2 ports USB 3.0 + 1 port USB 3.0 Type C
  - Stockage : Disque dur 256 Go

Appareils mobiles pour tests

- Téléphone portable Android : Samsung Galaxy A14 (tests de responsivité)

Périphériques complémentaires

- 1 modem WiFi de marque MTN pour la connexion internet
- 1 écran externe de marque HP pour améliorer la productivité

0.13 Coût De Réalisation du projet