JEE

# Rapport sur le Projet de Gestion de chambres d'hotel

**SLAOUI YASSINE**

# Introduction

Ce rapport documente le projet de développement d'un système de gestion de réservation de chambres d'hôtel en utilisant les fonctionnalités offertes par le framework Spring. L'objectif principal de ce projet était de fournir une plateforme permettant aux utilisateurs de rechercher, réserver et gérer des chambres d'hôtel en ligne, tout en garantissant une expérience fluide et sécurisée. Ce rapport détaille les composants clés du projet, les fonctionnalités principales, ainsi que les technologies et outils utilisés.

# Composants Clés du Projet

## 1. Spring Boot

- Utilisation de Spring Boot pour simplifier la configuration et le déploiement de l'application.
- Création d'une application autonome prête à être exécutée.

## 2. Spring MVC

- Implémentation du modèle MVC pour gérer les interactions utilisateur.
- Création de contrôleurs pour gérer les opérations telles que la recherche de chambres, la réservation, etc.

## 3. Spring Data

- Utilisation de Spring Data JPA pour interagir avec la base de données.

- Mappage des entités Java aux tables de la base de données pour stocker les informations sur les chambres, les réservations.

## 4. Spring Security

- Spring Security: Garantit la sécurité de la plateforme.
- Utilisation de JSON Web Tokens (JWT): Mécanisme d'authentification stateless.
- JWT encapsule les informations d'identification de l'utilisateur.
- Vérification de la validité du jeton à chaque requête HTTP.
- Autorisation des accès en fonction des informations du jeton.
- Renforce la sécurité en limitant l'accès aux fonctionnalités uniquement aux utilisateurs autorisés.
-

## 5. Postman

- Utilisation de Postman pour tester les API développées.
- Création de collections d'API et de scénarios de test pour vérifier le bon fonctionnement de l'application.

## 6. Swagger

- Configuration de Swagger pour générer une documentation interactive des API.
- Les utilisateurs peuvent consulter facilement la documentation des API et tester les appels directement depuis l'interface Swagger.

### 7.Base de Données

Pour ce projet, nous utiliserons une base de données relationnelle comme  PostgreSQL. Ces bases de données sont bien prises en charge par Spring Data JPA et offrent une structure de données adaptée pour stocker les informations sur les chambres d'hôtel, les réservations, les utilisateurs, et le reste des entités.

## Fonctionnalités Principales

## Les fonctionnalités du Guest :

#### Obtenir tous les invités :

- List<GuestDTO> getAllGuests()
- Fonctionnalité permettant de récupérer la liste de tous les invités enregistrés dans le système.

#### Rechercher des invités par mot-clé :

- Page<GuestDTO> getGuestsByKeyword(String name, int page, int size)
- Fonctionnalité permettant de rechercher des invités en fonction d'un mot-clé (nom) avec une pagination pour les résultats.
-

### Obtenir tous les invités avec pagination :

- Page<GuestDTO> getAllGuests(int page, int size)
- Fonctionnalité permettant de récupérer tous les invités avec une pagination pour les résultats.

### Obtenir un invité par e-mail :

- GuestDTO getGuestByEmail(String email)
- Fonctionnalité permettant de récupérer un invité en fonction de son adresse e-mail.

### Obtenir un invité par nom d'utilisateur :

- GuestDTO getGuestByUsername(String username)
- Fonctionnalité permettant de récupérer un invité en fonction de son nom d'utilisateur.

### Obtenir l'invité d'une critique :

- GuestDTO getGuestOfReview(Long reviewId)
- Fonctionnalité permettant de récupérer l'invité associé à une critique spécifique.

### Créer un nouvel invité :

- GuestDTO createGuest(GuestDTO guestDTO)
- Fonctionnalité permettant de créer un nouvel invité dans le système à partir des informations fournies dans l'objet GuestDTO.

### Mettre à jour un invité existant :

- GuestDTO updateGuest(GuestDTO guestDTO)
- Fonctionnalité permettant de mettre à jour les informations d'un invité existant dans le système en utilisant les données fournies dans l'objet GuestDTO.

### Obtenir un invité par ID :

- Guest getGuestById(Long guestId)
- Fonctionnalité permettant de récupérer un invité spécifique en fonction de son ID.

### Supprimer un invité :

- void deleteGuest(Long guestId)
- Fonctionnalité permettant de supprimer un invité spécifique du système en fonction de son ID.

## Les fonctionnalités des Réservations:

### Obtenir une réservation par ID :

- Reservation getReservationById(Long reservationId)
- Permet de récupérer une réservation spécifique à partir de son ID.

### Obtenir toutes les réservations avec pagination :

- Page<ReservationDTO> getAllReservations(int page, int size)

- Permet de récupérer toutes les réservations avec une pagination pour les résultats.

### Obtenir les réservations d'un invité pour une chambre spécifique avec pagination :

- Page<ReservationDTO> getReservationsByGuestAndRoom(Long guestId, Long roomId, int page, int size)
- Permet de récupérer les réservations d'un invité pour une chambre spécifique avec une pagination pour les résultats.

### Obtenir les réservations d'un invité avec pagination :

- Page<ReservationDTO> getReservationsByGuest(Long guestId, int page, int size)
- Permet de récupérer les réservations d'un invité avec une pagination pour les résultats.

### Obtenir les réservations d'une chambre avec pagination :

- Page<ReservationDTO> getReservationsByRoom(Long roomId, int page, int size)
- Permet de récupérer les réservations d'une chambre spécifique avec une pagination pour les résultats.

### Obtenir les réservations se terminant à une date spécifique avec pagination :

- Page<ReservationDTO> getReservationsEndingOn(Date endDate, int page, int size)

- Permet de récupérer les réservations se terminant à une date spécifique avec une pagination pour les résultats.

### Obtenir les réservations commençant à une date spécifique avec pagination :

- Page<ReservationDTO> getReservationsStartingOn(Date startDate, int page, int size)
- Permet de récupérer les réservations commençant à une date spécifique avec une pagination pour les résultats.

### Obtenir les réservations commençant à partir d'une date spécifique avec pagination :

- Page<ReservationDTO> getReservationsStartingFrom(Date startDate, int page, int size)
- Permet de récupérer les réservations commençant à partir d'une date spécifique avec une pagination pour les résultats.

### Obtenir une réservation d'un invité pour une chambre spécifique à une date d'arrivée spécifique :

- ReservationDTO getReservationByGuestAndRoomAndCheckInDate(Long guestId, Long roomId, Date checkInDate)
- Permet de récupérer une réservation spécifique d'un invité pour une chambre donnée à une date d'arrivée spécifique.

### Créer une nouvelle réservation :

- ReservationDTO createReservation(ReservationDTO reservationDTO)
- Permet de créer une nouvelle réservation en utilisant les informations fournies dans l'objet ReservationDTO.

### Mettre à jour une réservation existante :

- ReservationDTO updateReservation(ReservationDTO reservationDTO)
- Permet de mettre à jour les informations d'une réservation existante en utilisant les données fournies dans l'objet ReservationDTO.

### Supprimer une réservation :

- void deleteReservation(Long reservationId)
- Permet de supprimer une réservation spécifique du système en fonction de son ID.

### Ajouter une critique à une réservation :

- void addReviewToReservation(Long reservationId, Long reviewId)
- Permet d'associer une critique spécifique à une réservation.

### Vérifier la disponibilité d'une chambre à une date spécifique :

- boolean isRoomAvailable(Long roomId, Date checkInDate, Date checkOutDate)
- Permet de vérifier si une chambre est disponible pour réservation à une date spécifique.

### Obtenir les réservations qui se chevauchent avec une période de réservation spécifique pour une chambre donnée :

- List<ReservationDTO> getOverlappingReservations(Long roomId, Date checkInDate, Date checkOutDate)

- Permet de récupérer les réservations qui se chevauchent avec une période de réservation spécifique pour une chambre donnée.

### Obtenir les réservations qui se chevauchent avec une période de réservation spécifique pour une chambre donnée, à l'exclusion d'une réservation spécifique :

- List<ReservationDTO> getOverlappingReservationsExcludingReservation(Long reservationId, Long roomId, Date checkInDate, Date checkOutDate)
- Permet de récupérer les réservations qui se chevauchent avec une période de réservation spécifique pour une chambre donnée, en excluant une réservation spécifique.

### Mise à jour de la disponibilité des chambres à une heure spécifique (Tâche planifiée) :

- void updateRoomsAvailability()
- Permet de mettre à jour la disponibilité des chambres à une heure spécifique, planifiée par une tâche planifiée.

## Les fonctionnalités des Reviews:

### Obtenir une critique par ID :

- Review getReviewById(Long reviewId)
- Permet de récupérer une critique spécifique à partir de son ID.

### Obtenir toutes les critiques avec pagination :

- Page<ReviewDTO> getAllReviews(int page, int size)

- Permet de récupérer toutes les critiques avec une pagination pour les résultats.

### Obtenir toutes les critiques (sans pagination) :

- List<Review> getAllReviews()
- Permet de récupérer toutes les critiques sans pagination.

### Obtenir les critiques d'une chambre avec pagination :

- Page<ReviewDTO> getRoomReviews(Long roomId, int page, int size)
- Permet de récupérer les critiques d'une chambre spécifique avec une pagination pour les résultats.

### Obtenir les critiques d'un invité avec pagination :

- Page<ReviewDTO> getGuestReviews(Long guestId, int page, int size)
- Permet de récupérer les critiques d'un invité spécifique avec une pagination pour les résultats.

### Créer une nouvelle critique :

- ReviewDTO createReview(ReviewDTO review)
- Permet de créer une nouvelle critique en utilisant les informations fournies dans l'objet ReviewDTO.

### Mettre à jour une critique existante :

- ReviewDTO updateReview(ReviewDTO review)

- Permet de mettre à jour les informations d'une critique existante en utilisant les données fournies dans l'objet ReviewDTO.

### Supprimer une critique :

- void deleteReview(Long reviewId)
- Permet de supprimer une critique spécifique du système en fonction de son ID.

## Les fonctionnalités des Rôles:

### Créer un rôle :

- Role createRole(String roleName)
- Permet de créer un nouveau rôle avec le nom spécifié.

### Obtenir un rôle par son nom :

- Role getRole(String roleName)
- Permet de récupérer un rôle spécifique à partir de son nom.

## Les fonctionnalités des Chambres:

### Obtenir une chambre par ID :

- Room getRoomById(Long roomId)
- Permet de récupérer une chambre spécifique à partir de son ID.

### Obtenir toutes les chambres avec pagination :

- Page<RoomDTO> getAllRooms(int page, int size)

- Permet de récupérer toutes les chambres avec une pagination pour les résultats.

### Obtenir les chambres en fonction de filtres spécifiques avec pagination :

- Page<RoomDTO> getRoomsByFilters(List<FilterDTO> filters, int page, int size)
- Permet de récupérer les chambres en fonction de filtres spécifiques avec une pagination pour les résultats.

### Obtenir les nouvelles chambres pour un invité avec pagination :

- Page<RoomDTO> getNewRoomsForGuest(Long guestId, int page, int size)
- Permet de récupérer les nouvelles chambres disponibles pour un invité spécifique avec une pagination pour les résultats.

### Obtenir les chambres précédemment réservées par un invité avec pagination :

- Page<RoomDTO> getPreviouslyBookedRoomsForGuest(Long guestId, int page, int size)
- Permet de récupérer les chambres précédemment réservées par un invité spécifique avec une pagination pour les résultats.

### Obtenir toutes les chambres (sans pagination) :

- List<RoomDTO> getAllRooms()
- Permet de récupérer toutes les chambres sans pagination.

### Créer une nouvelle chambre :

- RoomDTO createRoom(RoomDTO roomDTO)
- Permet de créer une nouvelle chambre en utilisant les informations fournies dans l'objet RoomDTO.

### Mettre à jour une chambre existante :

- RoomDTO updateRoom(RoomDTO roomDTO)
- Permet de mettre à jour les informations d'une chambre existante en utilisant les données fournies dans l'objet RoomDTO.

### Supprimer une chambre :

- void deleteRoom(Long roomId)
- Permet de supprimer une chambre spécifique du système en fonction de son ID.

## Les fonctionnalités des Users:

### Obtenir tous les utilisateurs avec pagination :

- Page<UserDTO> getAllUsers(int page, int size)
- Permet de récupérer tous les utilisateurs avec une pagination pour les résultats.

### Obtenir un utilisateur par son ID :

- User getUserById(Long userId)

- Permet de récupérer un utilisateur spécifique à partir de son ID.

### Obtenir un utilisateur par son adresse e-mail :

- User getUserByEmail(String email)
- Permet de récupérer un utilisateur spécifique à partir de son adresse e-mail.

### Obtenir un utilisateur par son nom d'utilisateur :

- User getUserByUsername(String username)
- Permet de récupérer un utilisateur spécifique à partir de son nom d'utilisateur.

### Obtenir un utilisateur par son nom d'utilisateur ou son adresse e-mail :

- User getUserByUsernameOrEmail(String usernameOrEmail)
- Permet de récupérer un utilisateur spécifique à partir de son nom d'utilisateur ou de son adresse e-mail.

### Obtenir un utilisateur par son nom d'utilisateur, son adresse e-mail ou son ID :

- User getUserByUsernameOrEmailOrId(String keyword)
- Permet de récupérer un utilisateur spécifique à partir de son nom d'utilisateur, de son adresse e-mail ou de son ID.

### Créer un nouvel utilisateur avec nom d'utilisateur, adresse e-mail et mot de passe :

- User createUser(String username, String email, String password)
- Permet de créer un nouvel utilisateur en spécifiant son nom d'utilisateur, son adresse e-mail et son mot de passe.

### Créer un nouvel utilisateur avec adresse e-mail et mot de passe :

- User createUserWithEmail(String email, String password)
- Permet de créer un nouvel utilisateur en spécifiant son adresse e-mail et son mot de passe.

### Créer un nouvel utilisateur avec nom d'utilisateur et mot de passe :

- User createUserWithUsername(String username, String password)
- Permet de créer un nouvel utilisateur en spécifiant son nom d'utilisateur et son mot de passe.

### Mettre à jour un utilisateur :

- User updateUser(User user)
- Permet de mettre à jour les informations d'un utilisateur existant en utilisant l'objet User fourni.

### Attribuer un rôle à un utilisateur :

- void assignRoleToUser(String keyword, String roleName)
- Permet d'attribuer un rôle spécifié à un utilisateur spécifique en utilisant son nom d'utilisateur, son adresse e-mail ou son ID.

### Révoquer un rôle d'un utilisateur :

- void revokeRoleFromUser(String keyword, String roleName)
- Permet de révoquer un rôle spécifié d'un utilisateur spécifique en utilisant son nom d'utilisateur, son adresse e-mail ou son ID.

### Supprimer un utilisateur :

- void deleteUser(Long userId)
- Permet de supprimer un utilisateur spécifique du système en fonction de son ID.

## Conclusion

Ce projet offrira une plateforme complète de gestion de réservation de chambres d'hôtel, développée en utilisant les fonctionnalités avancées de Spring Framework. L'utilisation de Spring Boot, Spring MVC, Spring Data et Spring Security garantira la robustesse, la sécurité et la flexibilité de l'application. De plus, les API seront testées avec Postman et une documentation interactive pourra être fournie via Swagger, offrant ainsi une expérience utilisateur optimale. En option, le frontend pourra être développé en React ou Angular, pour une expérience utilisateur moderne et réactive.