DocumentationTechniqueDuProjetFinal

Informations Générales

Nom du Projet	Gestion d'Hôtel (hotelmanagementsystem)
Chef de Projet	Kanmegne Tabouguie Andre Yvan
Équipe	Kanmegne Tabouguie et Marc
GitHub Repo	Lien vers le repo

Table des matières

1-MVP

- **User Authentication** : Inscription, connexion et déconnexion des utilisateurs.
- **Authorization** : Protection des routes basées sur les rôles (admin ou guest).
- **Room Management** : Création, mise à jour, suppression et consultation des chambres par les administrateurs.
- **User Management** : Création, mise à jour, suppression et consultation des utilisateurs par les administrateurs.
- **Les utilisateurs peuvent parcourir les différentes chambres et cliquer sur leurs favorites.**

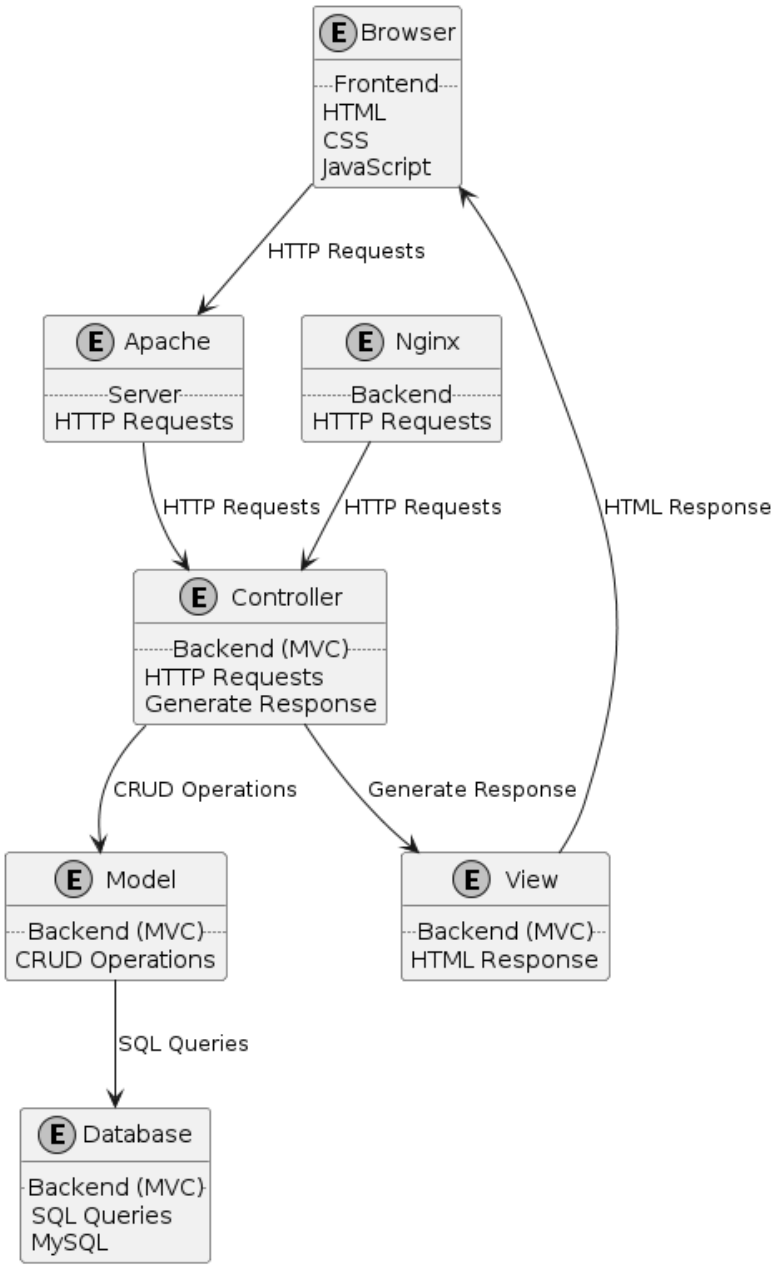
- **Token Invalidation** : Invalidation des tokens à la déconnexion.
- **Caching** : Mise en cache des données des chambres et des utilisateurs.
- **chatbox partie utilisateur pour repondre question**

2-Vue d'ensemble de l'architecture

Description générale

Ce projet est une application de gestion des utilisateurs et chambre de l’hôtel. Elle comprend plusieurs composants principaux : l'API backend, la base de données, et le serveur web.

Diagramme de l'architecture



3-Prototype

https://www.figma.com/design/rpyd0cU8pAXVsravue9Ox3/hotemanagement?node-id=0-1&t=0NS0erGixICCBJqe-1

4-Description des composants principaux

API Backend(Authentication JWT, memcached)

L'API backend est construite en utilisant Node.js et Express. Elle expose des endpoints pour gérer les utilisateurs, les chambres.

Base de données(Mysql)

La base de données utilise Mysql(php myAdmin) pour stocker les informations des utilisateurs, des chambres, des réservations, et des paiements.

Infrastructure(Vmware and Ubuntu server)

L'infrastructure du projet comprend plusieurs composants qui assurent la mise en œuvre, la disponibilité et la sécurité de l'application. Voici une description détaillée de chaque composant :

Serveur Web

- **Nginx** est utilisé comme serveur web et proxy inverse pour le backend. Il permet de distribuer les requêtes entrantes au backend Node.js.
- **Apache** est utilisé pour servir les fichiers statiques du frontend. Apache est un serveur HTTP flexible et performant, idéal pour héberger des fichiers HTML, CSS et JavaScript.

Environnement d'exécution

- **Node.js avec express** est utilisé pour exécuter le backend. Node.js est une plateforme JavaScript côté serveur qui permet de construire des applications web performantes et évolutives.

Base de données

- **MySQL** est utilisé pour gérer la base de données relationnelle. MySQL est un système de gestion de base de données populaire pour sa fiabilité et sa performance.

Gestion des versions

- **Git** est utilisé pour le contrôle de version du code source. Le dépôt Git permet de suivre les modifications du code, de collaborer avec d'autres développeurs et de gérer les versions du projet.

Sécurité

- **SSL/TLS** est utilisé pour sécuriser les communications entre les clients et le serveur web.
- **Firewall** et **Security Groups** sont configurés pour contrôler l'accès réseau et protéger l'infrastructure contre les menaces externes.

5-Modèle Conceptuel de Données (MCD)

Description des entités

User Table

- **id** (Primary Key)
- **username**
- **password** (hashed)
- **email**
- **role** (e.g., admin, guest)

Room Table

- **id** (Primary Key)
- **room_number**
- **type**
- **price_per_night**
- **availability**
- **image** (URL pointing to room image)

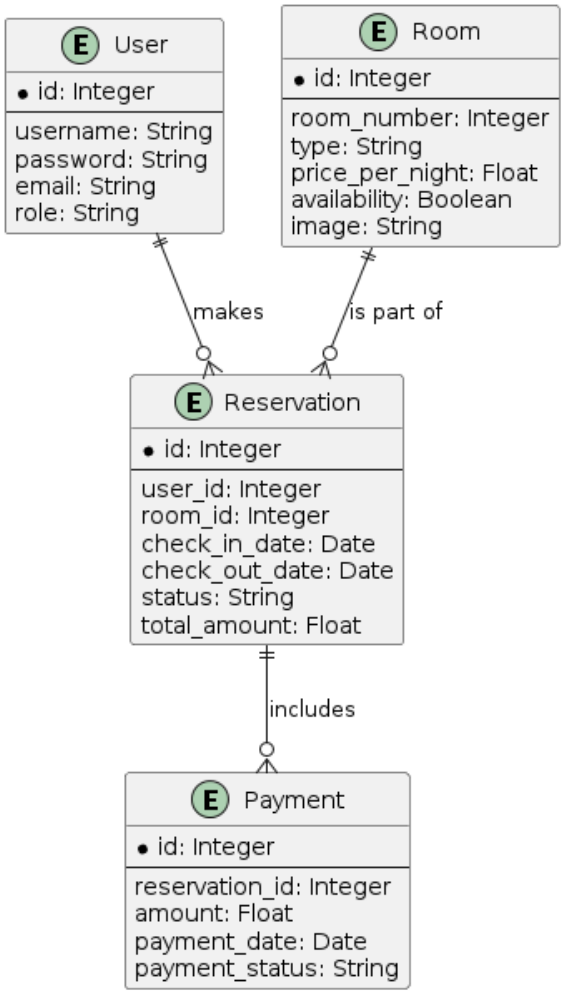
Reservation Table

- **id** (Primary Key)
- **user_id** (Foreign Key referencing User Table)
- **room_id** (Foreign Key referencing Room Table)
- **check_in_date**
- **check_out_date**
- **status** (e.g., pending, confirmed, checked_in, checked_out)
- **total_amount**

Payment Table

- **id** (Primary Key)
- **reservation_id** (Foreign Key referencing Reservation Table)
- **amount**
- **payment_date**
- **payment_status** (e.g., pending, successful, failed)

Diagramme UML



6-Détails du code

Structure du projet

Backend:

Afin d'optimiser la modularité et la fluidité du processus de développement, l'architecture du backend s'appuie sur le modèle de conception MVC (Modèle-Vue-Contrôleur). De plus, les principes de la programmation orientée objet (POO) ont été rigoureusement appliqués dans l'élaboration du code.

```
Backend/
│
├── config/
│   ├── db.js          // Database configuration
│
├── controllers/
```

```
│   ├── authController.js   // Authentication controller
│   ├── roomController.js   // Room-related controller
│   ├── userController.js  // user-related controller
│
├── models/
│   ├── user.js        // User model
│   ├── room.js        // Room model
│
├── routes/
│   ├── authRoutes.js   // Authentication routes
│   ├── roomRoutes.js   // Room-related routes
│   ├── userRoutes.js  // user-related routes
│
├── middleware/
│   └── authMiddleware.js   // Authentication middleware
│
├── services/
│   ├── authService.js     // Authentication service
│   ├── roomService.js     // Room-related service
│   ├── userService.js  // user-related service
│
├── .env
├-- cache.js
├── tokenStore.js
├── hotelmanagement.sql (database)
├── index.js          // Main entry point of the application
└── package.json      // Dependency and script management
```

Explication

Le modèle MVC permet de séparer les préoccupations de manière efficace :

- **Modèles (Models)** : représentent les données de l'application et les règles de gestion qui s'y appliquent. Par exemple, `user.js` et `room.js`.
- **Contrôleurs (Controllers)** : traitent les requêtes entrantes, interagissent avec les modèles et renvoient les vues appropriées. Par exemple, `authController.js`, `roomController.js`, `userController.js`.

Cette structure facilite la maintenance du code, permet de travailler en parallèle sur différents composants du projet et rend le développement plus structuré et modulable.

Frontend(HTML,Boostrap,css,Javascript)

Le frontend de ce projet est structuré pour offrir une interface utilisateur intuitive et responsive. Il est composé de plusieurs composants et scripts, chacun ayant un rôle spécifique pour garantir une expérience utilisateur fluide et interactive. Voici un aperçu de la structure du frontend :

```
frontend
├── components
│   ├── Assets
│   ├── css
│   ├── admindashboard.html
│   ├── create_room.html
│   ├── create_user.html
│   ├── guestdashboard.html
│   ├── login.html
│   ├── partialUpdateRoom.html
│   ├── register.html
│   ├── rooms.html
│   ├── updateRoom.html
│   ├── update_user.html
│   ├── users.html
├── scripts
│   ├── clock.js
│   ├── create_room.js
│   ├── createanewuser.js
│   ├── emailDomainAnalysis.js
│   ├── guestsectionroom.js
│   ├── histogramroom.js
│   ├── livechat.js
│   ├── login.js
│   ├── logout.js
│   ├── map.js
│   ├── news.js
│   ├── notewidget.js
│   ├── partialUpdateRoom.js
│   ├── register.js
│   ├── roomChart.js
│   ├── rooms.js
│   ├── script1.js
│   ├── search.js
│   ├── updateRoom.js
│   ├── updateuserdata.js
│   ├── userRoleDistribution.js
```

```
│   ├── users.js
│   ├── weather.js
```

Explication

- **Composants (Components)** : Ils comprennent les fichiers HTML et CSS nécessaires pour afficher les différentes pages et éléments de l'application. Les pages HTML telles que `admindashboard.html`, `create_room.html`, et `login.html` fournissent les interfaces utilisateur pour les différentes fonctionnalités de l'application.
- **Scripts** : Les fichiers JavaScript sont utilisés pour ajouter des fonctionnalités dynamiques à l'application. Par exemple, `create_room.js` gère la logique de création de chambres, tandis que `livechat.js` permet la fonctionnalité de chat en direct.

7-API Documentation

API Endpoint Documentation

8-Infrastructure et Reseaux Documentation

Documentation Configuration d'Infrastructure