***Makouar Abderrahim***

***Hamza Lachgar***

***Ayoub Maghrani***

***Abdelghani Aboumada***

**Ali Ghanam**

# 1. Architecture du projet

**Vue d’ensemble du projet**

Le projet consiste en une plateforme de gestion des réservations d’hôtels, basée sur une architecture multi-couches et compatible avec plusieurs protocoles d’API : REST, SOAP, GraphQL et gRPC. L’objectif principal est d’évaluer ces technologies en termes de performance, scalabilité, simplicité d’implémentation.

**Composants principaux**

* **Web Layer** : Gère les requêtes des clients via des protocoles REST, GraphQL, SOAP et gRPC.
* **DTO** : des objets légers utilisés pour transporter des données entre les différentes couches de l'application
* **Mapper** : responsables de la conversion entre les objets métier (entités) et les DTO.
* **Business Layer** : Implémente la logique métier via des services .
* **DAO Layer** : Communique avec la base de données via des répositories JPA.
* **Base de données** : H2 pour le stockage des entités (Client, Chambre, Réservation).

L’application est déployée sur un serveur Tomcat avec Spring IoC pour la gestion des dépendances.

# 2. Structure du projet

**Structure du projet**

L'organisation des fichiers et des packages est la suivante :

**Description des composants**

1. **Main**
   * **HotelApplication.java** : Point d'entrée principal de l'application Spring Boot.
2. **Controller**
   * Contient les classes responsables de la gestion des requêtes REST pour chaque entité (é.g., Chambre, Client, Réservation).
   * Exemple : ChambreController.java gère les opérations CRUD liées aux chambres.
3. **DTO (Data Transfer Object)**
   * Les classes DTO permettent de séparer les entités internes des objets transmis via les API.
   * Exemple : ChambreRequestDTO.java est utilisé pour les requêtes, et ChambreResponseDTO.java pour les réponses.
4. **Entity**
   * Représente les objets persistants de la base de données.
   * Exemple : Chambre.java modélise la table des chambres.
5. **Mapper**
   * Contient les classes pour convertir entre les entités et les DTO.
   * Exemple : ChambreMapper.java transforme un Chambre en ChambreResponseDTO.
6. **Repository**
   * Fournit des interfaces pour l'accès aux données, étendues de Spring Data JPA.
   * Exemple : ChambreRepository.java pour les opérations de la table Chambre.
7. **Services**
   * Contient la logique métier, séparée des contrôleurs et des repositories.
   * Exemple : ChambreService.java gère les règles de gestion liées aux chambres.
8. **Resources**
   * Contient les fichiers de configuration et les ressources statiques.
   * application.properties : Paramètres de configuration (e.g., connexion à la base de données).

# 3. Caractéristiques de la machine

**Ressources Matérielles**

* CPU : AMD rayzen 5
* RAM : 8 GB
* Stockage : SSD 250 GB

**Ressources Logicielles**

* **Système d'exploitation** : Windows 10
* **Serveur d'application** : Tomcat intégré à Spring Boot
* **Framework**:Spring Boot
* **Base de données** : H2
* **Outils de monitoring** : Prometheus et Grafana
* **Conteneurisation :**Docker
* **Langage de programmation** : Java 17
* **Outils de test :** JMeter

# 5- Resultat

**1. Performances : Temps de Réponse (Latence)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Taille du Message (KB) | Opération | REST (ms) | SOAP (ms) | GraphQL (ms) | gRPC (ms) |
| 1 KB | Créer | 15,4 | 25 | 20 | 10 |
|  | Consulter | 12,5 | 20 | 18 | 8 |
|  | Modifier | 14, | 22 | 19 | 9 |
|  | Supprimer | 13 | 21 | 18 | 8 |
| 10 KB | Créer | 18 | 35 | 25 | 12 |
|  | Consulter | 15 | 30 | 23 | 10 |
|  | Modifier | 17 | 33 | 24 | 11 |
|  | Supprimer | 16 | 31 | 23 | 10 |
| 100 KB | Créer | 25 | 60 | 35 | 18 |
|  | Consulter | 22 | 50 | 33 | 15 |
|  | Modifier | 24 | 55 | 34 | 16 |
|  | Supprimer | 23 | 53 | 33 | 15 |

**2. Performances : Débit (Throughput)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  |  |  | |  |  |  |  |   **Nombre de Requêtes Simultanées** | **REST (req/s)** | **SOAP (req/s)** | **GraphQL (req/s)** | **gRPC (req/s)** | | 10 | 500 | 300 | 400 | 700 | | 100 | 450 | 250 | 350 | 650 | | 500 | 400 | 200 | 300 | 600 | | 1000 | 350 | 150 | 250 | 550 | |

**3. Consommation des Ressources**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Nombre de Requêtes Simultanées** | **CPU REST (%)** | **CPU SOAP (%)** | **CPU GraphQL (%)** | **CPU gRPC (%)** | **Mémoire REST (MB)** | **Mémoire SOAP (MB)** | **Mémoire GraphQL (MB)** | **Mémoire gRPC (MB)** | | 10 | 15 | 20 | 18 | 12 | 50 | 70 | 60 | 45 | | 100 | 30 | 40 | 35 | 25 | 100 | 120 | 110 | 90 | | 500 | 50 | 70 | 60 | 40 | 200 | 300 | 250 | 180 | | 1000 | 70 | 90 | 80 | 60 | 350 | 500 | 450 | 300 | |

**4. Simplicité d’Implémentation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Critère** | **REST** | **SOAP** | **GraphQL** | **gRPC** |
| Temps d’implémentation (heures) | 8 | 15 | 12 | 10 |
| Nombre de lignes de code | 200 | 400 | 300 | 250 |
| Disponibilité des outils | Très bonne | Bonne | Très bonne | Bonne |
| Courbe d’apprentissage (jours) | 2 | 5 | 3 | 3 |

**5. Sécurité**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Critère** | **REST** | **SOAP** | **GraphQL** | **gRPC** | | Support TLS/SSL | Oui | Oui | Oui | Oui | | Gestion de l’authentification | Oui | Oui | Oui | Oui | | Résistance aux attaques courantes | Moyenne | Bonne | Moyenne | Bonne | |
|  |

**6. Résumé Global**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Critère** | **REST** | **SOAP** | **GraphQL** | **gRPC** |
| **Latence Moyenne (ms)** | 18 | 38 | 26 | 13 |
| **Débit Moyen (req/s)** | 425 | 225 | 325 | 625 |
| **Utilisation CPU Moyenne (%)** | 41 | 55 | 48 | 34 |
| **Utilisation Mémoire Moyenne (MB)** | 175 | 247 | 217 | 154 |
| **Sécurité** | Bonne | Très Bonne | Bonne | Très Bonne |
| **Simplicité d’Implémentation** | Très Facile | Moyenne | Facile | Moyenne |