SOMMAIRE

Objet du document

Ce document permet de modéliser l'architecture qui sera réalisée par les équipes de développement. Au-delà de la modélisation graphique, il s'agit également d'énoncer les principes sur lesquels l'architecture s'appuie, de justifier cette approche architecturale, mais également d'indiquer des transitions si nécessaire.

Objectifs du projet

Les objectifs du projets sont les suivants :

- Développer une nouvelle application déployée à l'international à destination des clients.

- Permettre aux clients de gérer toute la procédure de location de véhicule.

- Permettre aux clients la gestion de son profil et également la gestion des réservations.

- Répondre aux besoins fonctionnels et aux contraintes techniques qui sont liés à la performance et à la maintenabilité de l'application.

- Avoir accès à une API pour consulter et modifier les données traitées par l'application à destination des clients.

Principes d'architecture

L'architecture en couches est une architecture classique constituée de plusieurs couches qui forment l'application. Les principes de cette architecture sont les suivantes :

- Chaque couche a des responsabilités distinctes au sein de l'application. Lorsqu'une couche est modifiée, cela n'affecte pas les autres car elles sont indépendantes les unes des autres.

- Elle dispose à minima de 3 couches :
    - une couche interne qui effectue tout le traitement,
    - une couche externe qui communique avec les utilisateurs,
    - une couche qui interagit avec une base de données.

- Les services sont connectés via les interfaces de programmation dites API.

Architecture existante

Le projet ne s'appuie pas sur un produit existant qui serait repris puis mis à jour. Il n'y a donc pas d'architecture existante à définir.

Architectures

- **Métier**

  L'architecture d'un point de vue métier dite fonctionnelle traite de tout ce que fait l'application : les données manipulées et comment elle s'inscrit dans son écosystème applicatif. De ce fait quatre axes principaux permettent de détailler l'aspect métier : les utilisateurs, les données utilisées, les traitements réalisés et les interfaces avec les autres applications.

  - Les utilisateurs

    La première étape de l'analyse fonctionnelle est d'identifier qui va utiliser le système que l'on est en train de définir, et de l'usage fait du système.

  - Les données manipulées par le système

    Une fois les utilisateurs bien identifiés, on peut commencer à s'intéresser aux données qui seront manipulées. Ces informations vont permettre notamment d'affiner la composante sécurité de l'expression de besoin.

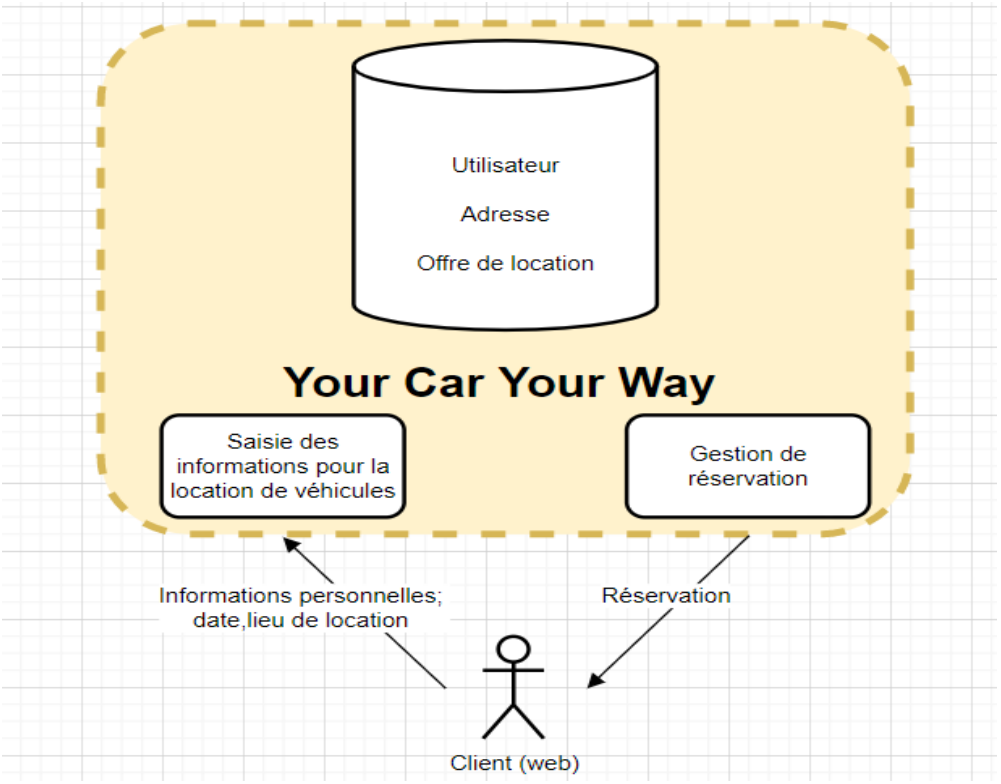
- Les traitements réalisés

  Lister les grands blocs fonctionnels par type d'usage : gestion d'inventaire, authentification des utilisateurs, production de rapports… Idéalement, c'est ici qu'on va associer des notions de performances : telle fonction doit répondre en moins de 3 secondes, tel rapport doit être produit chaque nuit, etc

- Les interfaces

  Dernier pan à analyser, les échanges entre les systèmes doivent être caractérisés par 2 éléments : la donnée portée et surtout le sens de transfert de la donnée.

Couche fonctionnelle de l'application



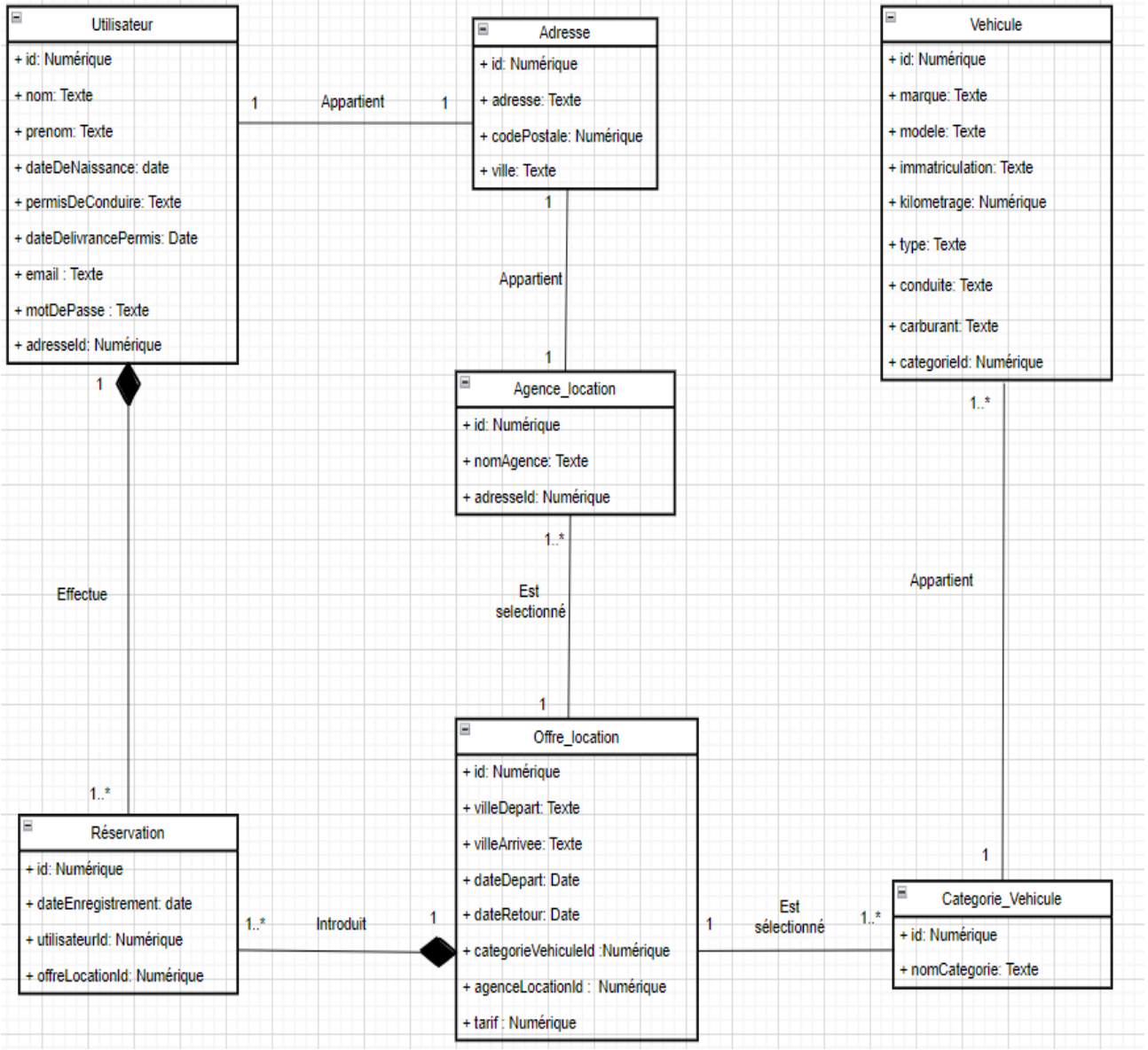
- **De données**

Une architecture d'un point de vue de données décrit comment les données sont gérées, de la collecte à la transformation, la distribution et la consommation. Elle définit le plan directeur des données et la manière dont elles circulent dans les systèmes de stockage de données.

Le modèle représenté est le modèle entité-association sous forme de diagramme, qui reproduit les relations entre les entités du monde réel de façon très similaire au modèle réseau. Il est souvent utilisé pour la création d'une base de données d'un point de vue conceptuel.

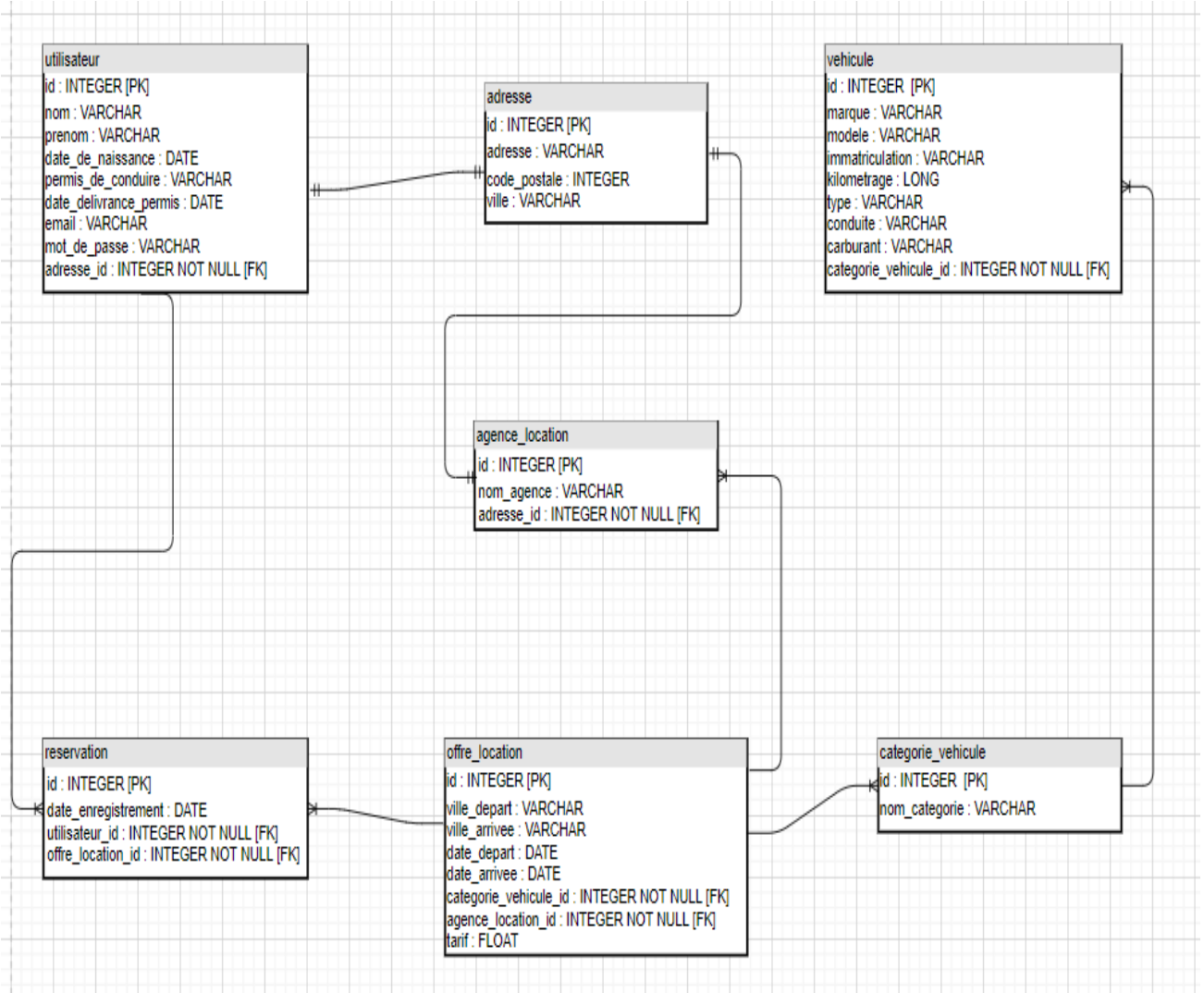
Ici, les personnes, les lieux et les objets à propos desquels les points de données sont stockés sont appelés entités, chacune d'entre elles possédant certains attributs qui, ensemble, composent leur domaine. On schématise aussi les relations entre les entités.

Diagramme de classe de l'application



A partir de ce diagramme, il est possible de modéliser son schéma de bases de données relationnelles..

Schéma de base de données de l'application



Dans ce schéma, on peut retrouver les types de données manipulées, les associations entre les tables (un à plusieurs, un à un), et également les liens entre les grâce aux clés étrangères définies par FK signifiant FOREIGN KEY. Une clé étrangère, c'est un attribut (ou groupe d'attributs) d'une table qui fait référence à la clé primaire d'une autre table, afin de modéliser le lien entre les lignes de ces deux tables.

- **Technologiques**

  L'architecture d'un point de vue technologique correspond à la description des composants logiciels, de leurs interactions, de leurs infrastructures. Cela inclut la définition des technologies mobilisées pour implémenter l'architecture.

Diagramme de composant de l'application

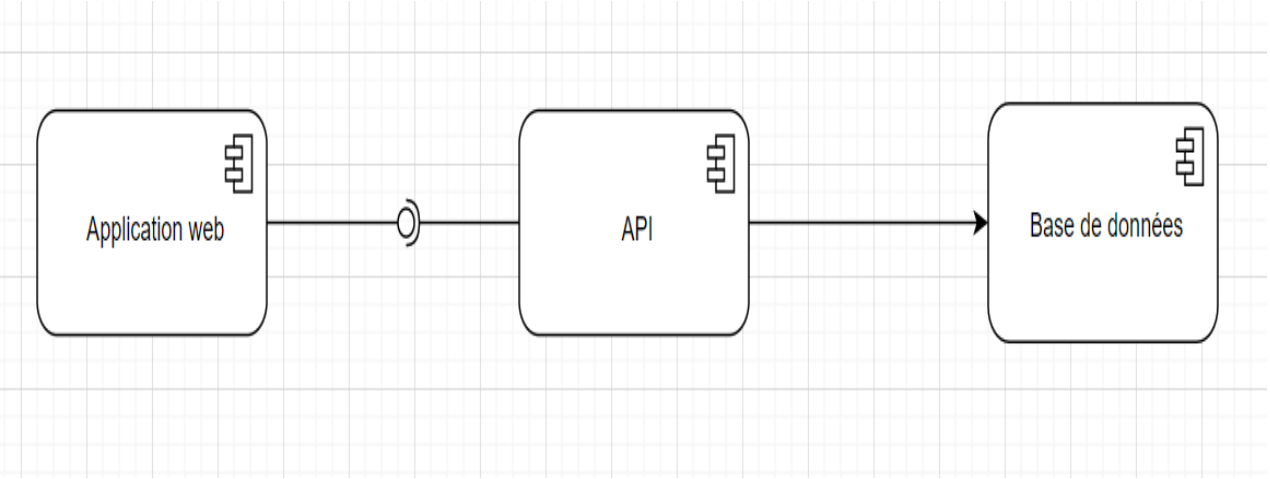
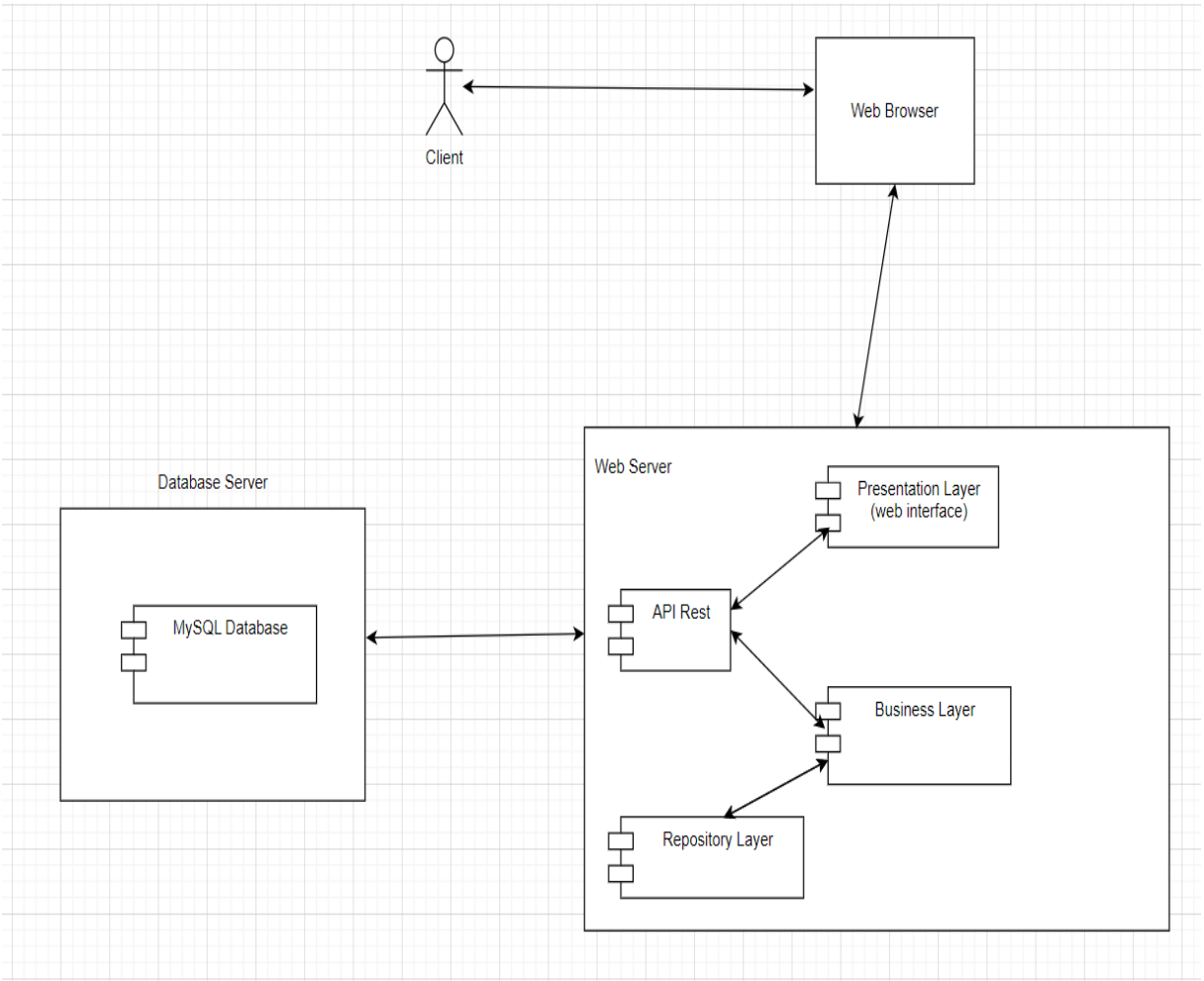


Diagramme de déploiement de l'application



Voici les technologies mobilisées pour implémenter l'architecture :

a. Back-End

- Java 17 : langage de programmation pour le développement de l'application côté serveur.

- Spring Boot : outil qui accélère et simplifie le développement d'applications. C'est un composant Spring au service des autres composants.

- Spring Data Rest : pour accélérer et simplifier l'exposition d'une ressource à l'aide d'une API REST.

- Spring Data JPA : c'est un module de Spring qui traite l'accès aux données par une couche de persistance.

- Spring Web : pour créer des applications Web basées sur Spring avec une configuration minimale, telle que le serveur Web Apache Tomcat.

- Spring Security : c'est un module de Spring qui permet d'ajouter automatiquement des fonctions d'authentification et de contrôle d'accès.

- WebSocket : permet d'ouvrir un canal de communication bidirectionnelle entre un navigateur et un serveur.

- JWT : permettant de sécuriser davantage l'application via des jetons qui sont générés et vérifiés rapidement et efficacement du côté serveur.

- Spring MVC Validator : pour restreindre l'entrée fournie par l'utilisateur, en utilisant l'API Bean Validation.

- Lombok : librairie Java qui se connecte automatiquement à l'IDE en fournissant un ensemble d'annotations utiles.

- MySQL : système de gestion de base de données recueillant les requêtes SQL.

b. Front-End

- HTML5 / CSS3 : langage de balisage pour le développement de l'application côté client.

- Angular 16 : framework JavaScript permettant de créer des applications web et mobiles. Il est basé sur TypeScript qui a pour but d'améliorer et de sécuriser le code JavaScript.

- Angular Material : librairie créée par l'équipe Angular, permettant de créer des interfaces simples et élégants, et ce de manière responsive.

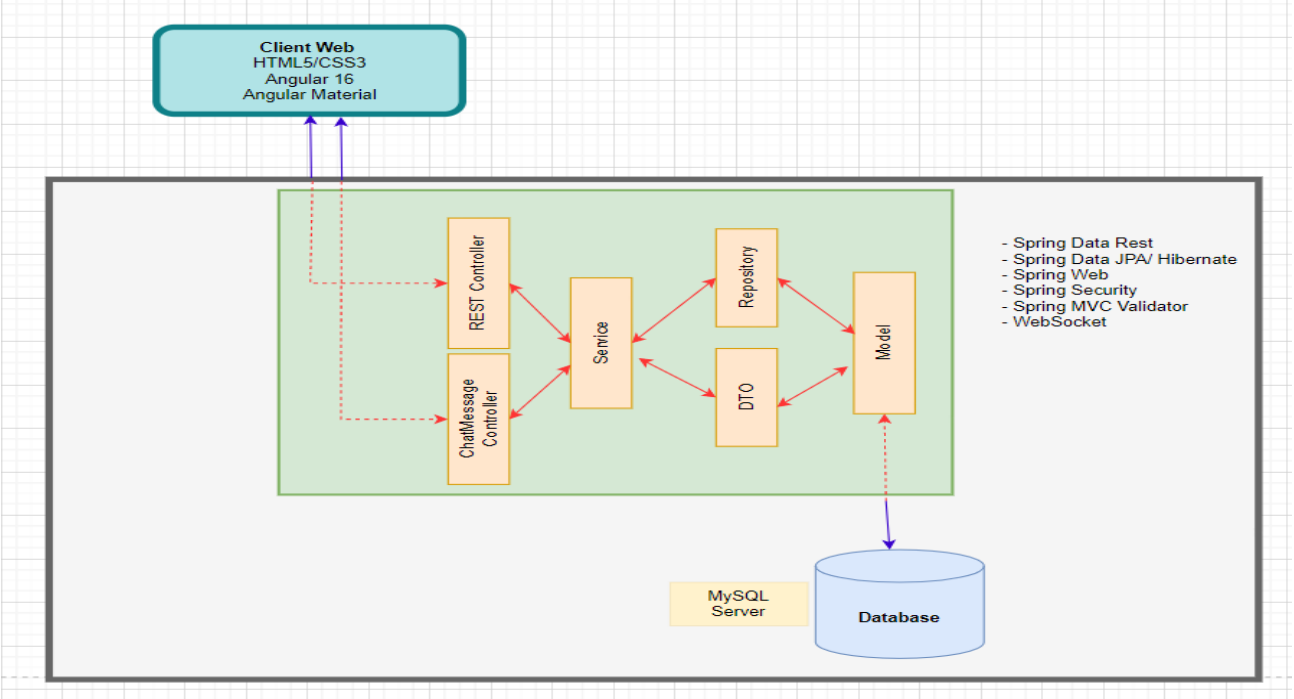
Justification de l'approche architecturale

Cette architecture permet de répondre aux objectifs attendus tout en respectant les principes en suivant les normes et bonnes pratiques.

En effet, elle prend en compte tous les aspects de l'application à destination du client comme le fait de gérer toute la procédure de location de véhicule à savoir la gestion de son profil, la gestion de l'offre de location et également la gestion des réservations; de répondre aux besoins fonctionnels et aux contraintes techniques qui sont liés à la performance et à la maintenabilité de l'application avec la séparation des couches de l'application tout en mobilisant les dernières technologies.

L'architecture est modélisée de la manière suivante :

- Couche Présentation dite Client Web : interface utilisateur qui permet d'afficher des informations et de les collecter.
- Couche REST Controller : gestion des interactions entre l'utilisateur de l'application et l'application avec l'usage des API REST.
- Couche ChatMessage Controller : canal de communication entre le navigateur et le serveur avec l'usage de l'API WebSocket.
- Couche Service : implémentation des traitements métiers.
- Couche DTO : objet léger utilisé pour transférer des données.
- Couche Repository : interaction avec les sources de données externes.
- Couche Model : implémentation des objets métiers.



Architecture de transition

Cette section étant optionnelle, je l'ai rédigé dans le cas où la mise en œuvre complète de toute l'architecture revêt une forte complexité.

De ce fait, ayant choisi une architecture en couche en architecture d'application, mon choix s'est porté sur l'architecture client-serveur dans le cas d'une transition. En effet, cette architecture permet une communication entre le client et le serveur de base de données via le serveur d'application afin qu'une demande soit traitée.

Voici les trois parties qui définissent une architecture client-serveur standard :

- *La partie présentation dite Front-End* qui interagit avec l'utilisateur. Sa principale fonction est d'afficher des informations à l'attention de l'utilisateur et d'en collecter de ce dernier. Elle peut s'exécuter sur un navigateur Web.

- *Le serveur d'application* contient les modules logiciels de l'application. C'est dans cette partie que les informations collectées depuis le navigateur Web sont traitées. Le serveur d'application se connecte à la base de données et interagit avec les utilisateurs afin de répondre aux requêtes effectuées par celui-ci.

- *Le serveur de base de données* porte sur l'accès aux données, c'est l'endroit où les informations traitées par le serveur d'application sont stockées et gérées. C'est ici que l'on peut créer des opérations d'insertion/de suppression/de mise à jour etc.