

**ECOLE DES HAUTES ETUDES D'INGENIERIE – OUJDA**

**Projet de Synthèse 3<sup>ème</sup> Année**

FILIÈRE : Génie Informatique

**Développement d'un Site web pour réaliser le  
problème de location de voiture**

**N°26**

Réalisé par :

<b>Nom et Prénom</b>	<b>Groupe GI</b>
ELJEMLI Houssam-Eddine	E
BOUSSIARI YASSER	D
GUERROUJ WAIL	E
ARBIB Oualid	E

Soutenance le 14/06/ 2024 devant le Jury :

M. ECHCHADLI BELKASSEM

M. MANI MOHAMMED ADIL

M. MOUHIB IMAD

M. SERGHINI ABDELHAFID

**Année universitaire : 2023 - 2024**

# I. Tableau de Matière :

I.	Tableau de Matière : .....	2
II.	Introduction : .....	5
III.	Conception : .....	6
1.	Description du système : .....	6
a.	Problématique : .....	6
b.	La Solution Proposées : .....	7
c.	Etude de marché : .....	8
d.	Contraintes : .....	10
e.	Les concurrents : .....	12
f.	Marketing : .....	13
2.	Partie Merise : .....	15
a.	Qu'est-ce que Merise : .....	15
b.	Les Formes Normales de Merise : .....	17
i.	La première forme normale : .....	17
ii.	La deuxième forme normale : .....	18
iii.	La Troisième forme normale : .....	18
c.	Modèle Conceptuel de données : .....	19
d.	Modèle Logique de Données : .....	20
e.	Qu'est-ce que l'Architecture Client / Serveur : .....	21
f.	Qu'est-ce qu'un SGBD : .....	23
g.	Qu'est-ce que SQL : .....	25
i.	L'histoire de SQL : .....	25
ii.	Comment fonctionne SQL ? .....	25
h.	Script de la base de données : .....	26
3.	Partie UML : .....	28
a.	Qu'est-ce que UML : .....	28
a.	Diagramme de contexte : .....	29
b.	Diagramme de Cas d'utilisation : .....	30
i.	Use Case de s'inscrire : .....	31
ii.	Use Case de s'authentifier : .....	32
iii.	Use Case de Consulter : .....	33
iv.	Use Case de Réservé : .....	34
v.	Use Case de Paiement : .....	35
vi.	Use Case de Rendre la voiture : .....	36
c.	Description Textuelle : .....	37

i.	Description pour Consulter : .....	37
ii.	Description pour Réserver : .....	39
iii.	Description pour Rendre une Voiture : .....	41
d.	Diagramme de séquence : .....	42
i.	Diagramme de séquence (Authentifier) : .....	42
ii.	Diagramme de séquence (Consulter) : .....	43
iii.	Diagramme de séquence (Annuler) : .....	44
e.	Les Concepts de l'Orienté Objet : .....	45
f.	Diagramme de Classe : .....	49
g.	Qu'est-ce que l'Architecture MVC : .....	50
<b>IV.</b>	<b>JAVA :</b> .....	<b>53</b>
1.	Architecture du Système : .....	53
a.	Classe user : .....	53
b.	Classe gérant (hérite d'User) : .....	55
c.	Classe contact : .....	60
d.	Classe Facture .....	63
e.	Claas Client : .....	65
f.	Class Voiture : .....	69
g.	Class main : .....	71
<b>V.</b>	<b>C++ :</b> .....	<b>73</b>
1.	Qu'est-ce que C++ : .....	73
a.	Caractéristiques Principales de C++ : .....	73
b.	Utilisations de C++ : .....	73
c.	Description de la partie du projet à coder avec C++ : .....	74
d.	Description de la partie du projet à coder avec C++ : .....	76
e.	Captures écrans des interfaces : .....	80
f.	Code source des différentes classes : .....	83
<b>VI.</b>	<b>Web :</b> .....	<b>89</b>
1.	Description : .....	89
2.	Diagramme De Navigation : .....	92
3.	Les pages Web : .....	93
a.	Page d'accueil : .....	93
i.	Accueil : .....	93
ii.	Conduite : .....	94
iii.	Services : .....	95
iv.	A propos : .....	96

v.	Equipe :	97
b.	Page de Réservation :	98
c.	Page de Login :	99
d.	Page d'Enregistrer :	100
e.	La Liste des Voitures :	101
f.	Page du Contrat :	102
VII.	Conclusion :	103

## II. Introduction :

Dans notre société moderne, la digitalisation transforme profondément nos modes de vie et nos interactions avec les services. Cette révolution numérique touche tous les secteurs, y compris celui de la location de voitures. En réponse à cette évolution, notre projet se concentre sur le développement d'un site web dédié à la location de voitures, visant à résoudre les divers problèmes rencontrés dans ce domaine.

L'objectif principal de ce projet est de créer une plateforme en ligne qui simplifie et améliore l'expérience de location de voitures pour les utilisateurs. Grâce à des fonctionnalités telles que la réservation en ligne, la gestion des disponibilités en temps réel et le paiement sécurisé, ce site web offrira une solution pratique et accessible à la fois pour les clients et les entreprises de location de voitures. En facilitant l'accès aux services de location et en optimisant la gestion des flottes de véhicules, notre plateforme ambitionne de répondre aux exigences croissantes d'un marché en pleine transformation numérique.

### **III. Conception :**

#### **1. Description du système :**

##### *a. Problématique :*

La problématique principale d'un système de location de voiture réside souvent dans la nécessité de gérer efficacement une flotte de véhicules pour répondre aux besoins de clients variés. Cela implique la coordination de réservation, la gestion des retours, la tarification compétitive, la garantie de la sécurité des conducteurs, et la mise en place d'un système fluide pour une expérience client optimale, la conception d'un système doit également prendre en compte les défis liés à la logistique à la gestion de données et à l'innovation technologique.

Prévoir un système à une agence de locations toute personne locataire provoquant des dégâts néfastes aux automobiles louées. Aussi de signales les personnes qui ne respectent pas les termes de paiements aux agences.

### *b. La Solution Proposées :*

Pour résoudre cette problématique qui se compose de plusieurs problèmes, on a Trouvé des petites solutions pour ces derniers. Pour la gestion de location de Voiture, il est crucial d'aborder les problèmes de sécurité, les accidents, le manque D'expérience de conduite et la conception. Voici quelques solutions pour chaque Domaine :

#### **Sécurité des voitures :**

- 1. Maintenance régulière :** Mettre en place un calendrier strict pour l'entretien des Véhicules afin de garantir leur bon fonctionnement.
- 2. Suivi technologique :** Utiliser des systèmes de suivi GPS pour localiser les Voitures en cas de problème ou de vol.
- 3. Inspections approfondies :** Avant et après chaque location, effectuer des Inspections pour vérifier l'état des véhicules.

#### **Accidents :**

- 1. Assurance robuste :** Offrir une assurance complète pour couvrir les accidents et Informer les locataires sur les procédures à suivre en cas d'accident.
- 2. Formation et sensibilisation :** Fournir des guides de sécurité routière et des vidéos De formation pour sensibiliser les conducteurs à la sécurité.

#### **Manque d'expérience de conduite :**

- 1. Évaluation des conducteurs :** Établir des critères pour évaluer les compétences de Conduite des locataires et offrir une formation en conséquence.
- 2. Options de conduite assistée :** Proposer des voitures équipées de technologies D'assistance à la conduite pour réduire les risques.

#### **Conception :**

- 1. Renouvellement de la flotte :** Introduire progressivement de nouveaux modèles Plus sûrs et technologiquement avancés.
- 2. Personnalisation des options :** Offrir une variété de véhicules pour répondre aux Besoins divers des clients, de la petite voiture citadine aux SUV pour les familles. Pour gérer efficacement ces solutions, un système de gestion de locations de Voitures est la bonne solution pour inclure tous ses derniers dans interface unifiée.

*c. Etude de marché :*

Au cours des dernières années, on a observé un changement significatif vers des modèles de mobilité partagée. Les services de covoiturage et de location de voitures à la demande ont gagné en popularité, offrant une plus grande flexibilité et accessibilité aux services de location. Une tendance croissante est l'intégration de véhicules électriques dans les flottes de location, en réponse aux préoccupations environnementales et aux incitations gouvernementales.

Avec l'avènement d'Internet, le secteur a subi une transformation majeure. Les entreprises ont adopté des modèles de réservation en ligne, étendant ainsi leur portée et offrant une plus grande commodité aux clients. Les plateformes de réservation en ligne et les applications mobiles ont joué un rôle crucial dans la facilitation des réservations des voitures. Les entreprises de location ont également mis en œuvre des technologies de pointe pour améliorer l'expérience utilisateur et optimiser leurs opérations.

Pour répondre aux besoins de tout le choix des clients, les entreprises de location ont élargi leurs options de paiement et de tarification. Des modèles de tarification flexibles, tels que la tarification à l'heure ou au kilomètre.

**Segments de Clientèle :**

**Tourisme** : Le tourisme est l'un des principaux moteurs de la demande de location des voitures, avec les voyageurs nationaux et internationaux recherchant des moyens de transport flexibles pour explorer des destinations touristiques.

**Voyages d'Affaires** : Les déplacements professionnels contribuent significativement à la demande de location des voitures, les entreprises ayant fréquemment besoin des services de location pour leurs employés en déplacement.

**Événements Spéciaux** : La tenue d'événements spéciaux peut entraîner une augmentation de la demande de location des voitures pour répondre aux besoins de déplacement des participants.

**Locaux** : Résidents locaux ayant besoin de services de location pour des déplacements ponctuels ou des occasions spéciales.

### Tendance du Marché :

**Plateformes de Réservation en Ligne** : L'évolution technologique a facilité la location de voitures avec des processus plus simples et rapides grâce aux plateformes de réservation en ligne et aux applications mobiles.

**Véhicules Connectés** : L'intégration de technologies connectées dans les véhicules de location offre des fonctionnalités avancées améliorant l'expérience utilisateur et la sécurité.

**Expérience Client Améliorée** : Les avancées technologiques visent à offrir des options de réservation plus simples, des véhicules connectés, et des processus plus efficaces.

### Tarification Actuelle sur le Marché :

La tarification dans le marché de la location de voitures peut varier en fonction de la région géographique, de la saison, de la durée de location, du type de véhicule, et des politiques spécifiques de chaque entreprise. Les tarifs peuvent être structurés de différentes manières, y compris des tarifs horaires, journaliers, hebdomadaires ou mensuels. Les coûts peuvent inclure des frais supplémentaires pour l'assurance, le carburant, les kilomètres supplémentaires, etc.

### Conseils pour les Consommateurs :

**Gamme de Véhicules** : Comparez les tarifs pour des types de véhicules similaires, en tenant compte de la catégorie, de la taille, du modèle, et des caractéristiques.

**Politiques d'Assurance** : Comparez les coûts liés à l'assurance, certains concurrents incluant des assurances complètes dans leurs tarifs, tandis que d'autres facturent des frais supplémentaires.

**Analyse des Coûts** : Effectuez une analyse approfondie des coûts pour déterminer des prix compétitifs tout en assurant la rentabilité, incluant les coûts liés à l'entretien des véhicules, à l'assurance, au carburant, et aux frais opérationnels.

*d. Contraintes :*

Parmi les contraintes qu'on a trouvé par rapport à cette solution il y a :

- 1. Développement technique** : Créer une plateforme en ligne robuste nécessite des Compétences en développement web, une infrastructure solide, une conception conviviale et des fonctionnalités sécurisées.
- 2. Intégrations** : Intégrer des fonctionnalités comme la réservation en temps réel, les Paiements en ligne sécurisés, la gestion des disponibilités des véhicules, etc., exige souvent l'utilisation de plusieurs API et systèmes externes.
- 3. Sécurité des données** : Protéger les données personnelles et financières des utilisateurs est essentiel pour établir la confiance. Cela implique des mesures de sécurité strictes pour prévenir les cyberattaques et les violations de données.
- 4. Expérience utilisateur** : Concevoir une interface conviviale, intuitive et accessible pour les utilisateurs de tous niveaux de compétence technologique est un défi constant. L'expérience utilisateur joue un rôle crucial dans la satisfaction des clients.
- 5. Gestion des paiements et des réservations** : Garantir la fiabilité des transactions en ligne, la gestion des annulations, la synchronisation des réservations avec la disponibilité réelle des véhicules, etc., sont des points critiques.
- 6. Mobile et compatibilité multiplateforme** : Assurer que le site fonctionne de manière Optimale sur différents navigateurs et appareils mobiles est une nécessité pour atteindre un plus large public.
- 7. Réglementations et conformité** : Respecter les réglementations locales et internationales en matière de locations de voitures, de paiements en ligne, de protection des consommateurs, etc., est essentiel pour éviter les litiges légaux.
- 8. Service client et support** : Fournir un support client efficace pour résoudre les problèmes techniques, répondre aux questions sur les réservations, etc., est indispensable pour maintenir la satisfaction client.
- 9. Évolutivité et maintenance** : Prévoir la croissance future, les mises à jour régulières, la résolution de bogues et la maintenance continue du site sont nécessaires pour rester Compétitif et fonctionnel.

**10. Coûts associés** : Le développement, la maintenance, la sécurité des données et la Promotion du site impliquent des coûts financiers substantiels qui doivent être pris en Compte.

En surmontant ces contraintes, une plateforme de location de voitures en ligne peut offrir une expérience convaincante aux clients et établir une présence forte dans le secteur de la location automobile

e. *Les concurrents :*

## Concurrents :



- **Tarif et Frais** : Nous offrons une clarté totale sur les coûts supplémentaires tels que les frais d'assurance, les frais de carburant et tout autre frais potentiel. Notre objectif est de fournir aux clients une compréhension complète des coûts pour éviter toute surprise désagréable lors de la facturation.
- **Service client** : Notre équipe est disponible pour répondre à toutes vos questions, traiter les préoccupations éventuelles et vous assurer une expérience de location fluide et agréable.
- **Politique de kilométrage** : Notre politique de kilométrage est conçue pour offrir une liberté maximale à nos clients. Nous proposons des options qui s'adaptent à différents besoins, que ce soit pour des trajets courts ou longs. Vous pouvez choisir la meilleure option qui correspond à votre itinéraire et à votre budget.
- **Une promotion à nos clients** : nous offrons régulièrement des promotions exclusives. Ces offres spéciales peuvent inclure des réductions sur les tarifs de location, des mises à niveau de véhicules gratuites ou d'autres avantages.

#### *f. Marketing :*

Le marketing d'un système d'information de location de voiture implique de mettre en avant ses fonctionnalités, sa facilité d'utilisation et ses avantages. Parmi les Méthodes pour faire Marketing de projet :

**1.Définition de Notre Cible** : Identifier notre public cible est une étape cruciale pour personnaliser notre message marketing. Nous cherchons à comprendre les critères démographiques tels que l'âge, le revenu, la géographie, mais également les comportements, tels que les habitudes de voyage, afin d'adapter notre approche de manière pertinente.

**2.Évolution de Notre Produit** : Nous mettons en lumière les fonctionnalités qui rendent notre système de location de voiture unique. De la facilité de réservation à la diversité de notre flotte et aux services supplémentaires, nous cherchons à démarquer notre offre sur le marché.

**3.Stratégie de Communication** : Notre stratégie de communication intègre habilement des canaux en ligne tels que notre site web et les réseaux sociaux, tout en exploitant des canaux hors ligne tels que les publicités et les événements. Nous choisissons des messages qui mettent en avant la valeur ajoutée de notre service pour susciter l'intérêt de notre audience.

**4.Partenariats Stratégiques** : En tant qu'entreprise proactive, nous collaborons avec des partenaires stratégiques tels que des agences de voyage, des hôtels, compagnie aérienne et des entreprises locales. Cette approche élargit notre portée et ouvre des opportunités de co-marketing, renforçant ainsi notre position sur le marché.

**5.Témoignages de Nos Clients** : Nous encourageons activement nos clients satisfaits à partager leurs expériences positives. Ces témoignages authentiques deviennent un pilier de nos campagnes marketing, renforçant la crédibilité de notre service et inspirant la confiance chez nos prospects.

**6.Optimisation de Notre Site Web** : La convivialité de notre site web est une priorité, avec une navigation intuitive et des pages de destination optimisées. L'optimisation pour les moteurs de recherche (SEO) est au cœur de notre stratégie pour garantir une visibilité en ligne maximale et attirer notre public cible de manière efficace.

**7. Utilisation d'Influenceurs :** nous avons décidé de collaborer avec un influenceur renommé. Cette initiative vise à exploiter l'influence et la portée considérables de l'influenceur pour créer une connexion authentique avec notre public cible

## 2. Partie Merise :

### a. Qu'est-ce que Merise :

Merise est une méthode française de modélisation utilisée pour la conception de systèmes d'information. Elle a été développée dans les années 1970 et 1980 pour apporter une approche structurée à l'analyse, la conception et la gestion des systèmes d'information.

Voici quelques points clés à inclure dans un rapport sur Merise :

**Objectif de Merise :** L'objectif principal de Merise est de fournir une méthode systématique pour la création de systèmes d'information qui répondent aux besoins des utilisateurs tout en étant flexibles, évolutifs et maintenables.

**Les trois niveaux d'abstraction :** Merise se caractérise par une approche par niveaux qui divise le processus de conception en trois étapes distinctes :

**Niveau conceptuel :** À ce niveau, les analystes modélisent les concepts fondamentaux du système d'information. Cela inclut des outils comme le modèle conceptuel de données (MCD) qui identifie les entités, leurs relations et leurs propriétés, et le modèle conceptuel des traitements (MCT) qui décrit les flux de traitement.

**Niveau logique :** Ici, les concepts du niveau conceptuel sont traduits en structures logiques qui peuvent être mises en œuvre dans un système informatique. Le modèle logique de données (MLD) et le modèle logique des traitements (MLT) font partie de ce niveau.

**Niveau physique :** Enfin, au niveau physique, le modèle logique est adapté aux contraintes techniques du système réel, comme la structure de base de données, les spécifications matérielles, etc.

**Les principes clés :** Merise repose sur des principes tels que la séparation des préoccupations, la normalisation des données et des traitements, et une approche descendante (top-down). Cela permet de gérer la complexité du système d'information en se concentrant sur différents aspects à chaque niveau.

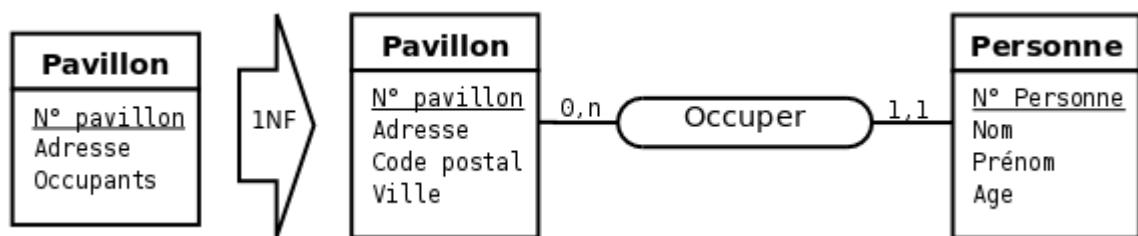
**Applications :** Merise est largement utilisé dans les milieux académiques, les entreprises et les administrations publiques en France et dans d'autres pays francophones. Elle est particulièrement adaptée aux projets de grande envergure où une méthode structurée est nécessaire pour garantir la cohérence et la qualité du système d'information.

Merise offre un cadre complet pour la conception des systèmes d'information, permettant aux analystes et aux développeurs de travailler de manière méthodique tout en restant flexibles pour répondre aux besoins des utilisateurs finaux.

b. Les Formes Normales de Merise :

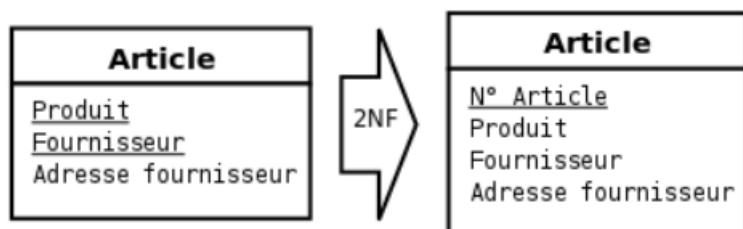
i. La première forme normale :

Une relation est en 1NF (Forme Normale) si elle possède au moins une clé et si tous ses attributs sont atomiques.



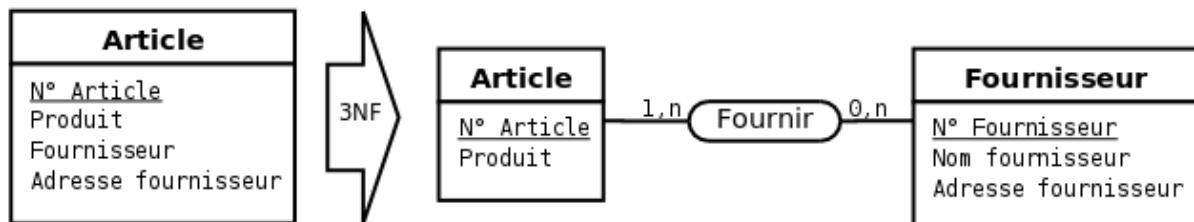
## ii. La deuxième forme normale :

La deuxième forme normale permet d'éliminer les dépendances entre des parties de clé et des attributs n'appartenant pas à une clé.

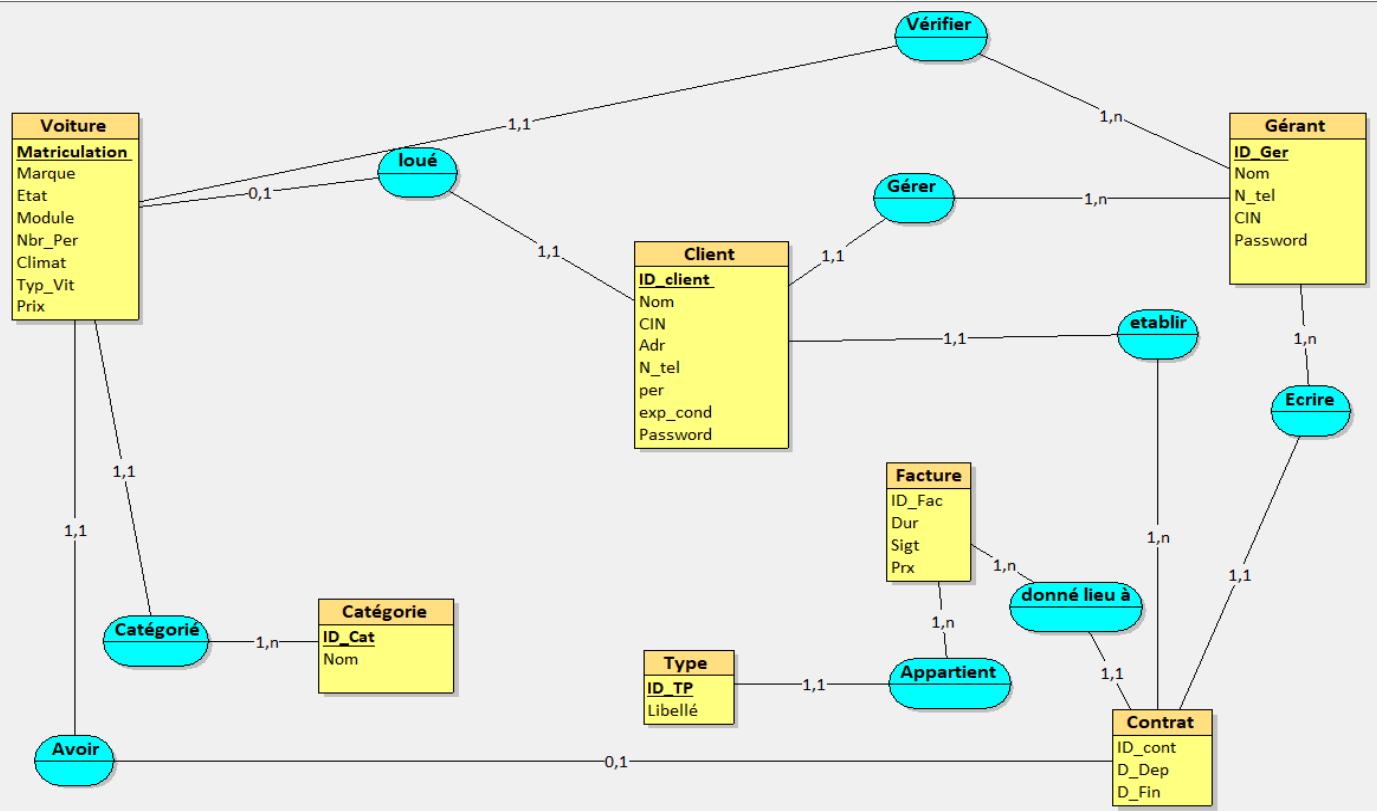


## iii. La Troisième forme normale :

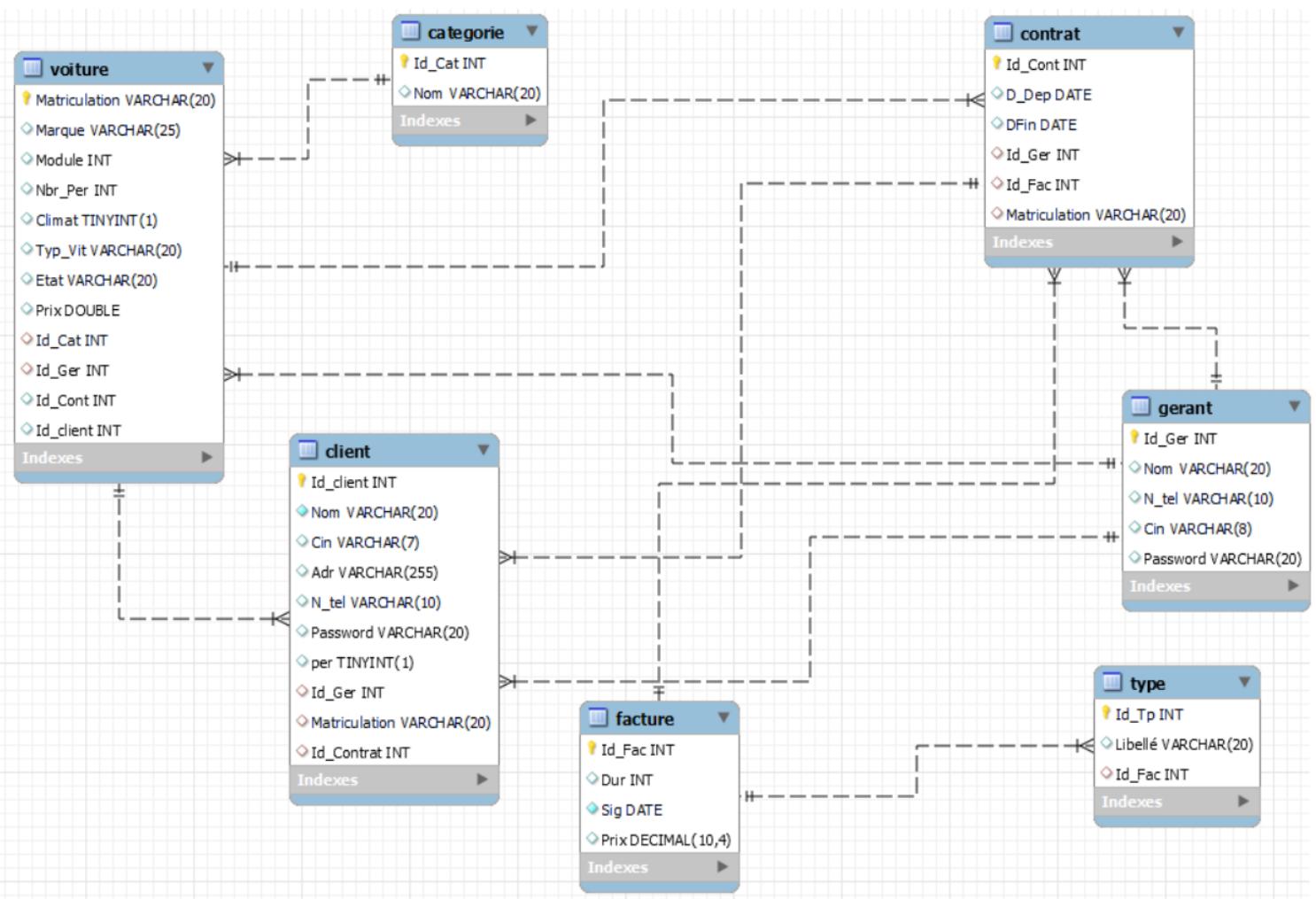
La troisième forme normale permet d'éliminer les dépendances entre les attributs n'appartenant pas à une clé.



c. Modèle Conceptuel de données :



d. Modèle Logique de Données :



*e. Qu'est-ce que l'Architecture Client / Serveur :*

L'architecture client-serveur est un modèle de conception fondamental dans le domaine des technologies de l'information. Dans cette architecture, les responsabilités de traitement de données et de présentation sont séparées entre deux types de programmes ou de machines : les clients et les serveurs. Voici une explication simplifiée de chaque composant :

**Client :**

Le client est l'entité qui sollicite des services ou des ressources auprès du serveur.

Il peut s'agir d'un navigateur web, d'une application mobile, d'un logiciel de bureau, etc.

Le client envoie des requêtes au serveur et traite les réponses reçues pour les afficher à l'utilisateur.

**Serveur :**

Le serveur est un programme ou un ensemble de programmes qui répondent aux demandes des clients.

Il fournit des services ou des ressources demandées par les clients.

Les serveurs peuvent être physiquement distants des clients et sont souvent hébergés dans des centres de données ou des services cloud.

Dans cette architecture :

Le client envoie des requêtes au serveur, généralement via un réseau, tel qu'Internet.

Le serveur reçoit ces requêtes, les traite en fonction des instructions de l'application, puis envoie les réponses appropriées au client.

Les données nécessaires au traitement des requêtes peuvent être stockées localement sur le serveur ou dans des bases de données externes.

Cette architecture est utilisée dans une variété de systèmes informatiques, tels que les applications web, les services web, les jeux en ligne, les systèmes de messagerie, etc. Elle offre une structure flexible qui permet la mise en place de systèmes évolutifs et modulaires.

*f. Qu'est-ce qu'un SGBD :*

Un SGBD, ou Système de Gestion de Base de Données, est un logiciel essentiel dans le domaine de l'informatique et de la gestion de l'information.

Il joue un rôle central dans le stockage, la manipulation et la gestion de vastes quantités de données de manière efficace et sécurisée.

Imaginez un classeur contenant toutes les informations importantes de votre entreprise : les détails des clients, les transactions, les inventaires, etc.

Maintenant, imaginez que ce classeur devient énorme au fur et à mesure que votre entreprise se développe, devenant difficile à gérer et à rechercher rapidement des informations spécifiques.

Un Système de Gestion de Base de Données (SGBD) agit comme un assistant organisé.

Il prend ce classeur et le transforme en une bibliothèque numérique, où chaque information est soigneusement classée et indexée pour une récupération rapide.

Les clients deviennent des fiches individuelles, les transactions sont enregistrées dans des dossiers séparés, et tout est accessible à l'aide de simples requêtes.

Si vous avez besoin de trouver un client spécifique ou de suivre une transaction particulière, le SGBD peut rapidement localiser les données pertinentes.

En simplifiant la gestion et l'accès aux informations, le SGBD permet à votre entreprise de fonctionner plus efficacement,

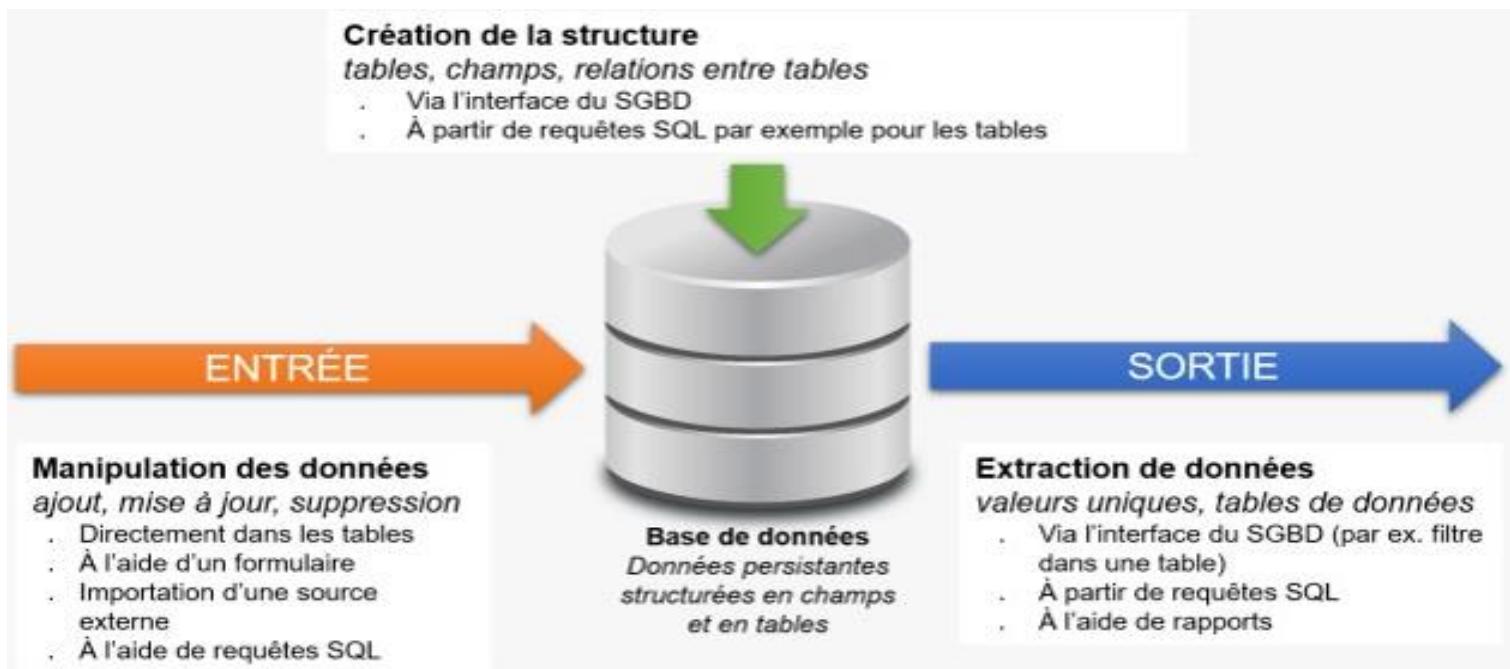
en vous aidant à prendre des décisions éclairées et à répondre aux besoins de vos clients de manière opportune.

En plus de cela, le SGBD assure la sécurité des données en contrôlant l'accès aux informations sensibles,

en garantissant leur intégrité et en permettant des sauvegardes régulières pour prévenir la perte de données.

Certains exemples de SGBD populaires incluent MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server et SQLite.

Chaque SGBD a ses propres caractéristiques, avantages et cas d'utilisation spécifiques.



*g. Qu'est-ce que SQL :*

*i. L'histoire de SQL :*

L'histoire de SQL commence en 1969, lorsque le chercheur d'IBM Edgar F.Codd définit le modèle de base de données relationnelle. Ce modèle repose sur l'association de » clés » avec diverses données. Par exemple, un nom d'utilisateur peut être associé à un vrai nom et à un numéro de téléphone.

Quelques années plus tard, IBM crée un langage pour les systèmes de gestion de bases de données relationnelles en se basant sur les travaux de Codd. Ce langage s'appellera d'abord SEQUEL, pour » Structured English Query Language « . Après plusieurs implémentations et révisions, il s'appellera finalement SQL.

Les tests débutent en 1978, puis IBM commence à développer des produits commerciaux comme SQL/DS en 1981 et DB2 en 1983. D'autres vendeurs suivront, comme Sybase, Ingres, ou Oracle qui lance son premier produit en 1979.

*ii. Comment fonctionne SQL ?*

Les applications peuvent être programmées avec différents langages comme Python, PHP ou Ruby. Plusieurs commandes SQL sont fréquemment utilisées pour travailler avec les bases de données. Par exemple, « CREATE DATABASE » permet de créer une base de données, « CREATE TABLE » permet de créer des tableaux. La commande » SELECT » permet de trouver ou d'extraire des données en provenance d'une base de données. « UPDATE » permet d'ajuster ou d'éditer les données, et « DELETE » permet de supprimer certaines données.

Il ne s'agit là que de quelques exemples de commandes très couramment utilisées, pour vous offrir un aperçu du fonctionnement de SQL. Plus la base de données est complexe, plus l'utilisateur devra utiliser de commandes.

Ces commandes permettent d'écrire des « requêtes » pour manipuler les données dans les bases de données. Le système interprète et traite ces commandes, par exemple pour créer un nouvel enregistrement dans une base de données.

*h. Script de la base de données :*

```
1 •  create database DBV;
2 •  USE DBV;
3 •  CREATE TABLE Categorie (
4      Id_Cat INT PRIMARY KEY auto_increment,
5      Nom VARCHAR(20)
6 );
7 •  CREATE TABLE Gerant (
8      Id_Ger INT PRIMARY KEY auto_increment,
9      Nom VARCHAR(20) UNIQUE,
10     N_tel VARCHAR(10),
11     Cin VARCHAR(8),
12     Password varchar(20)
13 );
14 •  CREATE TABLE Voiture (
15     Matriculation VARCHAR(20) PRIMARY KEY ,
16     Marque VARCHAR(25),
17     Module Int,
18     Nbr_Per Int,
19     Climat bool,
20     Typ_Vit varchar(20),
21     Etat VARCHAR(20),
22     Prix double,
23     Id_Cat INT,
24     Id_Ger INT,
25     Id_Cont INT,
26     Id_client int,
27     FOREIGN KEY (Id_Cat) REFERENCES Categorie(Id_Cat),
28     FOREIGN KEY (Id_Ger) REFERENCES Gerant(Id_Ger)
```

```

29      );
30 • Ⓜ CREATE TABLE Facture (
31     Id_Fac INT PRIMARY KEY auto_increment,
32     Dur INT,
33     Sig DATE NOT NULL,
34     Prix DECIMAL(10,4)
35 );
36 • Ⓜ CREATE TABLE Contrat (
37     Id_Cont INT PRIMARY KEY auto_increment,
38     D_Dep DATE,
39     DFin DATE,
40     Id_Ger INT,
41     Id_Fac int,
42     Matriculation Varchar(20),
43     FOREIGN KEY (Id_Ger) REFERENCES Gerant(Id_Ger),
44     FOREIGN KEY (Id_Fac) REFERENCES Facture(Id_Fac),
45     FOREIGN KEY (Matriculation) REFERENCES Voiture(Matriculation)
46 );
47 • Ⓜ CREATE TABLE Client (
48     Id_client INT PRIMARY KEY auto_increment,
49     Nom VARCHAR(20) NOT NULL,
50     Cin VARCHAR(7),
51     Adr VARCHAR(255),
52     N_tel VARCHAR(10),
53     Password varchar(20),
54     per BOOLEAN,
55     Id_Ger INT,
56     Matriculation VARCHAR(20),|
57     Id_Contrat INT,
58     FOREIGN KEY (Id_Ger) REFERENCES Gerant(Id_Ger),
59     FOREIGN KEY (Matriculation) REFERENCES Voiture(Matriculation),
60     FOREIGN KEY (Id_Contrat) REFERENCES Contrat(Id_Cont)
61 );
62 • Ⓜ CREATE TABLE Type (
63     Id_Tp INT PRIMARY KEY,
64     Libellé VARCHAR(20),
65     Id_Fac INT,
66     FOREIGN KEY (Id_Fac) REFERENCES Facture(Id_Fac)
67 );
68

```

### 3. Partie UML :

#### a. Qu'est-ce que UML :

**Unified Modeling Language (UML)** :est un langage de modélisation standardisé utilisé dans le génie logiciel pour spécifier, visualiser, construire et documenter les artefacts des systèmes logiciels. Il offre un ensemble de diagrammes graphiques permettant de représenter divers aspects du système, tels que sa structure, ses comportements et ses interactions. Grâce à UML, les équipes de développement peuvent mieux communiquer et comprendre les exigences et le design du système, ce qui conduit à une meilleure conception et à une documentation plus complète."

#### ➤ Principaux diagrammes UML :

- **Diagrammes structurels :**

Diagramme de classes : Représente les classes du système et leurs relations.

Diagramme d'objets : Montre des instances de classes à un moment donné.

Diagramme de composants : Illustre les composants du système et leurs interdépendances.

Diagramme de déploiement : Montre la configuration physique du matériel et des logiciels dans le système.

- **Diagrammes comportementaux :**

Diagramme de cas d'utilisation (use case) : Représente les interactions entre les acteurs externes et le système.

Diagramme de séquence : Montre l'interaction entre les objets selon une séquence temporelle.

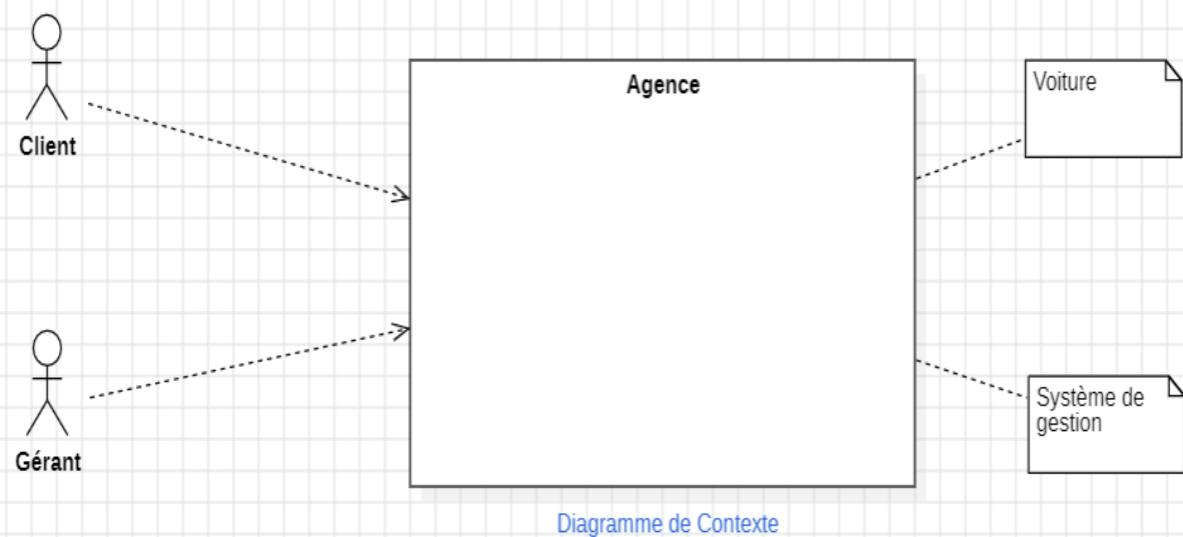
Diagramme de collaboration : Représente les interactions entre objets et leurs relations.

Diagramme d'état (statechart) : Montre les états possibles d'un objet et les transitions entre ces états.

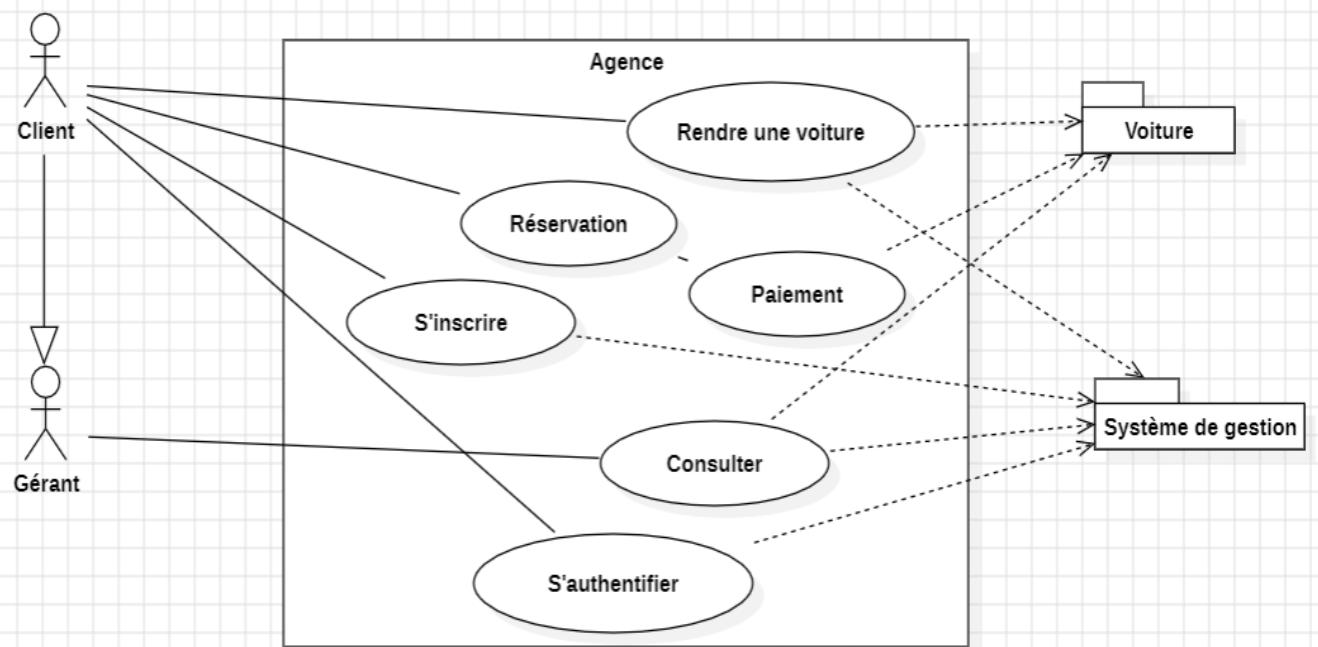
Diagramme d'activités : Décrit les flux de contrôle ou de données entre les activités.

UML est ainsi un outil indispensable pour la modélisation et la gestion de la complexité des systèmes logiciels, facilitant une meilleure conception, une documentation claire et une communication efficace au sein des équipes de développement.

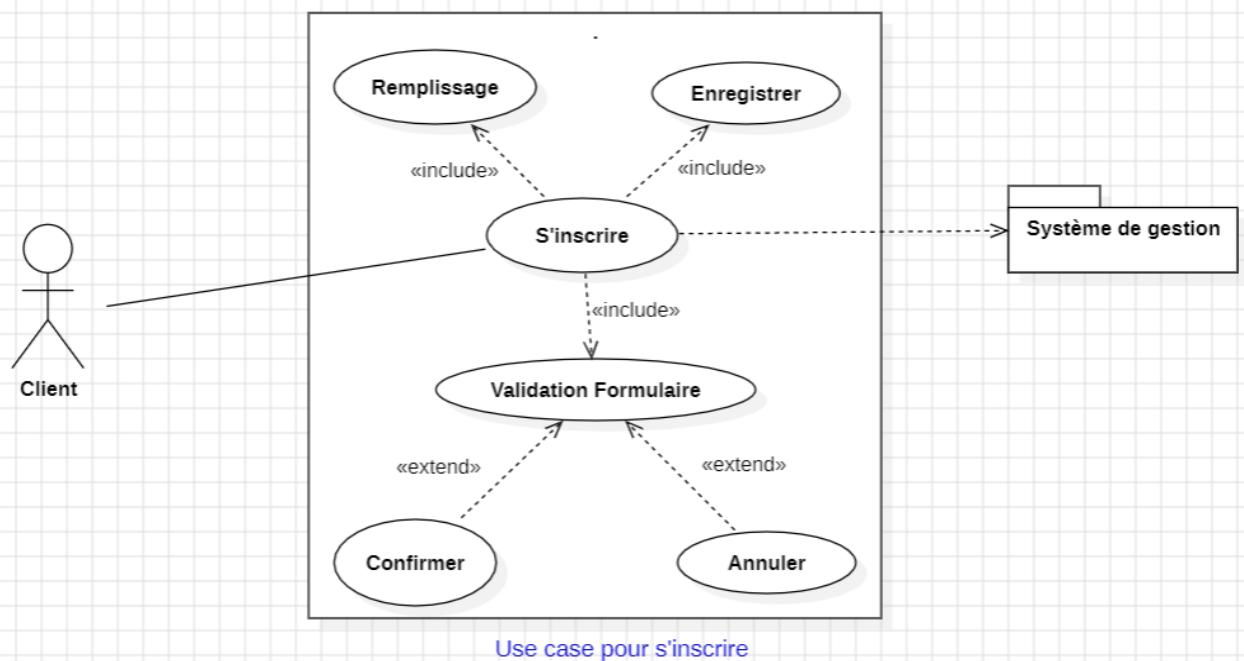
a. Diagramme de contexte :



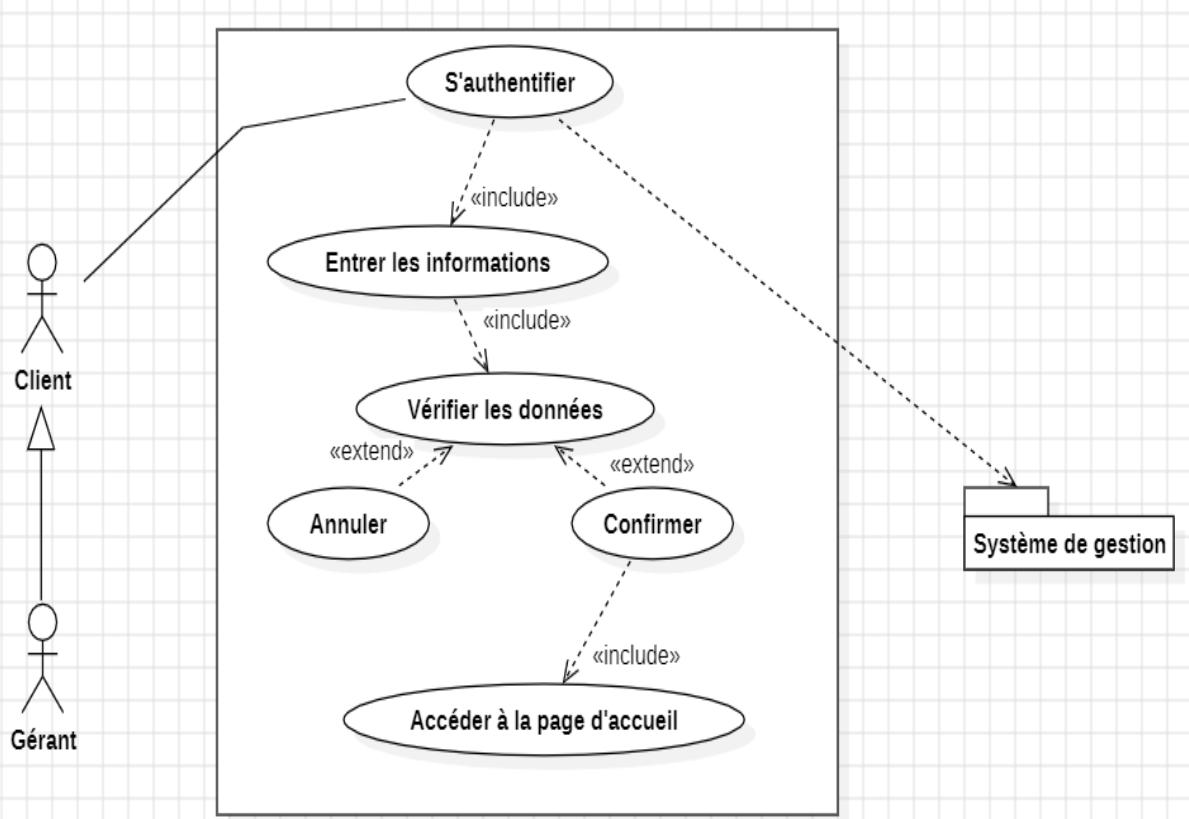
b. Diagramme de Cas d'utilisation :



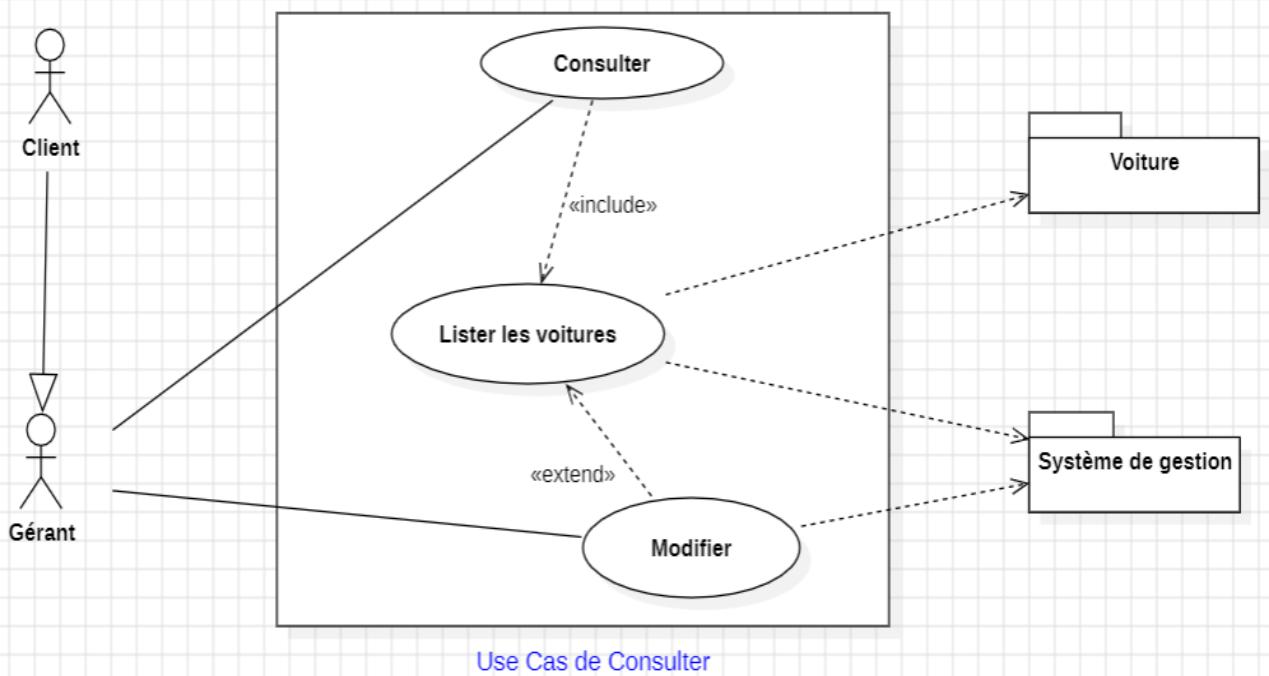
i. Use Case de s'inscrire :



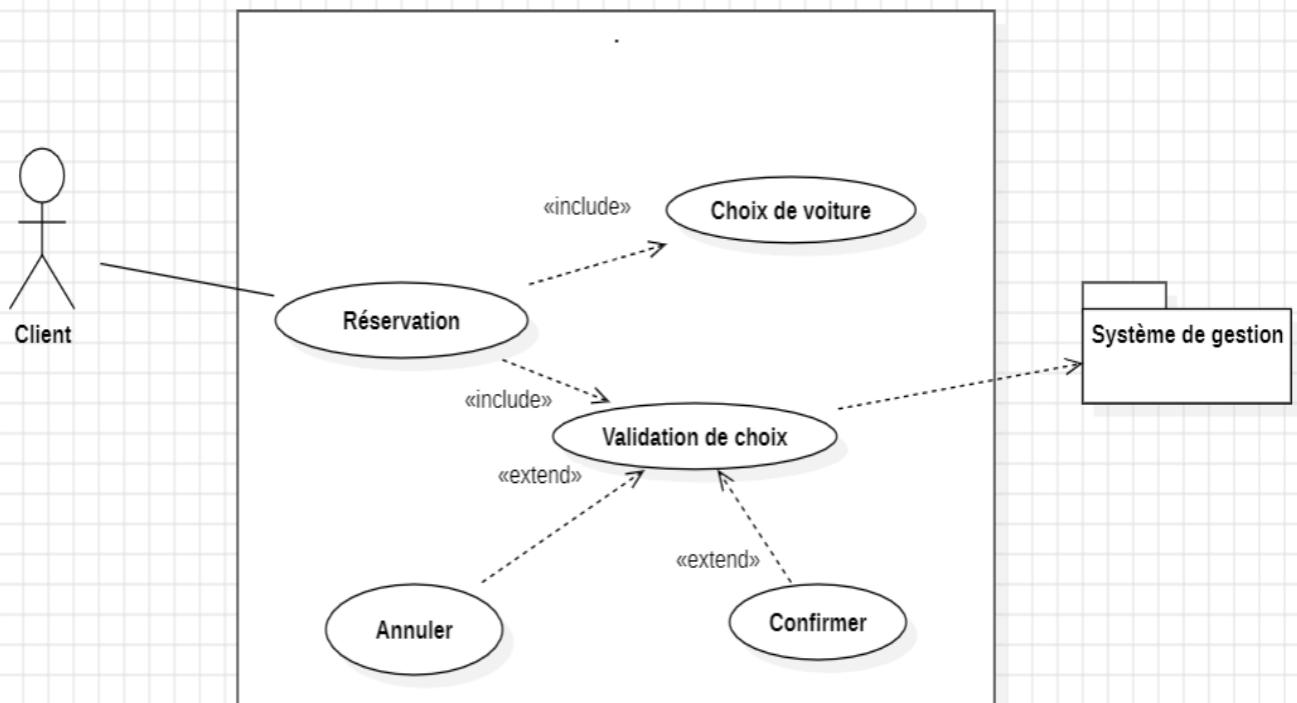
ii. Use Case de s'authentifier :



iii. Use Case de Consulter :

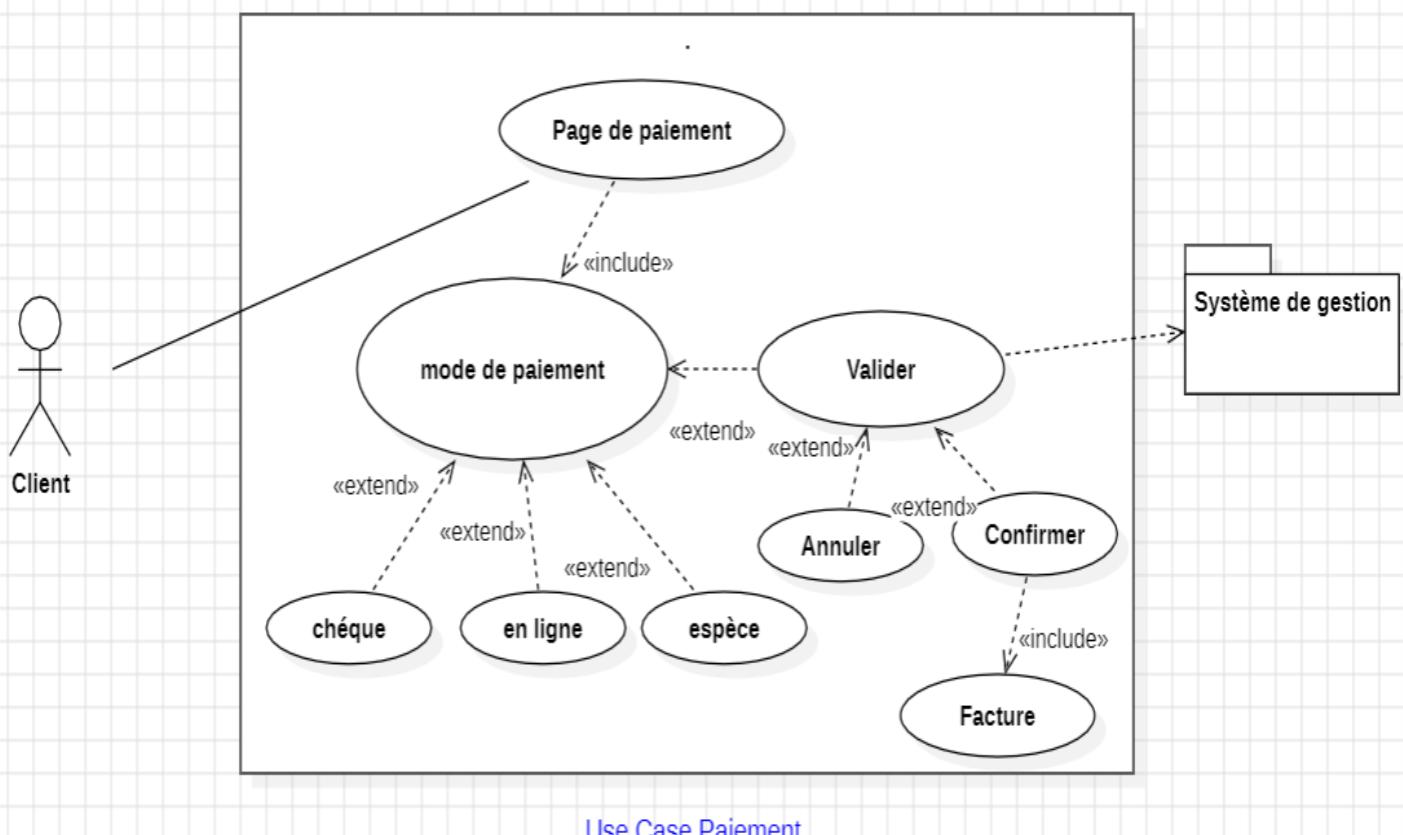


iv. Use Case de Réservé :

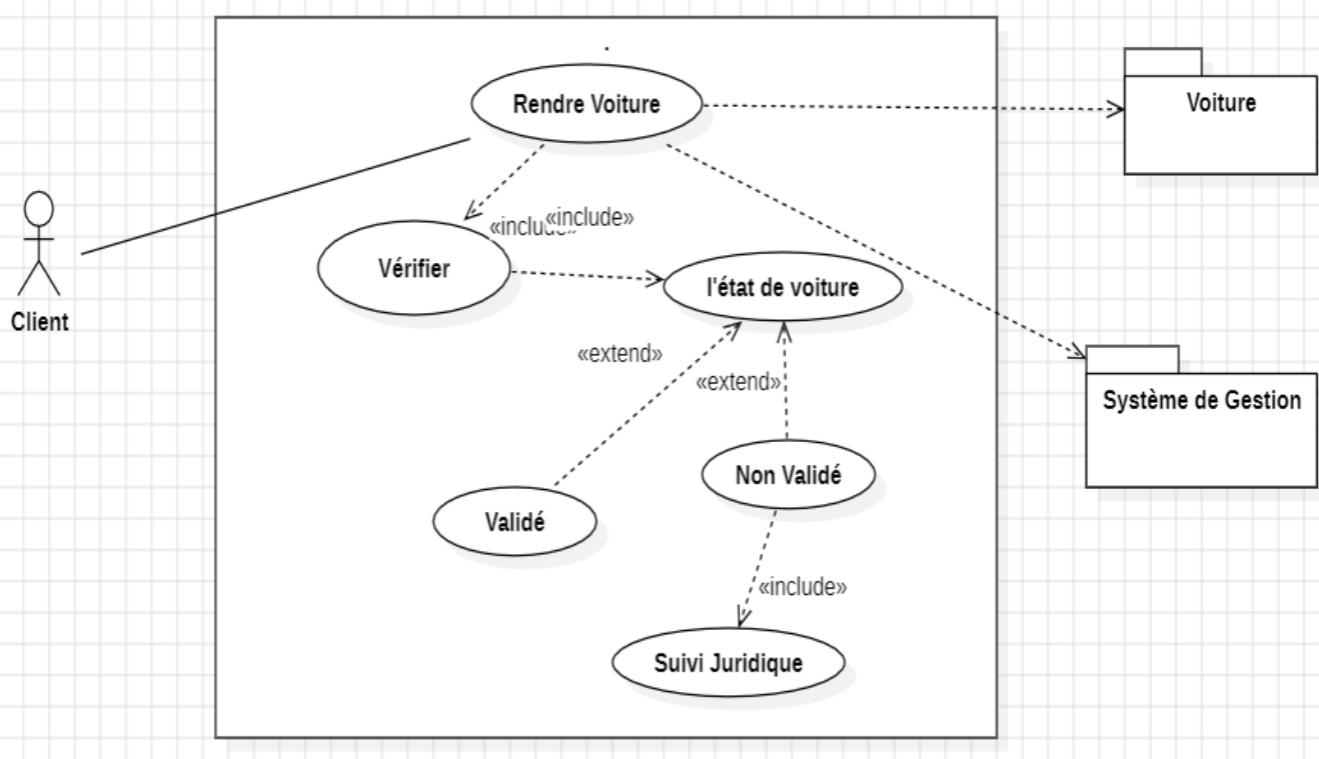


Use Case pour Réservation

v. Use Case de Paiement :



vi. Use Case de Rendre la voiture :



c. *Description Textuelle* :

i. Description pour Consulter :

**Titre** : Consultation de location de voiture

**Résumé** : Ce cas d'utilisation permet au client d'accéder à la partie de consultation dans le site et faire un choix.

**Acteur principal** : Client.

**Acteur secondaire** : Système de consultation

**Préconditions** :

L'utilisateur est connecté au système.

Des locations de voiture sont disponibles dans le système.

**Description textuelle consultation :**

**Scénario nominale** :

1. Le client accède aux consultations des locations de voitures.
2. Le système affiche la liste des voitures disponibles avec des détails tels que le modèle, le prix...
3. Le client explore les différentes offres et examine les détails de chaque voiture et peut s'inspirer des avis des autres clients.
4. Le client analyse et reclasse les offres en fonction de ses choix (date, type de voiture ...).

**Scénario alternatif** :

1. Le client accède aux consultations des locations de voitures.
2. Le système affiche la liste des voitures disponibles avec des détails tels que le modèle, le prix ...
3. Le client veut modifier le choix précédent à cause de présence d'autre choix voulu.
4. Le système enregistre la nouvelle consultation du client.
5. Le système affiche le message de confirmation de consultation au client.

**Scénario échec :**

1. Le client accède aux consultations des locations de voitures.
2. Le client n'a pas accepté la catégorie de voiture, ou/et période de location, ou problème éventuel du blocage de système.
3. Le client quitte brusquement le site.

**Postconditions :**

L'utilisateur a consulté les détails des locations de voiture.

## ii. Description pour Réserver :

**Titre :** réservation de voiture

**Résumé :** Ce cas d'utilisation permet au client de réserver une voiture à partir de la fonction réservée dans le site.

**Acteur principal :** Client

**Acteur secondaire :** Système de gestion.

**Description des scénarios**

**Préconditions :**

L'utilisateur est connecté au système.

Il a consulté les détails d'une location de voiture.

**Scénario nominale :**

1. Le client accède à la gestion de réservations du système.
2. Le client remplit le formulaire.
3. Le système demande au client de s'authentifier.
4. Après la confirmation de l'authentification, le client possède un compte.
5. Le client confirme le choix consulté dans la liste des voitures disponibles et définit l'heure, le début de location et sa durée prévue.
6. Le système confirme la disponibilité de la voiture pour la période spécifiée.
7. Le client peut à ce moment récupérer la voiture à la date et heure spécifiée.

**Scénario alternatif :**

1. À partir de l'étape 2 du scénario nominal :

Le client remplit les cases du formulaire en succession. Il remplit on Émail.

2. S'il a fait une erreur ou il n'a pas rempli la case EMAIL, le système lui demande de la remplir de nouveau
3. Si non, le système valide la case et passe à la case suivante NOM.
4. Il remplit son Nom, s'il a fait une erreur et/ou il n'a pas rempli la case NOM, le système lui demande de la remplir de nouveau
5. Si non, le système valide la case et passe à la case suivante MOT DE PASSE.
6. Il remplit son MOT DE PASSE, s'il a fait une erreur et/ou il n'a pas rempli la case MOT DE PASSE, le système lui demande de la remplir de nouveau

7. Si non, le système valide la case et passe à la case suivante REINSERER MOT DE PASSE.
8. Il réinsère de nouveau son MOT DE PASSE, s'il a fait une erreur et/ou il n'a pas rempli la case MOT DE PASSE, le système lui demande de la remplir de nouveau
9. Si non, le système valide la case et passe à la case suivante CIN.
10. Il remplit de son CIN, s'il a fait une erreur et/ou il n'a pas rempli la case CIN, le système lui demande de la remplir de nouveau
11. Si non, le système valide la case et passe à la case suivante ADRESSE.
12. Il remplit de son ADRESSE, s'il a fait une erreur et/ou il n'a pas rempli la case ADRESSE, le système lui demande de la remplir de nouveau
13. Si non, le système valide la case.
14. Le système demande la validation finale et enregistre le formulaire.
15. Pour passer au processus de réservation il faut s'authentifier en insérant l'email et le mot de passe, s'il a fait une erreur et/ou il n'a pas rempli la case EMAIL, le système lui demande de la remplir de nouveau
16. Si non, le système valide la case. Et passe à la case suivante MOT DE PASSE.
17. Il remplit de son MOT DE PASSE, s'il a fait une erreur et/ou il n'a pas rempli la case MOT DE PASSE, le système lui demande de la remplir de nouveau
18. Si non, le système s'authentifie.
19. Maintenant, le client est connecté et il peut remplir sa réservation.
20. Le client peut changer le choix à cause de la modification au niveau de la consultation.
21. Le client confirme la nouvelle sélection pour terminer le processus de réservation.
22. Le système affiche le message de confirmation de réservation au client.

#### Scénario échec :

1. Le client accède aux gestions des réservations du système.
2. Le client veut annuler la réservation à cause de changement de choix ou site bloqué.
3. Le client réinitialise la réservation ou quitte le site.

#### Postconditions :

-Une réservation est créée pour l'utilisateur avec les détails spécifiés.

-La disponibilité de la voiture est mise à jour dans le système.

### iii. Description pour Rendre une Voiture :

**Titre :** rendre une voiture de location

**Résumé :** Ce cas d'utilisation permet au client de rendre une voiture.

**Acteur principal :** client.

**Acteur principal :** gérant.

**Acteur secondaire :** système de rendement une voiture.

**Préconditions :**

- l'utilisateur utilise la voiture louée.
- l'utilisateur se prépare pour rendre la voiture.

**Description textuelle consultation :**

**Scenario nominal :**

1. Le client accédé à la procédure de la restitution de la voiture de location.
2. Le client nettoie la voiture de location.
3. Le client doit rendre le gasoil à son état de départ.
4. Le client doit rendre la voiture comme elle était dans l'état initial.

**Scénario alternatif :**

1. Le client accédé à la procédure de la restitution de la voiture de location.
2. Le gérant vérifie l'état de la voiture (la propreté, l'état des pneus, les ratures ...)
3. Le gérant vérifie le contenu du contrat s'il est respecté par le client (la date et l'heure du rendement)
4. Si le gérant trouve la voiture en bonne état il valide le rendement de cette voiture

**Scénario échec :**

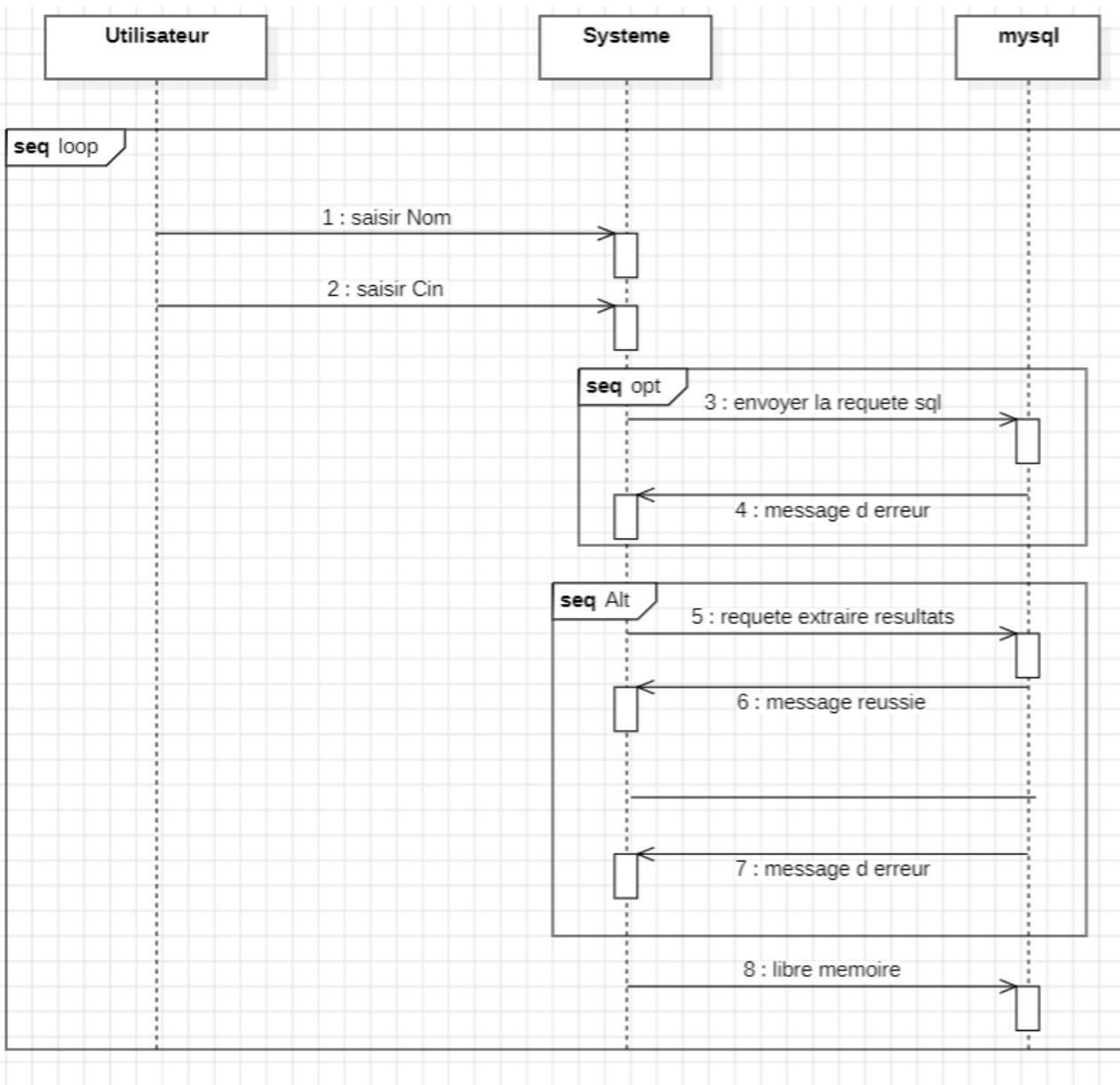
1. Le client accédé à la procédure de la restitution de la voiture de location.
2. Le gérant, après la vérification de la voiture n'a pas validé le rendement de la voiture car le client n'a pas respecté les conditions de contrat.
3. Le gérant oblige le client à payer les dégâts matériels sinon il va être suivi juridiquement Par le tribunal.

**Postconditions :**

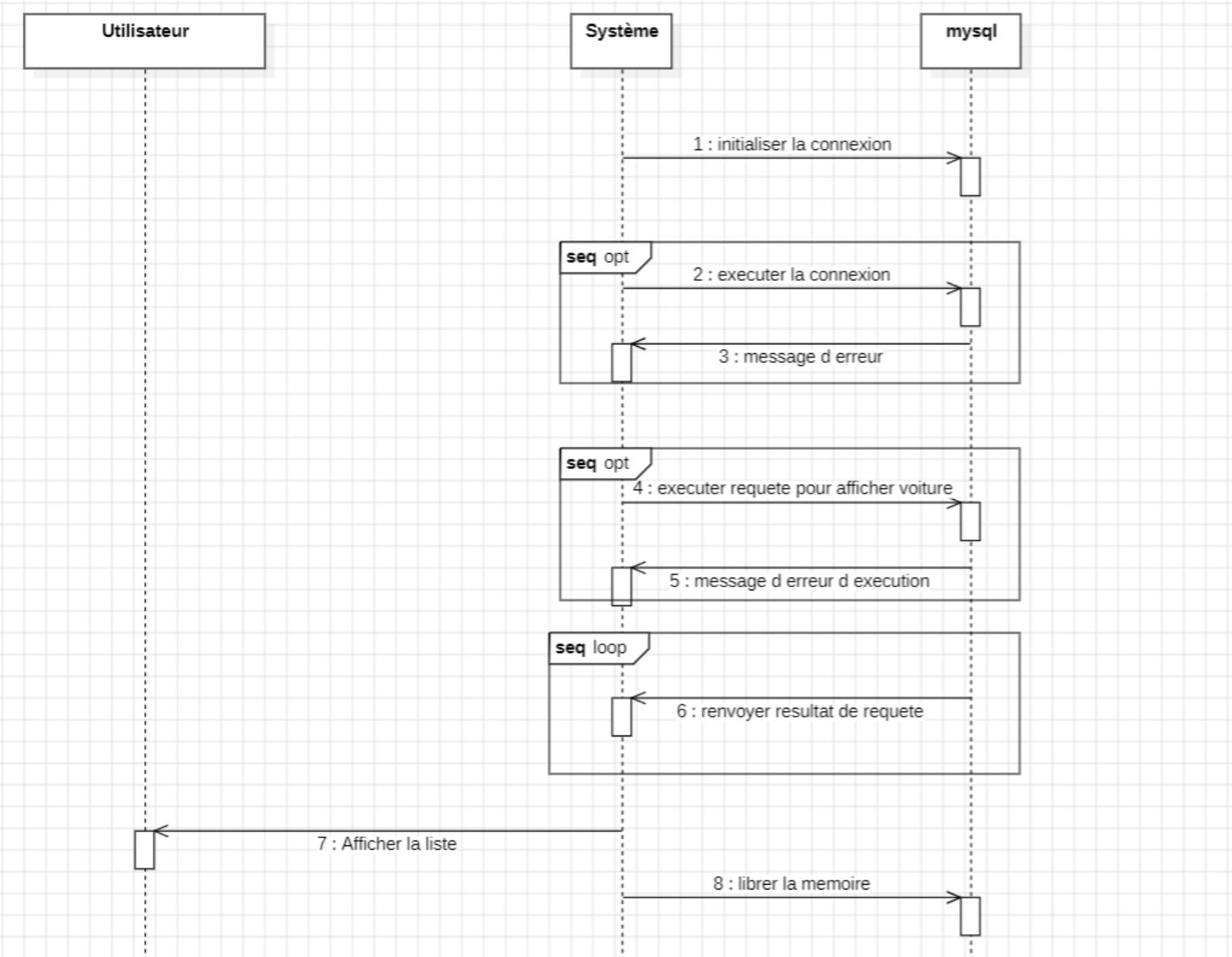
- le gérant vérifie les détails des voitures de location après le rendement.

d. Diagramme de séquence :

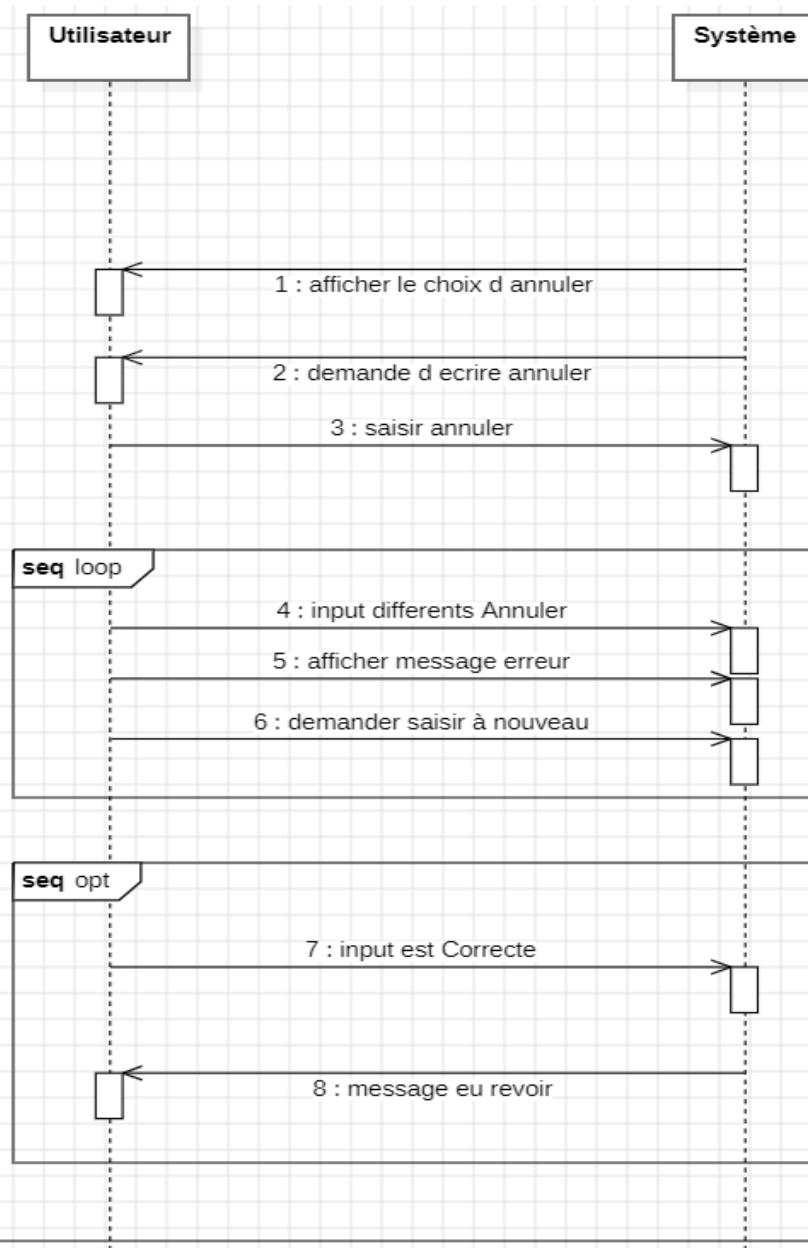
i. Diagramme de séquence (Authentifier) :



ii. Diagramme de séquence (Consulter) :



iii. Diagramme de séquence (Annuler) :



### *e. Les Concepts de l'Orienté Objet :*

Les concepts de la programmation orientée objet (POO) en Java sont fondamentaux pour comprendre comment développer des applications Java robustes et extensibles.

#### ➤ **Voici Quelques-uns des principaux concepts de la POO en Java :**

##### • **Classes et Objets :**

Une classe est un modèle ou un plan à partir duquel des objets sont créés.

Un objet est une instance d'une classe qui possède des états (attributs) et des comportements (méthodes).

##### • **Encapsulation :**

Encapsulation signifie regrouper les données (attributs) et les méthodes qui agissent sur ces données au sein d'une classe.

Les attributs d'une classe sont généralement déclarés comme privés et des méthodes publiques (getters et setters) sont utilisées pour y accéder ou les modifier, assurant ainsi la sécurité des données.

##### • **Héritage :**

L'héritage permet à une classe (appelée classe dérivée ou sous-classe) d'hériter des attributs et des méthodes d'une autre classe (appelée classe de base ou superclasse).

Il favorise la réutilisation du code et permet de créer une hiérarchie de classes.

##### • **Polymorphisme :**

Le polymorphisme signifie la capacité d'un objet à prendre différentes formes.

En Java, le polymorphisme peut être réalisé à travers le surchargement de méthodes (méthodes avec le même nom mais différents paramètres) et le polymorphisme d'exécution (via l'héritage et les méthodes redéfinies).

##### • **Abstraction :**

L'abstraction consiste à fournir une interface simple et générique pour un ensemble complexe de comportements ou de données.

En Java, l'abstraction est réalisée à l'aide de classes abstraites et d'interfaces. Une classe abstraite peut contenir des méthodes abstraites (non implémentées) qui doivent être redéfinies par les sous-classes. Une interface définit un contrat pour les classes qui l'implémentent.

## ➤ **Les concepts de l'orienté objet en c++ :**

En C++, les concepts de la programmation orientée objet (POO) sont similaires à ceux en Java, mais il y a quelques différences importantes.

### **Classes et Objets :**

Une classe est un modèle à partir duquel des objets sont créés. Elle définit la structure et le comportement des objets.

Un objet est une instance d'une classe qui possède des attributs (variables membres) et des méthodes (fonctions membres).

### **Encapsulation :**

C++ supporte l'encapsulation en permettant de définir des membres de classe comme publics, privés ou protégés.

Les membres privés sont accessibles uniquement à l'intérieur de la classe elle-même, tandis que les membres publics peuvent être accessibles depuis l'extérieur de la classe.

### **Héritage :**

L'héritage permet à une classe (appelée classe dérivée ou sous-classe) d'hériter des membres (attributs et méthodes) d'une autre classe (appelée classe de base ou superclasse).

C++ prend en charge l'héritage simple ainsi que l'héritage multiple.

### **Polymorphisme :**

C++ prend en charge le polymorphisme à la fois statique (surcharge de fonctions) et dynamique (liaison tardive ou polymorphisme d'exécution).

Le polymorphisme d'exécution est souvent implémenté à l'aide de fonctions virtuelles et de la redéfinition de méthodes dans les classes dérivées.

### **Abstraction :**

L'abstraction est prise en charge en C++ à l'aide de classes abstraites et d'interfaces.

Une classe abstraite peut contenir des méthodes virtuelles pures qui doivent être implémentées par les classes dérivées.

Les interfaces sont définies à l'aide de classes purement abstraites qui définissent uniquement les signatures des méthodes sans implémentation.

### **Constructeurs et Destructeurs :**

En C++, les classes peuvent avoir des constructeurs pour initialiser les objets lors de leur création et des destructeurs pour libérer les ressources lorsque les objets sont détruits.

Les constructeurs peuvent être surchargés et peuvent avoir des listes d'initialisation pour initialiser les membres de la classe.

### **Surchage des Opérateurs :**

C++ permet de surcharger les opérateurs pour les types définis par l'utilisateur, ce qui offre une flexibilité supplémentaire lors de la manipulation d'objets.

### **Gestion de la Mémoire :**

En C++, la gestion de la mémoire est souvent effectuée explicitement par le programmeur à l'aide d'opérateurs **new** et **delete** pour allouer et libérer de la mémoire dynamiquement.

### **Types de Relations :**

Les relations entre les classes en C++ peuvent être des associations, des agrégations ou des compositions, tout comme en Java. Cependant, C++ offre plus de contrôle sur la gestion des ressources et la visibilité des membres.

## ➤ **Voici quelques-unes des principales différences entre Java et C++ :**

### **Paradigmes de Programmation :**

Java est principalement orienté objet, ce qui signifie que tout est un objet et que le code est organisé en classes et objets. Il supporte également la programmation impérative et fonctionnelle.

C++ est un langage polyvalent qui prend en charge la programmation orientée objet, la programmation procédurale et la programmation générique.

### **Gestion de la Mémoire :**

En Java, la gestion de la mémoire est automatisée grâce au ramasse-miettes (garbage collector). Les développeurs n'ont pas besoin de libérer la mémoire explicitement.

En C++, les développeurs sont responsables de la gestion de la mémoire. Ils doivent allouer et libérer manuellement la mémoire à l'aide d'opérateurs comme **new** et **delete**. Cela offre plus de contrôle sur les ressources mais nécessite une attention particulière pour éviter les fuites de mémoire.

### **Syntaxe :**

Java a une syntaxe plus simple et plus uniforme par rapport à C++. Par exemple, Java n'a pas de pointeurs explicites, ce qui simplifie la gestion des références mémoire.

C++ a une syntaxe plus complexe avec des fonctionnalités avancées comme les pointeurs, les références, les opérateurs de surcharge, etc. Cela offre plus de flexibilité mais peut être plus difficile à maîtriser.

### **Portabilité :**

Java est conçu pour être "write once, run anywhere" (écrire une fois, exécuter partout).

Les programmes Java sont compilés en bytecode qui peut être exécuté sur n'importe quelle machine virtuelle Java (JVM) sans modification.

C++ est un langage compilé qui génère du code machine spécifique à une plate-forme.

Les programmes C++ doivent être recompilés pour chaque plate-forme cible, ce qui peut affecter la portabilité.

### **Gestion des Exceptions :**

En Java, les exceptions sont intégrées au langage et sont gérées à l'aide des blocs try-catch-finally. Les exceptions doivent être déclarées dans la signature de méthode ou être des sous-classes de RuntimeException.

En C++, les exceptions sont également gérées à l'aide de blocs try-catch mais la gestion des exceptions est plus flexible et les exceptions ne sont pas toujours déclarées dans la signature de méthode.

### **Héritage :**

Java supporte uniquement l'héritage simple, ce qui signifie qu'une classe ne peut hériter que d'une seule classe de base.

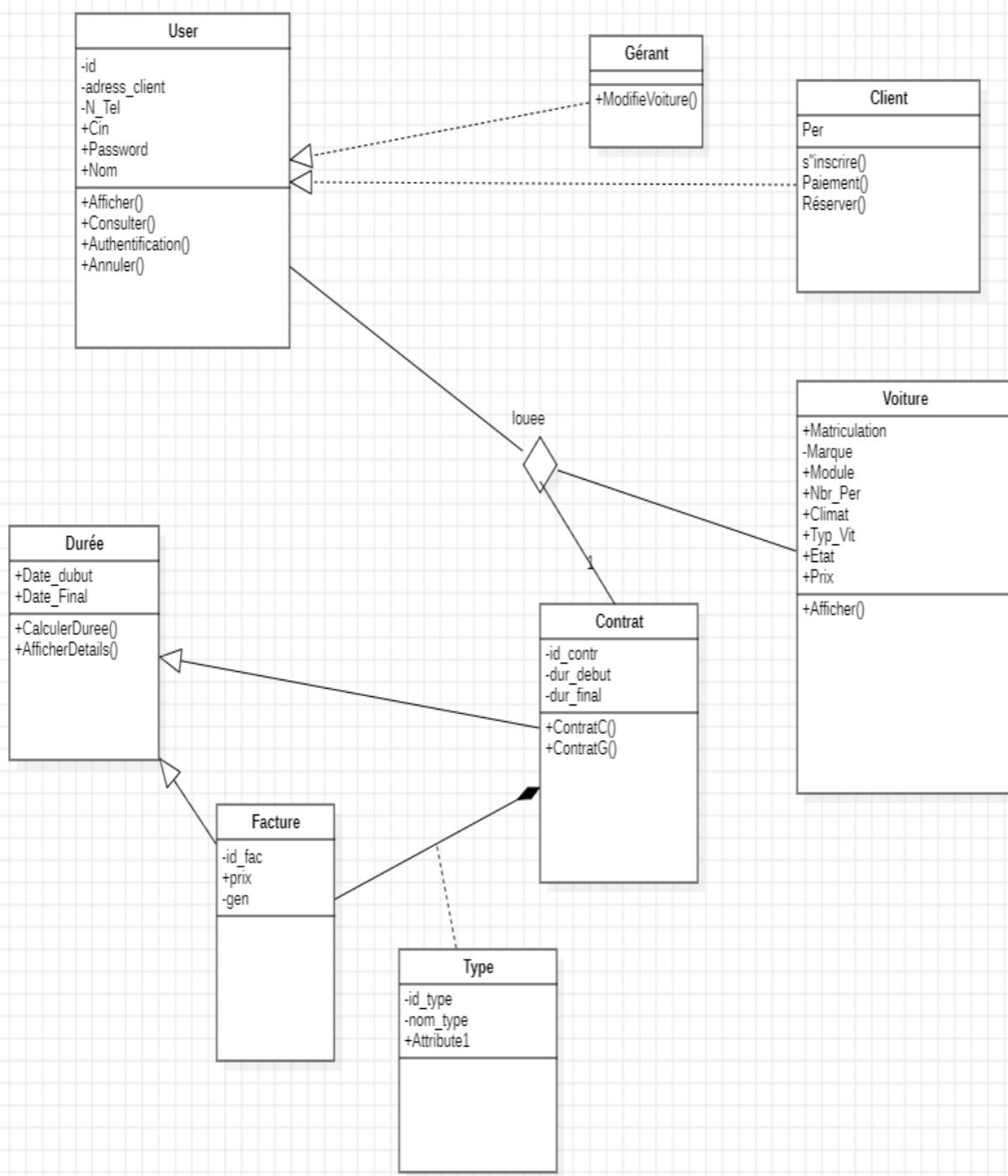
C++ prend en charge à la fois l'héritage simple et multiple, ce qui permet à une classe d'hériter de plusieurs classes de base.

### **Gestion des Types :**

Java est strictement typé avec vérification de type à la compilation. Il utilise le concept de polymorphisme pour prendre en charge le lien tardif (dynamique).

C++ offre plus de contrôle sur les types avec la possibilité d'utiliser des pointeurs et des références. Il prend en charge le polymorphisme à la fois statique (surcharge de fonctions) et dynamique.

f. Diagramme de Classe :



*g. Qu'est-ce que l'Architecture MVC :*

L'architecture MVC (Model-View-Controller) est un modèle architectural utilisé pour organiser le code dans les applications, principalement dans le développement de logiciels et les applications web. Elle permet de séparer les différentes préoccupations de l'application, facilitant ainsi la gestion, la maintenance et l'évolutivité du code. Voici une description détaillée des trois composants principaux du modèle MVC :

➤ **Modèle (Model) :**

Le modèle représente la logique de données de l'application. Il gère les données et les règles métier. Le modèle est responsable de :

- **Gestion des données** : Stockage, récupération et mise à jour des données, souvent en interaction avec une base de données.
- **Règles métier** : Logique qui définit comment les données peuvent être modifiées et quelles opérations sont autorisées.
- **Notifications** : Informer les vues (View) des changements de données afin qu'elles puissent se mettre à jour.

**Exemple** : Dans une application de gestion de livres, le modèle pourrait être une classe Book qui contient des attributs comme title, author, et ISBN, et des méthodes pour enregistrer, modifier et supprimer des livres.

➤ **Vue (View)**

La vue est responsable de la présentation des données. Elle définit l'interface utilisateur et la manière dont les informations sont affichées. La vue :

- **Affiche les données** : Prend les données du modèle et les affiche à l'utilisateur.
- **Interactions utilisateur** : Capte les interactions de l'utilisateur (comme les clics, les saisies de texte) et les transmet au contrôleur.

**Exemple** : Dans l'application de gestion de livres, la vue pourrait être une page HTML/CSS qui affiche une liste de livres ou un formulaire pour ajouter un nouveau livre.

### ➤ Contrôleur (Controller)

Le contrôleur sert d'intermédiaire entre le modèle et la vue. Il gère les entrées utilisateur et met à jour le modèle en conséquence. Le contrôleur :

- **Gestion des requêtes utilisateur** : Reçoit les entrées de l'utilisateur via la vue et les traite.
- **Mise à jour du modèle** : Met à jour les données du modèle en fonction des actions de l'utilisateur.
- **Mise à jour de la vue** : Demande à la vue de se mettre à jour en fonction des changements dans le modèle.

**Exemple** : Dans l'application de gestion de livres, le contrôleur pourrait être une classe BookController qui gère les requêtes pour ajouter, modifier ou supprimer un livre. Il prend les données saisies par l'utilisateur, les envoie au modèle pour traitement, puis demande à la vue de s'actualiser.

### ➤ Interaction entre les composants

- **L'utilisateur interagit avec la vue** (par exemple, en cliquant sur un bouton ou en soumettant un formulaire).
- **La vue envoie les données au contrôleur** sous forme d'événements ou de requêtes.
- **Le contrôleur traite les requêtes** et interagit avec le modèle pour récupérer ou mettre à jour les données.
- **Le modèle change ses données** et notifie les vues des modifications.
- **La vue se met à jour** en fonction des nouvelles données du modèle.

### ➤ Avantages de l'architecture MVC

- **Séparation des préoccupations** : Facilite la maintenance et la gestion du code en séparant clairement la logique métier, la présentation et la gestion des entrées utilisateur.
- **Réutilisabilité** : Les composants peuvent être réutilisés plus facilement. Par exemple, une même vue peut être utilisée avec différents modèles.
- **Testabilité** : Le code est plus facile à tester car les composants peuvent être testés indépendamment.

- **Collaboration :** Facilite la collaboration entre développeurs front-end (vues) et back-end (modèles et contrôleurs).

En résumé, l'architecture MVC est une manière structurée d'organiser le code d'une application pour améliorer la maintenance, la réutilisabilité et la testabilité, tout en séparant les responsabilités de chaque composant de l'application.

## IV. JAVA :

Ce projet de gestion de location de voitures couvre les principales fonctionnalités nécessaires pour gérer les voitures, authentifier les utilisateurs, permettre aux clients de consulter les voitures disponibles, s'inscrire, et réserver des voitures. Le code est structuré en différentes classes pour une meilleure organisation et maintenabilité.

### 1. Architecture du Système :

***Le projet est divisé en plusieurs classes principales :***

#### a. *Classe user :*

**Objectif :** La classe User est une classe abstraite représentant un utilisateur générique du système. Elle contient des méthodes abstraites que les sous-classes doivent implémenter, ainsi que des méthodes utilitaires communes.

#### **Attributs:**

**id:** Identifiant de l'utilisateur.

**Nom :** Nom de l'utilisateur.

**addr:** Adresse de l'utilisateur.

**tel:** Numéro de téléphone de l'utilisateur.

**email:** Adresse email de l'utilisateur.

**sc:** Scanner pour la lecture des entrées de l'utilisateur.

#### **Méthodes:**

##### **Méthode statique afficher:**

- Affiche des informations génériques sur l'utilisateur. (à implémenter)

##### **Méthode abstraite consulter:**

- Doit être implémentée par les sous-classes pour permettre à l'utilisateur de consulter des informations spécifiques.

##### **Méthode abstraite authentification:**

- Doit être implémentée par les sous-classes pour gérer l'authentification de l'utilisateur.

### Méthode abstraite modifierVoiture:

- Doit être implémentée par les sous-classes pour permettre à l'utilisateur de modifier des informations de voiture.

### Méthode annuler:

- Permet à l'utilisateur d'annuler une opération en saisissant "Annuler".

```

import java.util.Scanner;

public abstract class User {
    protected int id;
    protected String nom;
    protected String addr;
    protected int tel;
    protected String email;
    protected static Scanner sc = new Scanner(System.in);

    public static void afficher() {
    }

    public abstract void consulter();
    public abstract void authentification();
    public abstract void modifierVoiture();

    public void annuler() throws InterruptedException {
        System.out.println("Si vous souhaitez annuler, saisissez 'Annuler' : ");
        String annuler = sc.next();
        while (!annuler.equalsIgnoreCase("Annuler")) {
            System.out.println("Saisie incorrecte. Réessayez : ");
            annuler = sc.next();
        }
        System.exit(0);
    }
}

```

*b. Classe gérant (hérite d'User) :*

La classe gérant hérite de la classe abstraite User et représente un type spécifique d'utilisateur dans le système de gestion de voitures. Elle inclut des attributs spécifiques, Les méthodes implémentées et des informations de connexion à la base de données.

La classe gérant étend la classe User et représente un gérant dans un système de gestion de véhicules. Cette classe permet au gérant de consulter les voitures, de s'authentifier, de modifier l'état des voitures et de gérer les contrats de location.

Composants du Code :

**Attributs :**

- **URL, USER, PASSWORD** : Constantes pour les paramètres de connexion à la base de données.
- **SCANNER** : Scanner pour lire les entrées utilisateur depuis la console.
- **idGerant, nomGerant**: Identifiant et nom du gérant.

```
3
4+import java.sql.Connection;[]
5
6 public class gérant extends User {
7     private static final String URL = "jdbc:mysql://localhost:3306/DBV";
8     private static final String USER = "root";
9     private static final String PASSWORD = "Houssam123@.";
10    private static final Scanner SCANNER = new Scanner(System.in);
11
12    private int idGerant;
```

**Méthodes :**

## Constructeur :

```
public gérant(int id, String nom) {  
    this.idGerant = id;  
    this.nomGerant = nom;  
}
```

- **gérant(int id, String nom):** Initialise les variables idGerant et nomGerant avec les valeurs fournies.

## Consulter():

```
@Override  
public void consulter() {  
    try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);  
        PreparedStatement pstmt = con.prepareStatement("SELECT * FROM voiture")) {  
  
        ResultSet rs = pstmt.executeQuery();  
        while (rs.next()) {  
            String marque = rs.getString("Marque");  
            String matriculation = rs.getString("Matriculation");  
            String etat = rs.getString("Etat");  
            System.out.println("Voiture: " + marque + " - Matriculation: " + matriculation + " - Etat: " + etat);  
        }  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

- Établit une connexion à la base de données.
- Exécute une requête SQL pour récupérer toutes les voitures.
- Parcourt le jeu de résultats et affiche les détails de chaque voiture (marque, immatriculation, état).

- Utilise le bloc try-with-resources pour assurer la fermeture automatique des ressources (Connection, PreparedStatement, ResultSet).

### **authentification():**

```

• @Override
public void authentification() {
    boolean conn = false;
    while (!conn) {
        System.out.print("Entrez votre nom : ");
        String nom = SCANNER.next();
        System.out.print("Entrez votre mot de passe : ");
        String mdp = SCANNER.next();

        try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement pstmt = con.prepareStatement("SELECT * FROM gerant WHERE Nom = ? AND Password = ?")) {

            pstmt.setString(1, nom);
            pstmt.setString(2, mdp);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                System.out.println("Connexion réussie !");
                conn = true;
            } else {
                System.out.println("Nom ou mot de passe incorrect. Veuillez réessayer.");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

- Demande à l'utilisateur de saisir son nom et son mot de passe.
- Établit une connexion à la base de données et exécute une requête SQL pour vérifier les identifiants.
- Si les identifiants sont corrects, affiche un message de succès et met fin à la boucle.
- Si les identifiants sont incorrects, affiche un message d'erreur et redemande les informations.
- Utilise le bloc try-with-resources pour gérer les ressources.

## modifierVoiture() :

```
69
70    @Override
71    public void modifierVoiture() {
72        try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
73             PreparedStatement pstmt = con.prepareStatement("UPDATE voiture SET Etat = ? WHERE Matriculation = ?")) {
74
75            System.out.print("Entrez la matriculation de la voiture à modifier : ");
76            String matriculation = SCANNER.next();
77            System.out.print("Entrez le nouvel état de la voiture : ");
78            String nouvelEtat = SCANNER.next();
79
80            pstmt.setString(1, nouvelEtat);
81            pstmt.setString(2, matriculation);
82
83            int rowsAffected = pstmt.executeUpdate();
84            if (rowsAffected > 0) {
85                System.out.println("État de la voiture modifié avec succès.");
86            } else {
87                System.out.println("Erreur lors de la modification de l'état de la voiture.");
88            }
89        } catch (Exception e) {
90            e.printStackTrace();
91        }
92    }
93 }
```

- Demande à l'utilisateur de saisir l'immatriculation de la voiture à modifier et le nouvel état.
- Établit une connexion à la base de données et exécute une requête SQL UPDATE pour modifier l'état de la voiture.
- Affiche un message de succès ou d'erreur en fonction du résultat de la mise à jour.
- Utilise le bloc try-with-resources pour gérer les ressources.

### **annuler():**

- Appelle la méthode annuler de la classe parente User.

### **gererContrats():**

- Appelle la méthode contratG de la classe Contrat pour gérer les contrats.

```
94
95     @Override
96     public void annuler() throws InterruptedException {
97         super.annuler();
98     }
99
00     public void gererContrats() {
01         Contrat.contratG();
02     }
03 }
04
```

### c. Classe contact :

La classe Contrat permet de gérer les contrats de location de voitures en se connectant à une base de données MySQL. Cette classe fournit des méthodes pour consulter les détails des contrats et pour afficher les informations relatives à une voiture spécifique.

#### Variables:

- **URL, USER, PASSWORD** : Constantes pour les paramètres de connexion à la base de données.
- **SCANNER** : Scanner pour lire les entrées utilisateur depuis la console.

#### Méthodes :

##### contratG():

```
1 import java.sql.Connection;
2
3 public class Contrat {
4     private static final String URL = "jdbc:mysql://localhost:3306/DBV";
5     private static final String USER = "root";
6     private static final String PASSWORD = "Houssam123@.";
7     private static final Scanner SCANNER = new Scanner(System.in);
8
9     public static void contratG() {
10         try {
11             System.out.println("Entrez la matricule de votre choix : ");
12             String choix = SCANNER.next();
13
14             try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
15                  PreparedStatement pstmt =
16                  con.prepareStatement("SELECT v.*, c.* FROM voiture v INNER JOIN contrat c ON v.Matriculation = c.Matriculation WHERE v.Matriculation = ?")) {
17
18                 pstmt.setString(1, choix);
19                 ResultSet rs = pstmt.executeQuery();
20                 if (rs.next()) {
21                     afficherDetails(rs);
22                 } else {
23                     System.out.println("Aucun contrat trouvé pour cette matriculation.");
24                 }
25             }
26         } catch (SQLException e) {
27             e.printStackTrace();
28         }
29     }
30 }
```

- Demande à l'utilisateur de saisir la matriculation de la voiture.

- Établit une connexion à la base de données et exécute une requête SQL pour récupérer les informations de la voiture et du contrat correspondant à la matriculation.
- Si un contrat est trouvé, appelle la méthode afficherDetails pour afficher les détails.
- Utilise le bloc try-with-resources pour assurer la fermeture automatique des ressources (Connection, PreparedStatement, ResultSet).

### contratC(String choix) :

```

public void contratC(String choix) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
        PreparedStatement pstmt =
        con.prepareStatement("SELECT v.*, c.* FROM voiture v INNER JOIN contrat c ON v.Matriculation = c.Matriculation WHERE v.Matriculation = ?")) {

        pstmt.setString(1, choix);
        ResultSet rs = pstmt.executeQuery();
        if (rs.next()) {
            afficherDetails(rs);
        } else {
            System.out.println("Aucun contrat trouvé pour cette matriculation.");
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

- Accepte une chaîne de caractères choix représentant la matriculation de la voiture.
  - Établit une connexion à la base de données et exécute une requête SQL pour récupérer les informations de la voiture et du contrat correspondant à la matriculation.
  - Si un contrat est trouvé, appelle la méthode afficherDetails pour afficher les détails.
  - Utilise le bloc try-with-resources pour gérer les ressources.

### afficherDetails(ResultSet rs) :

```
6
7  private static void afficherDetails(ResultSet rs) throws SQLException {
8      String marque = rs.getString("Marque");
9      String matriculation = rs.getString("Matriculation");
10     System.out.println("Marque: " + marque + " - Matriculation: " + matriculation);
11
12     int idContrat = rs.getInt("Id_Contrat");
13     int duree = rs.getInt("Duree");
14     int idFacture = rs.getInt("Id_Facture");
15     int idClient = rs.getInt("Id_Client");
16
17     System.out.println("Contrat: " + idContrat);
18     System.out.println("Durée: " + duree);
19     System.out.println("Facture: " + idFacture);
20     System.out.println("Client: " + idClient);
21 }
22 }
```

- Récupère et affiche les détails de la voiture et du contrat à partir du ResultSet passé en paramètre.
- Affiche la marque, la matriculation de la voiture ainsi que les détails du contrat (ID du contrat, durée, ID de la facture, ID du client).

#### *d. Classe Facture*

La classe Facture est responsable de la gestion des factures dans le système. Cela inclut la génération de nouvelles factures et l'affichage des informations de factures existantes.

#### **Attributs :**

- **URL** : URL de la base de données MySQL.
- **USER** : Nom d'utilisateur pour la connexion à la base de données.
- **PASSWORD** : Mot de passe pour la connexion à la base de données.
- **SCANNER** : Scanner pour la lecture des entrées de l'utilisateur.
- **idFacture** : Identifiant de la facture.
- **Montant** : Montant de la facture.
- **idContrat** : Identifiant du contrat associé à la facture.

#### **Méthodes :**

##### **Constructeur Facture :**

- Initialise les attributs idFacture, montant, et idContrat.

##### **Méthode afficherFacture :**

- Affiche les détails de la facture.

##### **Méthode statique genererFacture :**

- Prend idContrat et montant comme paramètres.
- Se connecte à la base de données.
- Insère une nouvelle facture dans la table facture.
- Affiche un message de succès ou d'erreur selon le résultat de l'opération.

```

30 import java.sql.Connection;
8
9 public class Facture {
10     private static final String URL = "jdbc:mysql://localhost:3306/DBV";
11     private static final String USER = "root";
12     private static final String PASSWORD = "Houssam123@.";
13     private static final Scanner SCANNER = new Scanner(System.in);
14
15     private int idFacture;
16     private int montant;
17     private int idContrat;
18
19     public Facture(int idFacture, int montant, int idContrat) {
20         this.idFacture = idFacture;
21         this.montant = montant;
22         this.idContrat = idContrat;
23     }
24
25     public void afficherFacture() {
26         System.out.println("Facture ID: " + idFacture);
27         System.out.println("Montant: " + montant);
28         System.out.println("Contrat ID: " + idContrat);
29     }
30
31     public static void genererFacture(int idContrat, int montant) {
32         try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
33              PreparedStatement pstmt = con.prepareStatement("INSERT INTO facture (Id_Contrat, Montant) VALUES (?, ?)")) {
34
35             pstmt.setInt(1, idContrat);
36             pstmt.setInt(2, montant);
37
38             int rowsAffected = pstmt.executeUpdate();
39             if (rowsAffected > 0) {
40                 System.out.println("Facture générée avec succès.");
41             } else {
42                 System.out.println("Erreur lors de la génération de la facture.");
43             }
44
45         } catch (Exception e) {
46             e.printStackTrace();
47         }
48     }
49 }
```

e. *Claas Client* :

#### **Attributs :**

- **id** : Identifiant unique du client.
- **addr** : Adresse du client.
- **tel** : **int** - Numéro de téléphone du client.
- **cin** : **int** - Carte d'identité nationale du client.
- **permis** : Indique si le client possède un permis de conduire.
- **choix** : Stocke la matriculation de la voiture choisie par le client.
- **nom** : Nom du client (hérité de la classe User).

#### **Méthodes**

##### **Client(int id, String nom, String addr, int tel, int cin) :**

- Constructeur de la classe Client.
- Initialise les attributs du client avec les valeurs fournies.

##### **consulter() :**

- Affiche la liste des voitures disponibles.
- Se connecte à la base de données, exécute une requête pour sélectionner les voitures disponibles (Etat = 'disponible'), et affiche les résultats.

##### **authentification() :**

- Authentifie le client en demandant son nom et son mot de passe.
- Tant que l'authentification échoue, la méthode continue de demander les informations.
- Se connecte à la base de données pour vérifier les informations d'identification.

##### **modifierVoiture() :**

- Méthode non implémentée dans le code fourni, mais serait utilisée pour permettre au client de modifier des informations sur une voiture.

### **inscrire() :**

- Permet au client de s'inscrire en remplissant un formulaire.
- Demande des informations telles que le nom, l'adresse, le numéro de téléphone, le CIN, et si le client possède un permis.
- Insère les données dans la base de données si l'utilisateur confirme les informations.

### **paiement() :**

- Gère les choix de paiement pour le client (chèque, en ligne, en espèce).
- Affiche des messages en fonction du type de paiement choisi.

### **reserver() :**

- Permet au client de consulter les voitures disponibles et de réserver une voiture en choisissant sa matriculation.
- Crée un contrat pour la réservation.

### **annuler() :**

- Méthode héritée de la classe User qui permet d'annuler une action. Utilise super.annuler() pour appeler la méthode de la classe parente.

```

3• import java.sql.Connection;□
8
9 public class Client extends User {
10     private static final String URL = "jdbc:mysql://localhost:3306/dbv";
11     private static final String USER = "root";
12     private static final String PASSWORD = "Houssam123@ ";
13     private static final Scanner SCANNER = new Scanner(System.in);
14
15     private int id;
16     private String addr;
17     private int tel;
18     private int cin;
19     private boolean permis;
20     private String choix;
21
22•     public Client(int id, String nom, String addr, int tel, int cin) {
23         this.id = id;
24         this.nom = nom;
25         this.addr = addr;
26         this.tel = tel;
27         this.cin = cin;
28     }
29
30•     @Override
31     public void consulter() {
32         try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
33             PreparedStatement pstmt = con.prepareStatement("SELECT * FROM voiture WHERE Etat = 'disponible'")) {
34
35             ResultSet rs = pstmt.executeQuery();
36             while (rs.next()) {
37                 String marque = rs.getString("Marque");
38                 String matriculation = rs.getString("Matriculation");
39                 System.out.println("Voiture: " + marque + " - Matriculation: " + matriculation);
40             }
41
42         } catch (Exception e) {
43             e.printStackTrace();
44         }
45     }
46
47
48     public void authentification() {
49         boolean conn = false;
50         while (!conn) {
51             System.out.print("Entrez votre nom : ");
52             String nom = SCANNER.next();
53             System.out.print("Entrez votre mot de passe : ");
54             String mdp = SCANNER.next();
55
56             try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
57                  PreparedStatement pstmt = con.prepareStatement("SELECT * FROM client WHERE Nom = ? AND Password = ?")) {
58
59                 pstmt.setString(1, nom);
60                 pstmt.setString(2, mdp);
61                 ResultSet rs = pstmt.executeQuery();
62                 if (rs.next()) {
63                     System.out.println("Connexion réussie !");
64                     conn = true;
65                 } else {
66                     System.out.println("Nom ou mot de passe incorrect. Veuillez réessayer.");
67                 }
68
69             } catch (Exception e) {
70                 e.printStackTrace();
71             }
72         }
73     }
74
75•     @Override
76     public void modifierVoiture() {
77         // Implémentation de la modification de voiture par le client
78     }
79
80•     public void inscrire() {
81         System.out.println("Veuillez remplir le formulaire : ");
82         System.out.print("Saisir votre nom : ");
83         nom = SCANNER.next();
84         System.out.print("Saisir votre adresse : ");
85         addr = SCANNER.next();
86         System.out.print("Saisir votre numéro de téléphone : ");
87         tel = SCANNER.nextInt();
88         System.out.print("Saisir votre CIN : ");
89         cin = SCANNER.nextInt();
90

```

```

System.out.println("Avez-vous un permis ?");
System.out.println("1 - Oui");
System.out.println("2 - Non");
int op = SCANNER.nextInt();
while (op != 1 && op != 2) {
    System.out.println("Réponse incorrecte. Réessayez.");
    op = SCANNER.nextInt();
}
permis = (op == 1);

System.out.println("1 - Confirmer vos données");
System.out.println("2 - Annuler");
op = SCANNER.nextInt();
while (op != 1 && op != 2) {
    System.out.println("Choix incorrect. Réessayez.");
    op = SCANNER.nextInt();
}
if (op == 1) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);
        PreparedStatement pstmt = con.prepareStatement("INSERT INTO client (Id_client, Nom, Cin, Adr, N_tel, per, Id_Ger) VALUES (?, ?, ?, ?, ?, ?, ?")) {
        pstmt.setInt(1, id);
        pstmt.setString(2, nom);
        pstmt.setInt(3, cin);
        pstmt.setString(4, addr);
        pstmt.setInt(5, tel);
        pstmt.setBoolean(6, permis);
        pstmt.setInt(7, 1); // ID_Ger placeholder

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Inscription réussie !");
        } else {
            System.out.println("Erreur lors de l'inscription.");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
} else {
    System.out.println("Formulaire annulé.");
}

```

```

4
5● public void paiement() {
6     System.out.println("Choisissez le type de paiement : ");
7     System.out.println("1 - Par chèque");
8     System.out.println("2 - En ligne");
9     System.out.println("3 - En espèce");
0     int op = SCANNER.nextInt();
1     while (op < 1 || op > 3) {
2         System.out.println("Choix incorrect. Réessayez.");
3         op = SCANNER.nextInt();
4     }
5     switch (op) {
6         case 1:
7             System.out.println("Veuillez apporter votre chèque en main. Continuez votre réservation.");
8             break;
9         case 2:
0             System.out.println("Ce type de paiement n'est pas disponible. Nous passons à un paiement en espèce.");
1             break;
2         case 3:
3             System.out.println("Paiement enregistré avec succès. Continuez votre réservation.");
4             break;
5     }
6 }
7● public void reserver() {
8     consulter();
9     System.out.println("Entrez la matriculation de votre choix : ");
0     choix = SCANNER.next();
1     Contrat contrat = new Contrat();
2     contrat.contratC(choix);
3 }
4
5● @Override
6     public void annuler() throws InterruptedException {
7         super.annuler();
8     }
9 }

```

f. *Class Voiture :*

### Attributs

- **marque** : La marque de la voiture.
- **matriculation** : Le numéro de matriculation de la voiture.
- **etat** : L'état de la voiture (par exemple, "disponible", "louée").

### g

**Voiture(String marque, String matriculation, String etat) :**

- Initialise les attributs marque, matriculation, et etat avec les valeurs fournies.

### Méthodes

**getMarque()**

- Retourne la marque de la voiture.

**setMarque(String marque)**

- Définit la marque de la voiture.

**getMatriculation()**

- Retourne le numéro de matriculation de la voiture.

**setMatriculation(String matriculation)**

- Définit le numéro de matriculation de la voiture.

**getEtat()**

- Retourne l'état de la voiture

## setEtat(String etat)

- Définit l'état de la voiture.

## toString()

- Retourne une représentation textuelle de l'objet Voiture.

```
3 public class Voiture {  
4     private String marque;  
5     private String matriculation;  
6     private String etat;  
7  
8     public Voiture(String marque, String matriculation, String etat) {  
9         this.marque = marque;  
10        this.matriculation = matriculation;  
11        this.etat = etat;  
12    }  
13  
14    public String getMarque() {  
15        return marque;  
16    }  
17  
18    public void setMarque(String marque) {  
19        this.marque = marque;  
20    }  
21  
22    public String getMatriculation() {  
23        return matriculation;  
24    }  
25  
26    public void setMatriculation(String matriculation) {  
27        this.matriculation = matriculation;  
28    }  
29  
30    public String getEtat() {  
31        return etat;  
32    }  
33  
34    public void setEtat(String etat) {  
35        this.etat = etat;  
36    }  
37  
38    @Override  
39    public String toString() {  
40        return "Voiture{" +  
41            "marque='" + marque + '\'' +  
42            ", matriculation='" + matriculation + '\'' +  
43            ", etat='" + etat + '\'' +  
44            '}';  
45    }
```

g. Class main :

```
import java.util.Scanner;

public class Main {
    private static final Scanner SCANNER = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("Bienvenue dans l'application de gestion de location de voitures.");
        System.out.println("1. Client");
        System.out.println("2. Gérant");
        System.out.print("Choisissez votre rôle : ");
        int choix = SCANNER.nextInt();

        switch (choix) {
            case 1:
                Client client = new Client(1, "Houssam", "hey labyayed", 123456789, 123456);
                client.authentification();
                client.consulter();
                client.reserver();
                break;

            case 2:
                gerant gerant = new gerant(1, "Houssam");
                gerant.authentification();
                gerant.consulter();
                gerant.modifierVoiture();
                gerant.gererContrats();
                break;

            default:
                System.out.println("Choix non valide.");
        }
    }
}
```

## Résultats (en tant que client) :

```
Bienvenue dans l'application de gestion de location de voitures.  
1. Client  
2. Gérant  
Choisissez votre rôle : 1  
Entrez votre nom : Houssam  
Entrez votre mot de passe : azerty  
Connexion réussie !  
Voiture: Renault - Matriculation: 2345B6789  
Voiture: Peugeot - Matriculation: 3456C7890  
Voiture: Mercedes - Matriculation: 4567D8901  
Voiture: BMW - Matriculation: 5678E9012  
Voiture: Mercedes - Matriculation: aae1232  
Voiture: dacia - Matriculation: ADZD1213  
Voiture: Renault - Matriculation: 2345B6789  
Voiture: Peugeot - Matriculation: 3456C7890  
Voiture: Mercedes - Matriculation: 4567D8901  
Voiture: BMW - Matriculation: 5678E9012  
Voiture: Mercedes - Matriculation: aae1232  
Voiture: dacia - Matriculation: ADZD1213  
Entrez la matriculation de votre choix :  
2345B6789  
Marque: Renault - Matriculation: 2345B6789  
Contrat: 2  
Facture: 2
```

## Résultats (en tant que gérant) :

```
Bienvenue dans l'application de gestion de location de voitures.  
1. Client  
2. Gérant  
Choisissez votre rôle : 2  
Entrez votre nom : Ahmed  
Entrez votre mot de passe : pass1234  
Connexion réussie !  
Voiture: Dacia - Matriculation: 1234A5678 - Etat: nondisponible  
Voiture: Renault - Matriculation: 2345B6789 - Etat: Disponible  
Voiture: Peugeot - Matriculation: 3456C7890 - Etat: Disponible  
Voiture: Mercedes - Matriculation: 4567D8901 - Etat: Disponible  
Voiture: BMW - Matriculation: 5678E9012 - Etat: Disponible  
Voiture: Mercedes - Matriculation: aae1232 - Etat: disponible  
Voiture: dacia - Matriculation: ADZD1213 - Etat: disponible  
Voiture: renault - Matriculation: AZFDD231 - Etat: nondisponible  
Entrez la matriculation de la voiture à modifier : 1234A5678  
Entrez le nouvel état de la voiture : disponible  
État de la voiture modifié avec succès.  
Entrez la matricule de votre choix :  
1234A5678  
Marque: Dacia - Matriculation: 1234A5678  
Contrat: 1  
Facture: 1
```

## V. C++ :

### 1. Qu'est-ce que C++ :

C++ est un langage de programmation polyvalent et puissant qui a été conçu par Bjarne Stroustrup en 1979 aux laboratoires Bell. Il a été conçu comme une extension du langage C, d'où son nom, C++ (l'opérateur ++ en C et C++ indique l'incrémentation, symbolisant une amélioration ou une extension). C++ combine les paradigmes de la programmation procédurale, orientée objet et générique, ce qui en fait un outil très flexible pour une variété d'applications.

#### a. Caractéristiques Principales de C++ :

##### ➤ **Programmation Orientée Objet :**

C++ introduit la POO, permettant aux développeurs de créer des objets contenant des données et des fonctions. Les concepts clés de la POO en C++ incluent :

- **Classes et Objets** : Les classes définissent des types d'objets, regroupant données et fonctions.
- **Héritage** : Les classes peuvent hériter de propriétés et méthodes d'autres classes, favorisant la réutilisation de code.
- **Encapsulation** : Les données et les fonctions sont encapsulées au sein des objets, protégeant les données internes.

- **Polymorphisme** : Les fonctions et les opérateurs peuvent être redéfinis pour différents types de données.

- **Programmation Générique**
- **Gestion de la Mémoire**
- **Bibliothèque Standard (STL)**
- **Compatibilité avec le C**

#### b. Utilisations de C++ :

- **Développement de Logiciels Système**
- **Applications de Haute Performance**
- **Développement de Jeux Vidéo**
- **Logiciels Embarqué**

c. *Description de la partie du projet à coder avec C++ :*

L'objectif de ce projet est de développer une application pour la gestion d'une agence de location de voitures. Cette application permettra de réaliser toutes les opérations nécessaires pour la gestion des clients (locataires), la gestion des voitures et la gestion des locations.

➤ **Gestion des Clients (Locataires) :**

- Enregistrement des informations des clients (nom, prénom, numéro de téléphone, etc.).
- Recherche de clients par nom, numéro de téléphone, etc.
- Modification des informations des clients.
- Suppression de clients.

➤ **Gestion des Voitures :**

- Enregistrement des informations des voitures (marque, modèle, année, immatriculation, etc.).
- Recherche de voitures par marque, modèle, année, etc.
- Modification des informations des voitures.
- Suppression de voitures.

➤ **Gestion des Locations :**

- Enregistrement des contrats de location (client, voiture, date de début, date de fin, coût, etc.).
- Calcul du coût de location en fonction de la durée.
- Recherche de locations par client, voiture, date, etc.
- Modification des contrats de location.
- Suppression de contrats de location.

➤ **Inconvénients Potentiels :**

- La nécessité de gérer les exceptions et les erreurs (par exemple, dates invalides, fichiers inaccessibles, etc.).

*d. Description de la partie du projet à coder avec C++ :*

Dans la partie C++ on a travailler sur 2 classes sont Users et Gérant , alors pour détailler cette partie on utiliser l'environnement de développement CodeBlocks :

➤ **La classe Gérant :**

• **Constructeur et Destructeur :**

- Le constructeur Gerant initialise la connexion à la base de données MySQL.
- Le destructeur ~Gerant ferme la connexion à la base de données lorsqu'un objet Gerant est détruit.

• **Méthode Consulter :**

- Cette méthode récupère toutes les informations sur les voitures à partir de la base de données et les affiche à l'écran.

• **Méthode Authentification :**

- Cette méthode permet à un gérant de s'authentifier en saisissant son nom et son mot de passe.
- Si les informations sont correctes, l'accès est autorisé.

• **Méthode ModifierVoiture :**

- Cette méthode permet au gérant d'effectuer différentes actions sur les voitures :
  - Ajouter une nouvelle voiture avec des informations telles que la marque, la matriculation, l'état, le modèle, le prix, etc.
  - Supprimer une voiture existante.
  - Modifier les propriétés d'une voiture (par exemple, changer l'état, le prix, etc.).

• **Menu :**

- Vous affichez un menu avec trois options :
  - **Consulter les voitures** : Affiche toutes les informations sur les voitures dans la base de données.
  - **Modifier une voiture** : Permet au gérant d'ajouter, supprimer ou modifier les propriétés d'une voiture.
  - **Quitter** : Termine le programme.

• **Boucle principale :**

- Vous utilisez une boucle pour permettre au gérant de choisir différentes actions jusqu'à ce qu'il décide de quitter.

Cette classe hérite de la classe User et implémente toutes les méthodes virtuelles pures de User.

➤ **Voici une explication de chaque méthode :**

**Gerant(char Nom, char cin)\*\*** : Il s'agit du constructeur de la classe Gerant. Il prend deux arguments : un nom et un numéro de carte d'identité nationale (CIN). Ces informations sont probablement utilisées pour identifier le gérant.

**void Consulter()** : Cette méthode est une implémentation de la méthode virtuelle pure Consulter de la classe User. Elle pourrait être utilisée pour consulter des informations, mais sans le corps de la méthode, il est difficile de dire exactement ce qu'elle fait.

**bool Authentification()** : Cette méthode est une implémentation de la méthode virtuelle pure Authentification de la classe User. Elle retourne un booléen, ce qui suggère qu'elle pourrait être utilisée pour vérifier si l'authentification du gérant est réussie ou non.

**void ModifierVoiture()** : Cette méthode est une implémentation de la méthode virtuelle pure ModifierVoiture de la classe User. Elle pourrait être utilisée pour modifier les informations d'une voiture.

**void Annuler()** : Cette méthode est une implémentation de la méthode virtuelle pure Annuler de la classe User. Elle pourrait être utilisée pour annuler une action ou une opération.

**~Gerant()** : Il s'agit du destructeur de la classe Gerant. Il est appelé automatiquement lorsque l'objet Gerant est détruit.

En outre, cette classe contient plusieurs attributs privés, y compris des informations de connexion à une base de données MySQL et des informations sur le gérant. Notez que l'utilisation de mots de passe en texte clair dans le code est généralement une mauvaise pratique pour des raisons de sécurité. Il est préférable de stocker les mots de passe de manière sécurisée.

Et pour la classe User est une classe de base alors:

- **Constructeur par défaut (User::User()) :**

- Le constructeur par défaut est appelé lorsque vous créez un nouvel objet de la classe **User** sans fournir d'arguments.
- Dans votre cas, il ne fait rien (il est vide), mais vous pouvez y ajouter des initialisations ou d'autres actions nécessaires lors de la création d'un objet **User**.

- **Destructeur (User::~User()) :**

- Le destructeur est appelé lorsque l'objet **User** est détruit (par exemple, lorsque sa portée se termine ou lorsqu'il est supprimé explicitement).

➤ **Voici une explication de chaque méthode :**

**User()** : Il s'agit du constructeur de la classe User. Il est appelé automatiquement lorsqu'un nouvel objet de type User est créé. Actuellement, il ne fait rien car il n'a pas de corps. □

**~User()** : Il s'agit du destructeur de la classe User. Il est appelé automatiquement lorsque l'objet User est détruit. Comme le constructeur, il ne fait actuellement rien.

**void Consulter() = 0** : Il s'agit d'une méthode virtuelle pure, ce qui signifie qu'elle doit être implémentée par toute classe qui hérite de User. Cette méthode pourrait être utilisée pour consulter des informations.

**bool Authentification() = 0** : Il s'agit d'une autre méthode virtuelle pure qui doit être implémentée par les classes dérivées. Cette méthode pourrait être utilisée pour authentifier un utilisateur.

**void ModifierVoiture() = 0** : Cette méthode virtuelle pure pourrait être utilisée pour modifier les informations d'une voiture. Elle doit être implémentée par toute classe qui hérite de User.

**void Annuler() = 0** : Cette dernière méthode virtuelle pure pourrait être utilisée pour annuler une action ou une opération. Elle doit également être implémentée par toute classe qui hérite de User.

e. Captures écrans des interfaces :

Dans cette figure représente la réussite de la connexion du gérant avec la base de données.

```
*****Bienvenue*****
//////////entrez votre information///////////
Entrez votre Nom : ahmed
Entrez votre Mot de passe : pass1234
Connexion reussie !
```

Après la connexion avec la BD on voit si dessous dans la figure suivante la barre du menu avec multiples choix de fonctionnalités pour le gérant.

```
Menu:
1-Consulter les voitures
2-Modifier une voiture
3-Quitter
```

Par exemple si on choisit le premier choix on trouve la variété des voitures qui se sont disponibles dans la figure ultérieure.

```
Entrez votre choix: 1
voiture dans la base de données:
Matriculation:1234A5678 | Marque:Dacia | Modele:2020 | Etat:Disponible | Prix :150
Matriculation:2345B6789 | Marque:Renault | Modele:2021 | Etat:Disponible | Prix :200
Matriculation:3456C7890 | Marque:Peugeot | Modele:2019 | Etat:Disponible | Prix :180
Matriculation:4567D8901 | Marque:Mercedes | Modele:2022 | Etat:Disponible | Prix :300
Matriculation:5678E9012 | Marque:BMW | Modele:2023 | Etat:Disponible | Prix :350
```

Ensute dans le menu on choisit 2 choix nous donne d'autres choix pour des modifications sur les voitures et qui seront modifier mêmes dans la BD.

```
Entrez votre choix: 2
saisir votre choix
1-Ajouter Une Voiture
2-Supprimer Une Voiture
3-Modifier une proprete dans une Voiture
1
```

Apres l'étape précédente on voit les modifications qui ont fait par le gérant et qui se sont modifier avec succès dans la BD

```
ajouter la marque:
Dacai
Matriculation:
999999
Etat:
Disponible
Modele:
2021
Prix:
300
Nombre de personne :
5
Avec Climat (0 for false, 1 for true):
1
Type de vitesse :
auto
La Voiture a ÚtÚ ajoutÚe avec succ s !!!
```

Pour le choix supprimer on voit dans la figure suivante que la le choix à était fait avec succès.

```
Entrez votre choix: 2
saisir votre choix
1-Ajouter Une Voiture
2-Supprimer Une Voiture
3-Modifier une proprite dans une Voiture
2
Entrer La Matriculation de Voiture
999999
La Voiture ete SupprimU Avec succPs !!!
```

Et pour modifier quelques personnalisations sur une voiture il suffit d'entrer l'immatriculation et ensuite de spécifier les modifications.

```
Entrez la Matriculation de la Voiture Ó modifier: 1234A5678
Entrez la nouvelle Matriculation (ou tapez '--' pour ne pas changer): 999999
Entrez la nouvelle Marque (ou tapez '--' pour ne pas changer): Dacia
Entrez le nouvel Etat (ou tapez '--' pour ne pas changer): Disponible

Les modifications ont ÚtU effectuÙes avec succPs !!!

Menu:
1-Consulter les voitures
2-Modifier une voiture
3-Quitter
Entrez votre choix: 1
voiture dans la base de donnÙes:
Matriculation:999999 | Marque:Dacia | Modele:2020 | Etat:Disponible | Prix :150
Matriculation:2345B6789 | Marque:Renault | Modele:2021 | Etat:Disponible | Prix :200
Matriculation:3456C7890 | Marque:Peugeot | Modele:2019 | Etat:Disponible | Prix :180
Matriculation:4567D8901 | Marque:Mercedes | Modele:2022 | Etat:Disponible | Prix :300
Matriculation:5678E9012 | Marque:BMW | Modele:2023 | Etat:Disponible | Prix :350
```

f. *Code source des différentes classes :*

Dans le cadre de ce projet, nous allons développer une application de location de voiture en utilisant la programmation orientée objet en C++. Les deux principales classes que nous allons implémenter sont User et Gérant. La classe User représente un utilisateur général de notre système, tandis que la classe Gérant représente un gérant ayant des priviléges spécifiques pour gérer les locations de voiture.

**Class User :**

La classe User représente un utilisateur du système de location de voiture. Chaque utilisateur a un identifiant, un nom et une adresse e-mail.

Code Source :

```
#ifndef USER_H
#define USER_H

class User
{
public:
    User();
    virtual ~User();
    virtual void Consulter() = 0;
    virtual bool Authentification() = 0;
    virtual void ModifierVoiture() = 0;
    virtual void Annuler() = 0;

protected:

private:
};

#endif // USER_H
```

## Class Gérant :

La classe Gérant hérite d'User et ajoute des privilèges spécifiques liés à la gestion de la location de voitures. Un gérant a des privilèges de gestion des voitures et des utilisateurs.

### Code source :

```
#include "Gerant.h"
#include <iostream>
#include <cstdio>
#include <cstring>

Gerant::Gerant( char* Nom,  char* cin) : server("localhost"), user("root"), password("Houssam123@."), database("DBV"){

    conn = mysql_init(NULL);

    if (!conn) {
        std::cerr << "Erreur d'initialisation MySQL : " << mysql_error(conn) << std::endl;
        exit(1);
    }

    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
        std::cerr << "Erreur de connexion MySQL : " << mysql_error(conn) << std::endl;
        exit(1);
    }
}

Gerant::~Gerant()
{
    if (conn) {
        mysql_close(conn);
    }
}

void Gerant::Consulter()
{
    conn=mysql_init(NULL);
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)){
        std::cerr<<"erreur de connexion mysql :"<<mysql_error(conn)<<std::endl;
        return;
    }

    if (mysql_query(conn, "SELECT * FROM voiture")) {
        std::cerr<<"erreurs de requête mysql :"<<mysql_error(conn)<<std::endl;
        return;
    }

    res = mysql_use_result(conn);

    std::cout<<"voiture dans la base de données:"<<std::endl;
    while((row=mysql_fetch_row(res))!=0){
        std::cout<<"Matriculation:"<<row[0]<<" | Marque:"<<row[1]<<" | Modèle:"<<row[2]<<" | Etat:"<<row[6]<<" | Prix :"<<row[7]<<std::endl;
    }
}

mysql_free_result(res);
mysql_close(conn);
}

//*****Authentification*****
bool Gerant::Authentification() {
    char Nom[50], Password[50];
    bool authentifie = false;

    do {
        std::cout<<"*****Bienvenue*****";
        std::cout<<"\n//////////entrez votre information/////////\n";
        std::cout << "Entrez votre Nom : ";
        std::cin >> Nom;
        std::cout << "Entrez votre Mot de passe : ";
        std::cin >> Password;

        char query[256];
        sprintf(query, sizeof(query), "SELECT * FROM gerant WHERE Nom = '%s' AND Password = '%s'", Nom, Password);
        if (mysql_query(conn, query)) {
            std::cerr << "erreur de requête MySQL : " << mysql_error(conn) << std::endl;
            return false;
        }

        res = mysql_use_result(conn);

        if ((row = mysql_fetch_row(res)) != NULL) {
            std::cout << "Connexion réussie !" << std::endl;
            authentifie = true;
        }
    } while(!authentifie);
}
```

```

        } else {
            std::cout << "Nom ou mot de passe incorrect." << std::endl;
        }

        mysql_free_result(res);
    } while (!authentifie);

    return true;
}

void Gerant::ModifierVoiture() {
    int choix;
    bool authentifie = false;
    std::cout << "saisir votre choix" << std::endl;
    std::cout << "1-Ajouter Une Voiture" << std::endl;
    std::cout << "2-Supprimer Une Voiture" << std::endl;
    std::cout << "3-Modifier une propreté dans une Voiture" << std::endl;
    std::cin >> choix;

    while ((choix!=1) && (choix!=2) && (choix!=3)) {
        std::cout << "votre choix incorrecte";
        std::cin >> choix;
    }
    switch(choix) {
        /*****choix1*****/
        case 1:

            char Marque[50];
            char Matriculation[50];
            char Etat[50];
            int Module;

            char Matriculation[50];
            char Etat[50];
            int Module;
            double Prix;
            char Typ_Vit[20];
            bool Climat;
            int Nbr_Per;

            std::cout << "ajouter la marque:" << std::endl;
            std::cin >> Marque;
            std::cout << "Matriculation:" << std::endl;
            std::cin >> Matriculation;
            std::cout << "Etat:" << std::endl;
            std::cin >> Etat;
            std::cout << "Modele:" << std::endl;
            std::cin >> Module;
            std::cout << "Prix:" << std::endl;
            std::cin >> Prix;
            std::cout << "Nombre de personnes :" << std::endl;
            std::cin >> Nbr_Per;

            std::cout << "Avec Climat (0 for false, 1 for true):" << std::endl;
            int tempClimat;
            std::cin >> tempClimat;
            Climat = static_cast<bool>(tempClimat);

            std::cout << "Type de vitesse :" << std::endl;
            std::cin >> Typ_Vit;

            conn = mysql_init(NULL);
            if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {

```

```

conn = mysql_init(NULL);
if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
    std::cerr << "Erreur de connexion MySQL :" << mysql_error(conn) << std::endl;
}

char query[512];
sprintf(query, sizeof(query), "INSERT INTO voiture (Matriculation, Marque, Etat, Module, Prix, Nbr_Per, Climat, Typ_Vit) VALUES ('%s', '%s', '%s', '%d', '%f', '%d',
    Matriculation, Marque, Etat, Module, Prix, Nbr_Per, Climat, Typ_Vit);");
if (mysql_query(conn, query)) {
    std::cerr << "Erreur de requête MySQL :" << mysql_error(conn) << std::endl;
    mysql_close(conn);
    return;
}

std::cout << "La Voiture a été ajoutée avec succès !!!" << std::endl;
mysql_close(conn);

break;
    /*****choix2*****/
}

case 2:

std::cout << "Entrer La Matriculation de Voiture" << std::endl;
std::cin >> Matriculation;

conn = mysql_init(NULL);

    std::cout << "Entrer La Matriculation de Voiture" << std::endl;
    std::cin >> Matriculation;

conn = mysql_init(NULL);

if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
    std::cerr << "Erreur de connexion MySQL :" << mysql_error(conn) << std::endl;
    return;
}

sprintf(query, sizeof(query), "DELETE FROM voiture WHERE Matriculation = '%s'", Matriculation);
if (mysql_query(conn, query)) {
    std::cerr << "Erreur de requête MySQL :" << mysql_error(conn) << std::endl;
    return;
}

std::cout << "La Voiture a été Supprimé Avec succès !!!" << std::endl;
mysql_close(conn);

break;
    /*****choix3*****/
}

case 3:

char newMat[50], newMarque[50], newEtat[50], MatAnc[50];
Consulter();

```

```

    std::cout << "\nEntrez la Matriculation de la Voiture à modifier: ";
    std::cin >> MatAnc;

    std::cout << "Entrez la nouvelle Matriculation (ou tapez '-' pour ne pas changer): ";
    std::cin >> newMat;
    std::cout << "Entrez la nouvelle Marque (ou tapez '-' pour ne pas changer): ";
    std::cin >> newMarque;
    std::cout << "Entrez le nouvel Etat (ou tapez '-' pour ne pas changer): ";
    std::cin >> newEtat;

    conn = mysql_init(NULL);

    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
        std::cerr << "Erreur de requête MySQL : " << mysql_error(conn) << std::endl;
        return;
    }

    if (strcmp(newMat, "-") != 0) {
        char query[256];
        sprintf(query, sizeof(query), "UPDATE voiture SET Matriculation = '%s' WHERE Matriculation = '%s'", newMat, MatAnc);
        if (mysql_query(conn, query)) {
            std::cerr << "Erreur de requête MySQL : " << mysql_error(conn) << std::endl;
            return;
        }
        strcpy(MatAnc, newMat);
    }

    if (strcmp(newMarque, "-") != 0) {
        char query[256];
        sprintf(query, sizeof(query), "UPDATE voiture SET Marque = '%s' WHERE Matriculation = '%s'", newMarque, MatAnc);
        if (mysql_query(conn, query)) {
            if (strcmp(newEtat, "-") != 0) {
                char query[256];
                sprintf(query, sizeof(query), "UPDATE voiture SET Etat = '%s' WHERE Matriculation = '%s'", newEtat, MatAnc);
                if (mysql_query(conn, query)) {
                    std::cerr << "Erreur de requête MySQL : " << mysql_error(conn) << std::endl;
                    return;
                }
            }
            std::cout << "\nLes modifications ont été effectuées avec succès !!! " << std::endl;
        }
        mysql_close(conn);
        break;
    }
}

void Gerant::Annuler() {
    std::cout << "Si vous pouvez Annuler, Saisir('Annuler') : ";
    std::string AN;
    std::cin >> AN;

    while (AN != "Annuler") {
        std::cout << "Votre Saisie Incorrecte, Réessayez : ";
        std::cin >> AN;
    }
}

exit(0);

```

## Main :

Dans Main, nous allons créer des objets des classes User et Gérant, et afficher leurs informations ainsi que les actions spécifiques du gérant.

Code source :

```
#include "Gerant.h"
#include <iostream>
#include <cstdlib>

int main() {
    char Nom[50], Password[50];

    Gerant gerant(Nom, Password);

    gerant.Authentification();

    .
    .
    .

    int choix;
    bool continuer = true;

} while (continuer) {
    std::cout << "\nMenu:\n";
    std::cout << "1-Consulter les voitures\n";
    std::cout << "2-Modifier une voiture\n";
    std::cout << "3-Qitter\n";
    std::cout << "Entrez votre choix: ";
    std::cin >> choix;

    switch (choix) {
    case 1:
        gerant.Consulter();
        break;
    case 2:
        gerant.ModifierVoiture();
        break;
    case 3:
        gerant.Annuler();
        break;
    default:
        std::cout << "Choix invalide, veuillez réessayer.\n";
        break;
    }
}

return 0;
```

## VI. Web :

### 1. Description :

Dans ce projet, Nous sommes développé la partie web qui se compose de plusieurs pages HTML et CSS. Le site web est conçu pour une entreprise de location de voitures et comprend les sections suivantes :

#### ➤ **Page d'Accueil :**

La page d'accueil est la première interface que les utilisateurs voient lorsqu'ils accèdent au site. Elle est structurée pour offrir une vue d'ensemble de tous les services disponibles.

Les sections principales de la page d'accueil incluent :

- **Accueil** : Présentation générale de l'entreprise.
- **Réservation** : Un bouton ou lien pour accéder à la page de réservation.
- **Conduite** : Informations sur Comment louer une Voiture.
- **Services** : Un exemple des voitures qui nous offrons.
- **À propos** : Les Valeurs plus.
- **Avis** : Informations sur l'équipe.

Toutes ces sections sont présentées sous forme de sous-pages intégrées dans la même page d'accueil, facilitant ainsi la navigation et offrant une expérience utilisateur fluide.

➤ Pages Spécifiques :

En plus de la page d'accueil, il y a des pages spécifiques dédiées à certaines fonctions :

- **Page de Réservation** : Permet aux utilisateurs de réserver une voiture en ligne. Cette page contient un formulaire où les utilisateurs peuvent sélectionner le véhicule souhaité, indiquer la durée de la location, et soumettre leur demande.
- **Page d'Inscription (Register)** : Un formulaire permettant aux nouveaux utilisateurs de créer un compte sur le site.
- **Page de Connexion (Login)** : Permet aux utilisateurs existants de se connecter à leur compte.

➤ Fonctionnalité de Génération de PDF :

Une fois connecté, l'utilisateur accède à une page de listes de voitures disponibles à la réservation. Après avoir choisi un véhicule et confirmé les détails de la location, un générateur de PDF est utilisé pour créer un contrat de location de voiture.

Ce contrat inclut :

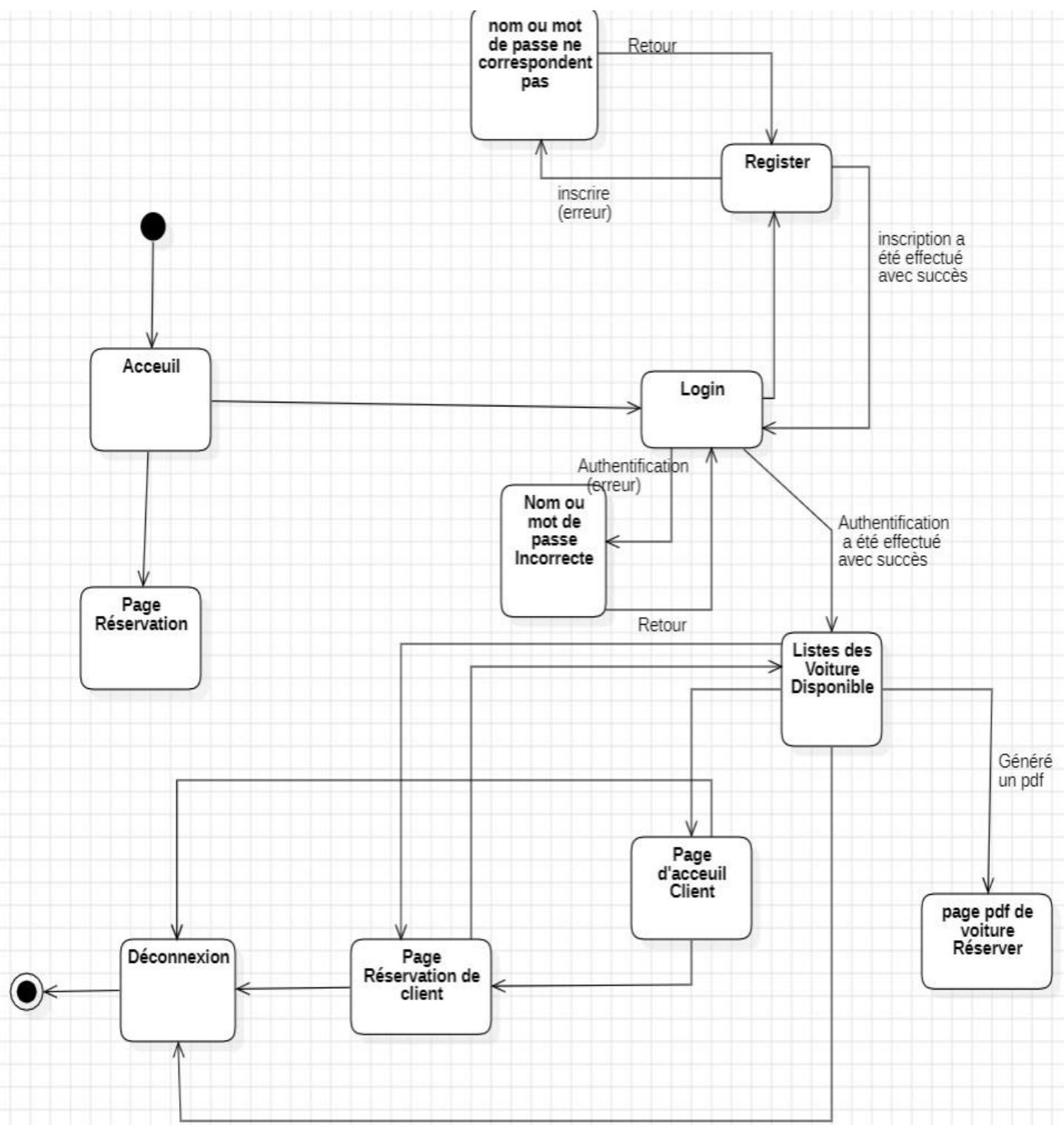
- **Nom du Client** : nom de l'utilisateur ayant réservé la voiture.
- **Marque du Véhicule** : Le modèle et la marque de la voiture choisie.
- **Nombre de Jours** : La durée de la location.
- **Prix Total** : Calculé en multipliant le prix par jour du véhicule par le nombre de jours de location.

Cette fonctionnalité assure que toutes les informations importantes sont bien documentées et facilement accessibles pour l'utilisateur et l'entreprise

➤ Technologies Utilisées :

- **HTML** : Pour la structure des pages web.
- **CSS** : Pour le style et la mise en page des pages web.
- **JavaScript** : Pour ajouter des fonctionnalités interactives et améliorer l'expérience utilisateur.
- **PHP** : Pour rendre le site dynamique et gérer les interactions avec la base de données. La connexion à la base de données permet de gérer les utilisateurs, les réservations et autres données dynamiques. Le système de login assure l'authentification des utilisateurs.
- **Bibliothèques PDF** : Pour la génération des contrats PDF.

## 2. Diagramme De Navigation :



### 3. Les pages Web :

#### a. Page d'accueil :

##### i. Accueil :

The screenshot shows the homepage of the Carsouk website. At the top, there is a navigation bar with links for Accueil, Réservation, Conduite, Services, A propos, and Equipe. There are also 'Register' and 'Login' buttons. Below the navigation bar, the main heading 'Louer Votre Voiture' is displayed in large, bold letters. Underneath the heading, a subtext reads 'Un service facile et efficace. Votre problème est notre préoccupation'. On the left side of the page, there are social media icons for Instagram and Facebook. A search form is overlaid on the image of a silver Volkswagen car's front end. The search form includes fields for 'Localisation' (with a placeholder 'search place'), 'date de départ' (with a date input field 'jj/mm/aaaa'), 'date de retour' (with a date input field 'jj/mm/aaaa'), and a 'Submit' button. In the bottom left corner of the image, there is a small red banner with the text 'localhost/projet/site.php#Home'.

ii. Conduite :

**comment louer une voiture?**

<b>1-Renseignez la date/lieu</b> Lorem ipsum dolor sit amet consectetur adipisicing elit. Error ullam accusantium nemo. Vel eos exercitationem temporibus maiores. Sapiente, molestias ipsa nemo itaque autem, vel blanditiis deleniti nisi asperiores, porro consequuntur.	<b>2-choisissez votre voiture</b> Lorem ipsum dolor sit amet consectetur adipisicing elit. Error ullam accusantium nemo. Vel eos exercitationem temporibus maiores. Sapiente, molestias ipsa nemo itaque autem, vel blanditiis deleniti nisi asperiores, porro consequuntur.	<b>3-validatez le paiement</b> Lorem ipsum dolor sit amet consectetur adipisicing elit. Error ullam accusantium nemo. Vel eos exercitationem temporibus maiores. Sapiente, molestias ipsa nemo itaque autem, vel blanditiis deleniti nisi asperiores, porro consequuntur.	<b>4-votre voiture est prêt</b> Lorem ipsum dolor sit amet consectetur adipisicing elit. Error ullam accusantium nemo. Vel eos exercitationem temporibus maiores. Sapiente, molestias ipsa nemo itaque autem, vel blanditiis deleniti nisi asperiores, porro consequuntur.

Nos Services  
**Explore Nos Voiture**  
**Et Selectioner Votre Choix**

iii. Services :

Nos Services

**Explore Nos Voiture  
Et Selectioner Votre Choix**



2020

Dacia-Docker  
400DH/J

Réserver Maintenant



2020

Dacia Duster  
300DH/J

Réserver Maintenant



2020

Dacia Logan  
400DH/J

Réserver Maintenant



2020

CLIO 4  
400DH/J

Réserver Maintenant



iv. A propos :

A propos de nous

## meilleure expérience client



\* **Tarif et Frais** : Nous offrons une clarté totale sur les coûts supplémentaires tels que les frais d'assurance, les frais de carburant et tout autre frais potentiel. Notre objectif est de fournir aux clients une compréhension complète des coûts pour éviter toute surprise désagréable lors de la facturation.

\* **Service client** : Notre équipe est disponible pour répondre à toutes vos questions, traiter les préoccupations éventuelles et vous assurer une expérience de location fluide et agréable.

\* **Politique de kilométrage** : Notre politique de kilométrage est conçue pour offrir une liberté maximale à nos clients. Nous proposons des options qui s'adaptent à différents besoins, que ce soit pour des trajets courts ou longs. Vous pouvez choisir la meilleure option qui correspond à votre itinéraire et à votre budget.

\* **Une promotion à nos clients** : nous offrons régulièrement des promotions exclusives. Ces offres spéciales peuvent inclure des réductions sur les tarifs de location, des mises à niveau de véhicules gratuites ou d'autres avantages

+ encore plus

v. Equipe :

Avis

## Equipe de Développement



G in @ Ⓜ

ELJEMLI Houssam-Eddine



G in @ Ⓜ

BOUSSIARI Yasser



G in @ Ⓜ

GUERROUJ Wael



G in @ Ⓜ

ARBIB Walid



[Home](#) [Reservation](#) [Conduite Services](#) [A propos](#) [Avis](#)

©2024 All right reserved

b. Page de Réservation :

CARSOUK

Accueil    Réservation    Conduite    Services    A propos    Equipe    Register    Login

Type de Voiture    Date de départ    Date de retour    Filtrer

-- Sélectionnez une marque --    jj/mm/aaaa    jj/mm/aaaa

**Dacia**



5 Personne    Annulation Gratuite  
Oui    150.00 MAD Par 1 Jour  
Manuelle    Réserver

**Renault**



5 Personne    Annulation Gratuite  
Oui    200.00 MAD Par 1 Jour  
Automatique    Réserver

c. Page de Login :



d. Page d'Enregistrer :



## e. La Liste des Voitures :



Accueil    Réervation    Conduite    Services    À propos    Equipe



La Liste des Voitures Disponibles

Houssam

Modifier le profil >

Déconnexion >

Voiture	Marque	Modèle	Matriculation	Nombre de personnes	Prix	
	Dacia	2020	1234A5678	5	150 MAD/J	<a href="#">Réserver</a>
	Renault	2021	2345B6789	5	200 MAD/J	<a href="#">Réserver</a>
	Peugeot	2019	3456C7890	7	180 MAD/J	<a href="#">Réserver</a>
	Mercedes	2022	4567D8901	4	300 MAD/J	<a href="#">Réserver</a>
	BMW	2023	5678E9012	4	350 MAD/J	<a href="#">Réserver</a>
	Mercedes	2020	aae1232	5	1200 MAD/J	<a href="#">Réserver</a>

f. Page du Contrat :

The screenshot shows a web browser window with the URL "localhost / doc" in the address bar. The page content is as follows:

**CarSouk**

Bienvenue a CarSouk  
Votre site pour Reserver Votre Voiture

---

Nom de client	Marque	Modele	Prix	Nombre de Jours
Houssam	Dacia	2020	600	4

Votre Reservation est Effectuee Avec Succes

\* Veuillez apporter cette feuille a la location du voiture

## VII. Conclusion :

Ce projet de site web de location de voitures a été développé pour répondre à la demande croissante de solutions de location de véhicules en ligne. En intégrant des fonctionnalités modernes et une interface utilisateur intuitive, nous avons réussi à simplifier et améliorer l'expérience de location pour les utilisateurs.

Le travail en équipe a joué un rôle crucial dans la réalisation de ce projet. La collaboration a permis de diviser les tâches efficacement et de partager des idées innovantes, ce qui a conduit à une solution de haute qualité. Chaque membre de l'équipe a pu apporter ses compétences spécifiques, ce qui a enrichi le projet et accéléré son avancement.

En résumé, ce projet a non seulement fourni une solution pratique et efficace pour les utilisateurs, mais il a également permis à l'équipe de renforcer ses compétences techniques et sa capacité à travailler ensemble. Avec une base solide en place, nous sommes bien positionnés pour continuer à améliorer le site et explorer de nouvelles opportunités de croissance.