**TROISIEME CHAPITRE : METHODOLOGIE**

### ****3.1. Introduction****

Ce hapitre présente la méthodologie suivie pour analyser, concevoir et structurer le système informatisé de gestion du garage GARAGE PEINT BUTEMBO. Il expose les différentes étapes ayant permis de passer des besoins exprimés par les utilisateurs à une solution technique cohérente et fonctionnelle. Dans un premier temps, l’attention est portée sur **l’expression des besoins**, à travers l’élaboration du **cahier des charges,** l**’identification des acteurs et de leurs rôles,** ainsi que la **modélisation des contextes**. Cette phase permet de bien cerner les attentes des utilisateurs et de poser les bases fonctionnelles du système à développer. Ensuite, **dynamique du système,** les **cas d’utilisation**, en illustrant les interactions et comportements à l’aide de **diagrammes de séquence, diagrammes d’activités,** **diagrammes d’état-transition** et **diagrammes de package.** Cette modélisation permet de comprendre comment le système réagit face aux actions des différents acteurs. Après cela, la **modélisation statique,** pour comprendre la conception **du modèle conceptuel des données** avec le **diagramme de classes**, l’élaboration du **schéma relationnel,** les **diagrammes d’objets**, ainsi que la **conception des interfaces graphiques** et le **diagramme de déploiement.** Cette dernière étape permet de structurer techniquement la solution en vue de sa mise en œuvre.

**3.2. Méthodes et techniques**

**3.2.1. Méthodes**

Pour ce travail, nous avons utilisé le langage de modélisation **UML (Unified Modeling Language)**, en adoptant la méthode **UP (Unified Process)**, qui décrit un processus de développement logiciel itératif et incrémental. Ce processus comprend plusieurs phases essentielles : l'expression des besoins, l'analyse, la conception, l'implémentation, les tests et le déploiement de la solution (Jacobson et al., 1992).

Par ailleurs, nous avons également opté pour l’approche de prototypage. Cette approche consiste à développer une version préliminaire de l’application afin d’obtenir des retours d’utilisateurs avant le développement final. Dans ce cadre, un prototype fonctionnel a été conçu à l’aide de Visual Studio, en utilisant le langage C# et Windows Forms. Ce prototype comprend notamment une interface de gestion des pièces de rechange, l’enregistrement des véhicules, la consultation des factures et le suivi des réparations. Par exemple, l’interface de saisie des véhicules permet de tester la saisie de l’immatriculation, du nom du client, de la marque et des problèmes signalés, ce qui a permis d'ajuster les champs et les contrôles nécessaires à la phase de conception finale.

L’application de cette méthode repose sur un plan d'action structuré en plusieurs étapes, permettant d'atteindre les objectifs définis. Ces étapes sont généralement accompagnées de diagrammes UML pour faciliter la compréhension, la communication avec les parties prenantes et la validation des processus.

**III.2.2. Techniques et outils**

#### **a) Techniques**

Pour la collecte et l’analyse des besoins, nous avons eu recours aux techniques suivantes :

* **Observation directe** : Cette technique nous a permis d’analyser le déroulement des activités au sein du garage, notamment la réception des véhicules, la gestion des pièces de rechange et la facturation. Elle a été utile pour identifier les lacunes du système actuel et les processus à informatiser.
* **Interview (ou entretien)** : Nous avons mené des entretiens semi-structurés avec les responsables du garage (réceptionniste, mécaniciens, caissier, etc.) afin de mieux comprendre leurs besoins, leurs attentes vis-à-vis du futur système et les difficultés rencontrées avec la gestion manuelle. Cette technique a permis de recueillir des informations qualitatives précieuses pour orienter la conception du système.

#### **b) Outils**

Plusieurs outils ont été utilisés tout au long du projet, chacun jouant un rôle spécifique dans la réalisation du système :

* **Microsoft Visio et Draw.io** : Ces outils de modélisation graphique ont servi à concevoir les différents diagrammes UML (cas d'utilisation, classes, séquences, etc.), facilitant la visualisation et la documentation du système.
* **Visual Studio (avec C#)** : Cet environnement de développement intégré (IDE) a été utilisé pour coder l’application. Le langage **C#** a été choisi pour sa robustesse et sa compatibilité avec les interfaces Windows.
* **SQL Server** : Ce système de gestion de base de données relationnelle (SGBDR) a été utilisé pour la création, la gestion et l'interrogation des données de l’application. Il assure la sécurité, la cohérence et la persistance des informations du garage.
* **Microsoft Word** : Utilisé pour la rédaction du présent rapport ainsi que pour la production de la documentation technique liée à l’analyse, à la conception et à la mise en œuvre du système.

**III.3. MODELISATION DE LA SOLUTION**

**III.3.1. EXPRESSION DES BESOINS**

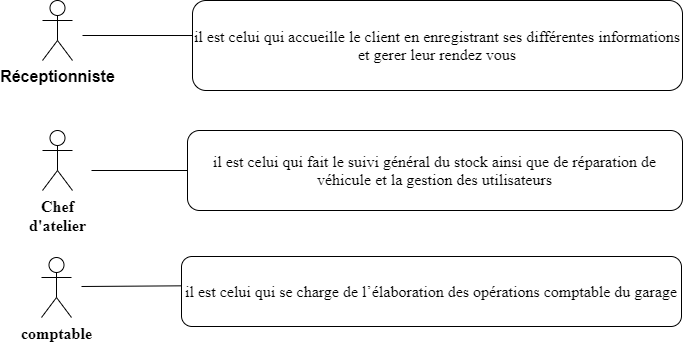
**III.3.1.1. Cahier des charges**

Le cahier des charges décrit les besoins fonctionnels et non fonctionnels exprimés par les utilisateurs. Il sert de référence tout au long du développement du système pour s’assurer que les attentes initiales sont bien prises en compte (Dennis, Wixom, & Roth, 2021).

|  |
| --- |
| Cahier des charges Système informatisé de gestion de garage  **Besoins fonctionnels**   * + Ajouter, modifier, supprimer et consulter les pièces disponibles.   + Gérer les stocks avec des alertes sur les niveaux critiques.   + Générer des rapports d'inventaire et de mouvements de stock.   + Enregistrer les paiements des clients.   + Gestion de rendez-vous   + Notifier aux clients   + Suivi de paiement   + Générer et imprimer des factures.   + Création, modification et suppression de comptes utilisateurs.   + Affichage des statistiques des ventes et de l'état des stocks.   + Gingerer des rapports périodiques   **Besoins opérationnels (ou non fonctionnels)**   * **Performance :** le système permettra de traiter rapidement les opérations (recherche, saisie, facturation…), afin de réduire le temps d’attente des clients et d’améliorer la fluidité du service. * **Sécurité** : L’accès au système sera sécurisé par une **authentification avec mot de passe**. Chaque utilisateur disposera d’un **rôle spécifique** avec des **droits d’accès limités**. Des **sauvegardes automatiques** seront mises en place pour éviter toute perte de données, et les actions sensibles seront **journalisées** pour assurer la traçabilité. * **Ergonomie :** Interface conviviale et intuitive adaptée aux utilisateurs non experts. * **Accessibité** : l’application sera accessible à Travers un ordinateur pc ou mac et sera installée localement sur le système windows, lunix, c’est-à-dire l’aplication fonctionnera sans connexion internet mais à travers le réseau local   **Choix techniques**   * Langage de programmation : C# * Environnement de développement : Visual Studio * Base de données : SGBD Mysql * Architecture : Client-serveur 2/3 * langage de mod~~e~~lisation UML * Système d'exploitation : Windows * Interface graphique : Application desktop en C# avec Windows Forms ou WPF |

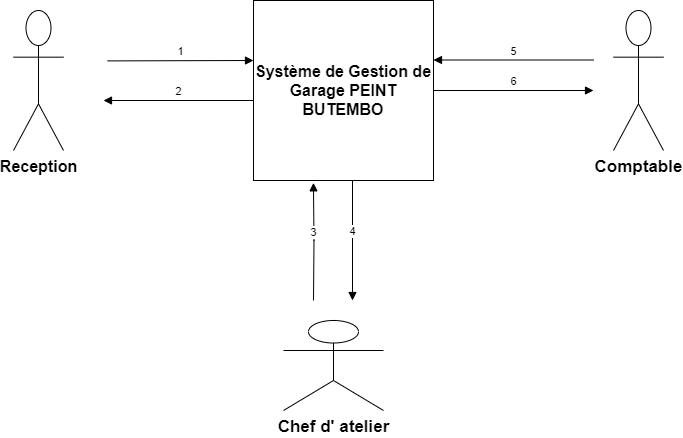
**III.3.1.2. Identification des acteurs et leurs rôles**

Cette étape consiste à identifier les différents utilisateurs du système (acteurs) ainsi que leurs rôles respectifs dans l'interaction avec le système. Parmi les acteurs identifiés, on retrouve : Chef d’atelier, le Réceptionniste, le Comptable. Chacun joue un rôle clé dans la gestion quotidienne des activités du garage GARAGE **PEINT BUTEMBO** (Dennis, Wixom, & Roth, 2021).



**III.3.1.3. Modélisation des contextes**

Le **diagramme de contexte** est un outil de modélisation utilisé en phase d’analyse pour représenter le système à un niveau global, sans entrer dans les détails internes. Il illustre les interactions principales entre le système étudié et les acteurs externes (utilisateurs, autres systèmes, entités organisationnelles), en montrant les flux d’information échangés. Selon Satzinger, Jackson et Burd (2020), un diagramme de contexte sert à « visualiser la frontière entre le système et son environnement, tout en identifiant les entrées et sorties d'informations essentielles ».



**Légende du diagramme de contexte**

1. Le réceptionniste envoie une requête pour enregistrer un véhicule à réparer ou pour planifier un rendez-vous
2. Le système renvoie un message personnalisé de confirmation : « Véhicule enregistré avec succès », « Rendez-vous planifié », ou un message d’erreur comme « Données incomplètes » ou « Véhicule déjà enregistré ».
3. Le chef d’atelier soumet des informations pour : gérer les utilisateurs (ajout, modification), gérer le stock (entrée, sortie de pièces), ou suivre l’état d’une réparation (en cours, terminé).
4. Le système renvoie une notification selon l’action réalisée : « Utilisateur ajouté », « Stock mis à jour », « Réparation enregistrée », ou un message d’erreur en cas de problème (ex. : « Quantité insuffisante »).
5. Le comptable soumet une requête de paiement en sélectionnant un dossier client et en saisissant le montant reçu et le mode de paiement.
6. Le système génère la facture ou le reçu correspondant et renvoie un message comme : « Paiement enregistré – Reçu généré avec succès ».

**III.2. MODELISATION DYNAMIQUE DU SYSTEME**

L’analyse permet une formalisation du système à développer en réponse à l’expression des besoins formulés par les utilisateurs. L’analyse se concrétise par l’élaboration de tous les diagrammes donnant une représentation du système tant statique (diagramme de classe principalement), que dynamique (diagramme des cas d’utilisation, de séquence, d’activité, d’état-transition…) (Dennis, Wixom, & Roth, 2021).

**III.2.1. Identification des cas d’utilisation**

L’identification des cas d’utilisation constitue une étape cruciale dans l’analyse fonctionnelle du système. Elle consiste à recenser l’ensemble des fonctionnalités que le système doit offrir aux différents acteurs, en tenant compte de leurs besoins spécifiques et de leurs interactions avec le système. Ces cas d’utilisation permettent de délimiter les périmètres fonctionnels du système, de définir les services à fournir et de structurer la modélisation des exigences.

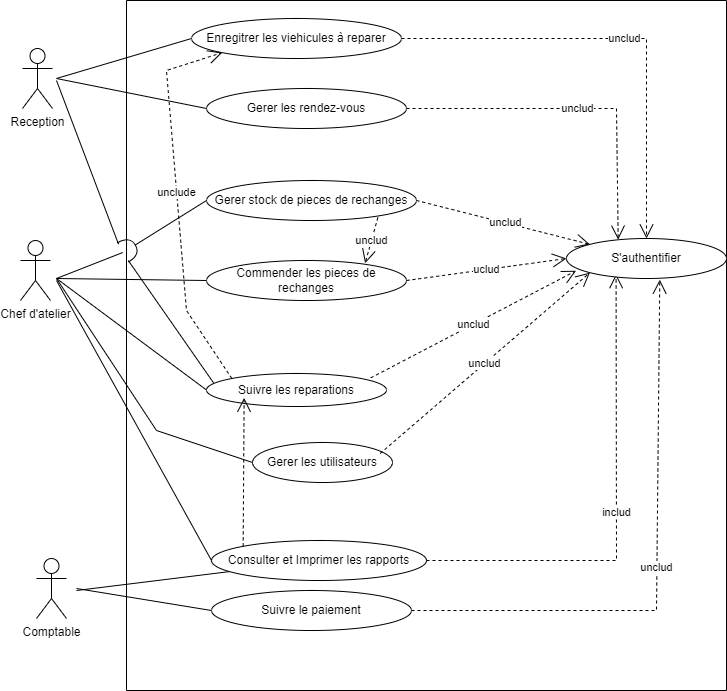
Les cas d’utilisation identifiés pour ce projet sont :

1. S’authentifier
2. Enregistrer les véhicules à réparer
3. Gérer les rendez-vous
4. Gérer le stock de pièces de rechange
5. Commander les pièces de rechange
6. Suivre les réparations
7. Gérer les utilisateurs
8. Imprimer les rapports
9. Suivre le paiement

Ces cas d’utilisation serviront de base pour la réalisation des diagrammes de cas d’utilisation, des fiches de description textuelle, ainsi que des diagrammes de séquence et d’activités correspondants.

**III.2.2. Diagramme de cas d’utilisation**

Le **diagramme de cas d’utilisation** (ou *use case diagram*) est un diagramme UML (Unified Modeling Language) utilisé pour modéliser les interactions entre les utilisateurs (ou acteurs) et le système. Il permet de représenter graphiquement les différents services ou fonctionnalités que le système offre aux acteurs, sous forme de cas d’utilisation. Chaque cas d’utilisation décrit un scénario spécifique dans lequel un acteur interagit avec le système pour atteindre un objectif précis. D’après Sommerville (2016), ce type de diagramme est essentiel pour capturer les exigences fonctionnelles d’un système logiciel en fournissant une vue claire des interactions attendues.



**Tu mets une petite référence ici de l’image et si c’est une figure tu mets aussi la refenrence et idem pour le tableau. Ça te permets de n’est plus y revenir lorsque tu feras les partie là preliminaire de liste de figure et de tableau ainsi aue de figures, ça se fais automatiquement ; je penses aue tu sais comment proceder car si tu le fais par ecrit ça ne viendra pas lors de la gerations de ces liste, c’est aui ne sera paq bon**

**III.3.2.5. Description textuelle des cas d'utilisation**

La **description textuelle des cas d'utilisation** est une méthode structurée permettant de détailler.

les interactions entre un acteur et le système dans un contexte fonctionnel spécifique. Elle complète le diagramme de cas d’utilisation en précisant, sous forme de fiche descriptive, les objectifs, les conditions, les scénarios et les résultats attendus d’un cas donné. Selon Dennis, Wixom et Roth (2021), cette description facilite la compréhension des exigences fonctionnelles par l’ensemble des parties prenantes (analystes, développeurs, testeurs, utilisateurs finaux) et sert de guide précis pour la conception et la mise en œuvre du système

**1. Description textuelle du cas s’authentifier**

|  |
| --- |
| **Sommaire d’identification**  **Titre : S’authentifier**  **Résume : ce cas permet aux utilisateur de se connecter au système via un pwd et un identifiant**  **Acteurs principaux :** chef d’atelier**, Réceptionniste, Comptable.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description de Senarios nominal**  **Preconditions** :  Lancement de l’application et avoirs les identifiants enregistrer dans la base de données  **Senario nominal**:   * 1. Le système affiche une interface de connexion.   2. L’utilisateur saisit son nom et son mot de passe.   3. Le système vérifie les informations d’authentification dans la base de données.   4. L’utilisateur est redirigé vers sa page selon son rôle.   **Enchainement alternatif** :  **A1**: Mot de passe ou nom incorrecte ;  **Le système signale** que **le nom ou mot de passe ne sont pas reconnu**  **Le scénario reprend au point 3 du Nominal**  **Post conditions (resultant attendu)** :   * + L’utilisateur accède à son espace avec les droits qui lui sont attribués. |

2. **Enregistrer les véhicules à réparer**

|  |
| --- |
| **Sommaire d’identification**  **Titre : Enregistrer les véhicules à réparer**  **Résume : ce cas permetd’enregistrer un vehicule**  **Acteurs : Réceptionniste.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description de Senarios nominal**   * **Preconditions** :   + L’utilisatgeur est connecté comme Receptionniste * **Senario** nominal:  1. L’utilisateur choisit du menu gestion véhicule 2. Système affiche la page de gestion des véhicules 3. Il saisit les informations générales du véhicule à réparer : numéro de plaque, Marque, modèle, état général, date d’entrée… 4. Le système vérifie l’unicité de numéro de la plaque 5. Le système renvoie un message de confirmation de l’enregistrement  * **Enchainement alternatif**   **A1**: doublon de l’immatriculation  **Le système communique que l’immatriculation saisie est déjà dans le système**  **Retour au point 3 du Nominal**   * **Post conditions (resultant attendu)** :   + Les informations générales du véhicule à réparer sont sauvegardées. |

3. **Cas d’utilisation : Gérer les rendez-vous**

|  |
| --- |
| **Sommaire d’identification**  **Titre : S’authentifier**  **Résume : ce cas permet de planifier les Rendez-vous**  **Acteur : Réceptionniste.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des scénarios**  **Preconditions** :   * + L’utilisateur est authentifié comme réceptionniste.   **Scénario nominal:**   1. L'utilisateur le menu de gestion des rendez-vous et des délais. 2. Système affiche les fonctionnalités de création 3. L'utilisateur sélectionne l'option de créer un nouveau rendez-vous 4. Le système Affiche le formulaire correspondant 5. L'utilisateur saisit les informations nécessaires telles que la date, l'heure, la description et les participants pour le rendez-vous, ou la date limite et la description pour le délai puis enregistre les informations et reçoit une confirmation 6. Le système Vérifie l’unicité de l’heure  * **Enchainment alternatif** :   **A1**: En cas de conflit de calendrier  L**e système** notifie l'utilisateur et lui propose de modifier la date ou l'horaire du rendez-vous.  **Le scénario reprend au point 5 du Nominal**   * **Post conditions (resultant attendu)** :   + Le rendez-vous est planifié. |

4**. Cas d’utilisation : Gérer paiement**

|  |
| --- |
| **Sommaire d’identification**  **Titre : Gerer paiement**  **Résume : ce cas permet d’enregistrer les infos compteble**  **Acteur : Comptable.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des senarios**  **Preconditions**:   * L’utilisateur est connecté comme comptable. * Les informations du véhicule et des pièces utilisées ont été enregistrées par le réceptionniste.   **Description des scenarios**:   * + Il sélectionne l’immatriculation du véhicule ou du client concerné.   + Le système affiche la facture correspondante (pièces utilisées, services, etc.).   + Le comptable saisit le montant reçu.   + Le système vérifie si le montant saisi correspond au montant de la facture   + Le système génère la facture paiement enregistre.   + Le comptable imprime ou envoie le reçu au client.   **Enchainement alternatif** :  **A1 : Aucune immatriculation sélectionnée**  **A1. Le système demande à l’utilisateur de sélectionner une immatriculation**  **Retour au point 1 du nominal.**  **A2**: Le montant ne correspond pas au montant de facture  **4. Le système** notifie l'utilisateur le montant restant  **Retour au point 3 du Nominal**  **Postconditions** :   * + Le paiement est enregistré dans le système.   + Un reçu est imprimé ou envoyé.   + Les données sont disponibles pour les rapports financiers. |

**5. Description textuelle Cas d’utilisation : Gérer les utilisateurs**

|  |
| --- |
| **Sommaire d’identification**  **Titre : Gérer les utilisateurs**  **Résume : Enregistrer, supprimer, modifier, attribuer les taches aux utilisateurs**  **Acteur : Chef d’atelier.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des enchainements/ Séquencements**  **Préconditions** :   * + L’utilisateur est connecté au système chef d’atelier.   **Description des scenarios**:  1. Choisir le module **Gestion des utilisateurs**.   * + Le système affiche la page des utilisateurs   + Il choisit une action :     - Ajouter un nouvel utilisateur     - Modifier les informations d’un utilisateur existant     - Supprimer un utilisateur     - Changer le rôle d’un utilisateur (réceptionniste, caissier, etc.)   + Afficher la page correspond à l’action choisie   + Il saisit ou modifie les données nécessaires (nom utilisateur, mot de passe, rôle, etc.) et il valide les modifications.   + Le système vérifie l’unicité des informations saisies   + Le système enregistre les changements et affiche un message de confirmation.   **Scénarios alternatifs** :  A1. Information saisi existe da la base de donnée  Le système indique que le nom utilisateur est déjà utilisé  Le scenario reprend au point 5 de scenario nominale  **Post conditions (résultat attendu)**:  Mise à jour d’un utilisateur effectué avec succès |

**6. Description textuelle cas d’utilisation Gérer les pièces de rechange**

|  |
| --- |
| **Sommaire d’identification**  **Titre : Gérer les pièces de rechange**  **Résume : Enregistrer, supprimer, modifier, une pieces de rechannge**  **Acteur : Chef d’atelier.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des scénarios**   * **Preconditions** :   + Le chef d’atelier est connecté au système.   **Description du scénario principal** :  1. Choisir le module **Gestion des pièces**.   * + Le système affiche la page gestion des pièces   + Il sélectionne l’action à effectuer :     - Ajouter une nouvelle pièce (désignation, référence, prix, quantité…)     - Modifier les informations d’une pièce existante     - Supprimer une pièce obsolète ou indisponible   + Le système Affiche la page correspond à l'action choisie   + Il saisit ou modifie les données nécessaires et valide les modifications.   + Le système vérifie si la saisie a été complet   + Opération réussie.   **Scénarios alternatifs**:  A1. Saisie incomplète ou erronée  Le système indique que les informations saisies sont incomplètes  Le scenario reprend au point 5 de scenario nominal  **Postconditions (résultat attendu)** :   * + Le stock est mis à jour en temps réel. |

**7. Description textuelle du Cas d’utilisation : Consulter et imprimer un rapport**

|  |
| --- |
| **Sommaire d’identification**  **Titre : Gérer les pièces de rechange**  **Résume : ce cas permet de Consulter et imprimer un rapport**  **Acteur : Chef d’atelier, réceptionniste, comptable.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des scénarios**  **Preconditions**:   * + L’utilisateur est connecté (authentifié).   + Le système contient des données à afficher sous forme de rapport (paiements, réparations, historique, etc.). * **Description du scenario**:   + L’utilisateur choisie le module de **rapport/statistiques**.   + Le système affiche les rapports   + Il sélectionne le type de rapport à consulter :     - Rapport de paiements     - Rapport des véhicules réparés     - Rapport des pieces utilisées     - Rapport d’activités par utilisateur   + Il peut filtrer par période (date, mois, client, utilisateur, etc.).   + Le système génère un aperçu du rapport.   + Le système affiche le resultat   + L’utilisateur peut choisir d’imprimer ou d’exporter (PDF, Excel).   + Le système fournit l’impression.   **Scénarios alternatifs** :  **A1**. Aucun résultat ne correspond au filtre  **Le système affiche « Aucun résultat trouvé »**  **Le scenario reprend au niveau 3 de scenarios nominal**  **Postconditions (résultat attendu)** :   * + Le rapport est affiché à l’écran et/ou imprimé avec succès.   + Les informations consultées restent accessibles pour d’autres usages |

### ****8. Description textuelle du Cas d’utilisation : Suivre les réparations****

|  |
| --- |
| **Sommaire d’identification**  **Titre : Suivre les réparations**  **Résume : ce cas permet de Consulter les réparations**  **Acteur : Chef d’atelier, réceptionniste.**  **Date de création :15/04/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des scenarios**  **Préconditions** :  L’utilisateur est connecté (authentifié).  **Description du scénario principal** :   1. L’utilisateur accède au module de suivi. 2. Le système affiche la liste des véhicules en réparation. 3. Il sélectionne un véhicule. 4. Le système affiche les infos de réparation liés à ce vehicule 5. Il met à jour l’état (diagnostic, en cours, terminé). 6. Le système enregistre les changements. 7. Le système affiche un message de confirmation.   **Scénarios alternatifs** :  A1. Aucune mise à jour sélectionnée  Le système affiche un message « veiller sélectionner un mis en jour »  Retour au point 5 du scénario nominal  **Post conditions**  L’état de la réparation est actualisé |

**9**. **Description textuelle** **Cas d’utilisation : commander les pièces de rechange**

|  |
| --- |
| **Sommaire d’identification**  **Titre : Suivre les réparations**  **Résume : ce cas permet de** Réapprovisionner les pièces manquantes  **Acteur : Chef d’atelier, réceptionniste.**  **Date de création : 12/05/2025**  **Date de ms à jour : 14/08/2025**  **Version : 1.0**  **Responsable : Raymond**  **Description des scenarios**  **Preconditions**:   * + L’utilisateur est authentifié comme chef d’atelier   + La base de données du stock est à jour (les pièces manquantes sont connues)   **Scenarios nominal**:   1. L’administrateur Ouvre le module de commande 2. Affiche l’interface de commande et liste automatiquement les pièces manquantes 3. Il sélectionne les pièces à commander. 4. Génère un bon de commande pré-rempli avec les pièces sélectionnées 5. Le système génère un fichier PDF du bon de commande et l’enregistre localement 6. Le système affiche : “Commande enregistrée et bon de commande généré avec succès.”   **Scénarios alternatifs** :  A1. Aucun produit sélectionné  3. Le système affiche : “Aucune pièce à commander actuellement.”  Le scénario nominal commence au point 3  A2. Problème de générer un pdf  4. Le système affiche un message : “Erreur lors de la création du bon de commande. Veuillez réessayer.”  **Post conditions** :   * Un bon de commande est généré automatiquement en **PDF** et peut être envoyé chez le fournisseur * La commande est enregistrée dans la base de données locale. |

**III.2.3. Diagramme de séquence**

Le diagramme de séquence est un outil essentiel de la modélisation UML qui permet de représenter la **chronologie des interactions** entre les objets du système pour un **scénario fonctionnel spécifique**. Il illustre l’enchaînement des **messages échangés** entre les acteurs et le système, en mettant l’accent sur la dimension **temporelle des opérations**. Chaque message est représenté par une flèche, tandis que les objets sont alignés horizontalement, et leur **durée de vie verticale** permet de visualiser le déroulement logique d’une fonctionnalité. Ce type de diagramme offre une **vision dynamique du système**, facilitant la compréhension et la conception des processus métier (Arlow & Neustadt, 2013 ; Larman, 2004).

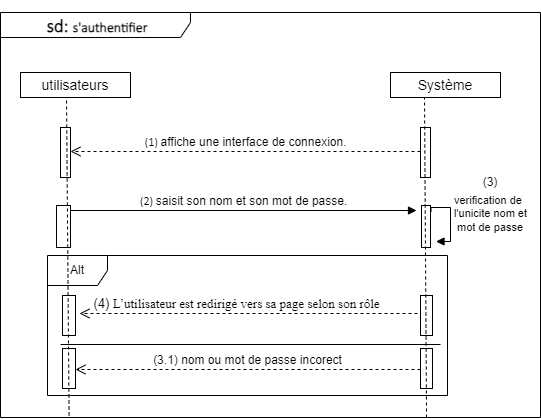
Le diagramme de séquence joue un rôle crucial pour :

* clarifier la **logique d’exécution** des opérations ;
* identifier les **responsabilités des classes** ou composants impliqués ;
* faciliter la **conception technique** en amont de l’implémentation ;
* soutenir les étapes de **tests fonctionnels**.

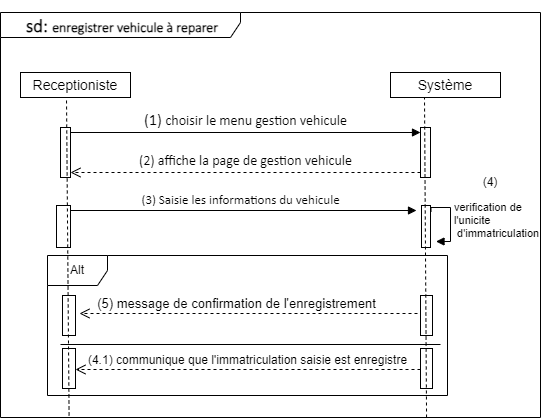
Dans le cadre du projet **GARAGE PEINT BUTEMBO**, ces diagrammes de séquence sont dérivés des **fiches de description textuelle des cas d’utilisation**, et représentés sous forme de **frames** (cadres) annotés en haut à gauche par sd (*sequence diagram*) suivis du nom de la fonctionnalité concernée. Voici les principaux diagrammes de séquence prévus :

* 1. **S’authentifier**

Ce diagramme de séquence illustre les interactions entre l’utilisateur et le système lors du processus de connexion. Il montre comment le système vérifie les identifiants et détermine si l’utilisateur peut accéder aux fonctionnalités selon son rôle.

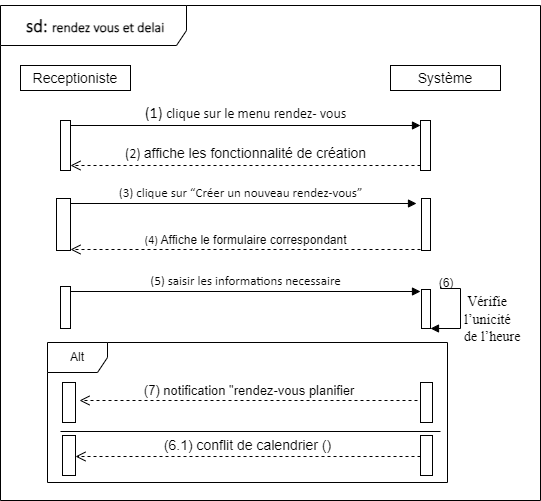


### 2. ****Enregistrer les véhicules à réparer**** Ce diagramme montre les échanges entre le réceptionniste et le système lorsqu’un véhicule est enregistré. Il détaille la saisie des informations du véhicule, la vérification des doublons et l’enregistrement des données.



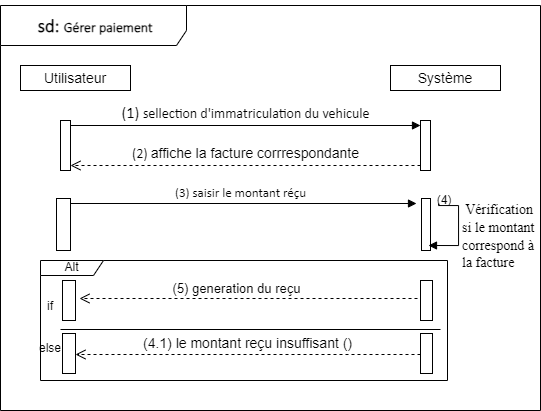
### 3. ****Gérer les rendez-vous****

Ce diagramme illustre comment le réceptionniste interagit avec le système pour planifier ou modifier un rendez-vous. Il inclut la sélection d’une date, la saisie des détails, l’enregistrement et la confirmation.



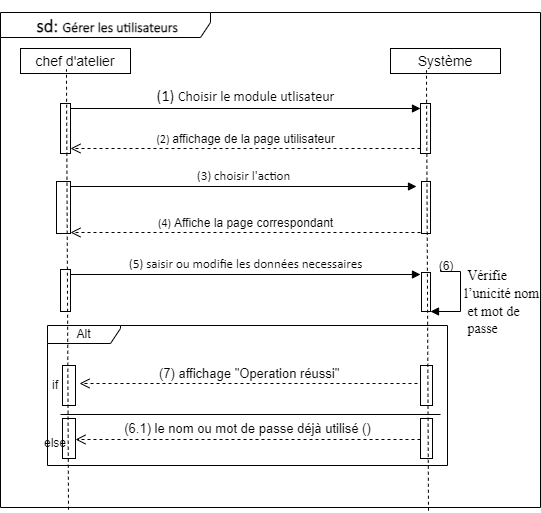
### 4. ****Suivre le paiement****

Ce diagramme décrit les interactions entre le comptable et le système lors de la validation d’un paiement : affichage de la facture, saisie du montant, vérification et génération du reçu.



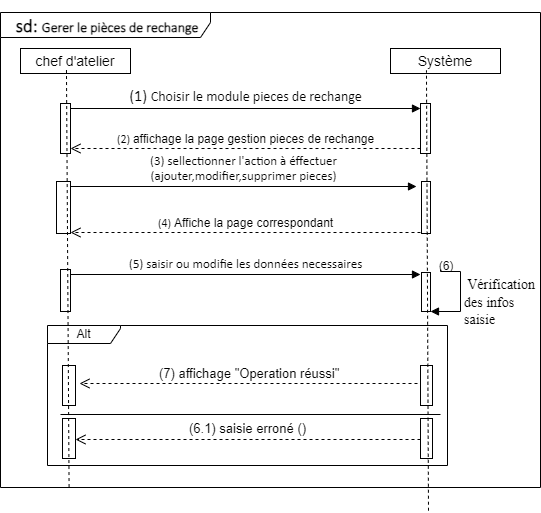
### 5. ****Gérer les utilisateurs****

Ce diagramme détaille comment l’administrateur ajoute, modifie ou supprime un utilisateur, et comment le système enregistre ces actions.



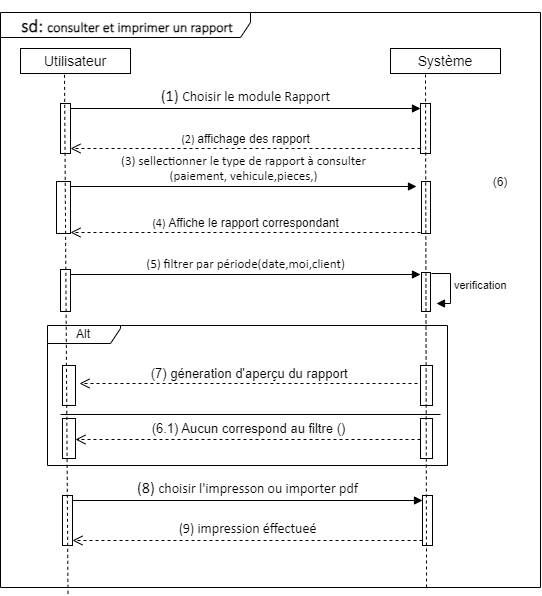
### 6. ****Gérer le stock de pièces de rechange****

Ce diagramme présente les échanges entre le chef d’atelier et le système pour consulter, ajouter, modifier ou supprimer une pièce dans le stock.



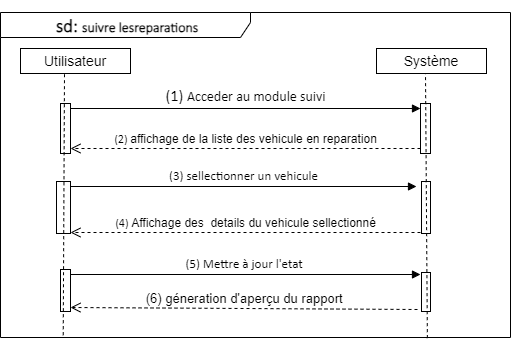
### 7. Consulter et ****Imprimer les rapports****

Ce diagramme montre comment un utilisateur génère et imprime un rapport, depuis le choix du type de rapport jusqu’à l’envoi à l’imprimante.



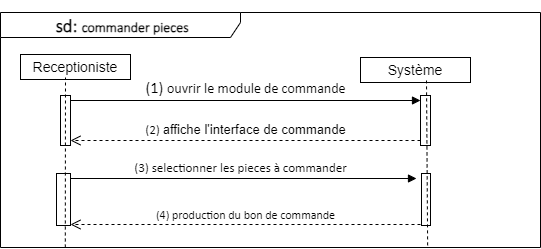
### 8. ****Suivre les réparations****

Ce diagramme décrit les échanges entre l’utilisateur et le système pour accéder au suivi des réparations : sélection du véhicule, affichage des détails et mise à jour de l’état.



### 9. ****Commander les pièces de rechange****

Ce diagramme retrace les interactions nécessaires pour passer une commande de pièces. Le chef d’atelier sélectionne les pièces manquantes, indique les quantités et valide la commande.



Ces diagrammes assurent une continuité cohérente entre l’analyse fonctionnelle (cas d’utilisation) et la conception orientée objet, contribuant ainsi à une meilleure anticipation des comportements du système dans chaque situation métier.

### ****III.2.4. Diagramme d’activités****

Selon Dennis, Wixom et Tegarden (2021), le diagramme d’activités permet de capturer le **comportement fonctionnel** d’un processus métier ou d’un cas d’utilisation, et de faciliter la **compréhension des procédures internes**. Il est souvent utilisé pour analyser les workflows, spécifier des algorithmes ou identifier les points de contrôle et de décision dans un système.

Dans le contexte du projet **GARAGE PEINT BUTEMBO**, les diagrammes d’activités ont pour but de modéliser les différents processus métiers tels que la **gestion des paiements**, la **prise en charge des véhicules**, la **gestion du stock des pièces de rechange**, ou encore l’**authentification des utilisateurs**. Chacun de ces processus sera décrit à l’aide d’un diagramme d’activités afin de:

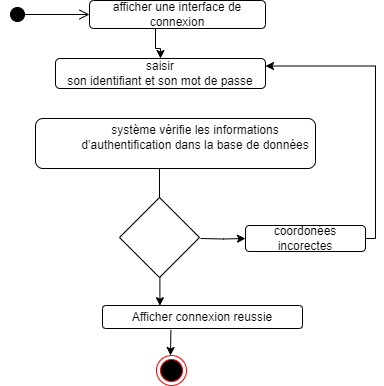
* Comprendre le **déroulement global** d’une fonctionnalité ;
* Identifier les **étapes critiques** et les **points de contrôle** ;
* Repérer les **flux alternatifs** ou les **chemins conditionnels** ;
* Favoriser une **implémentation structurée** et alignée sur les exigences métier.

### Activités modélisées

Les principaux diagrammes d’activités développés dans le cadre de ce projet couvrent un ensemble de processus essentiels à la gestion des opérations du GARAGE PEINT BUTEMBO. Ils modélisent notamment les activités telles que **l’authentification des utilisateurs, la gestion des paiements, l’enregistrement des informations et des pannes d’un véhicule,** ainsi que le **listing des pièces à utiliser pour une réparation**. D’autres activités clés incluent la **gestion des utilisateurs** (ajout, modification, suppression), la **consultation et l’impression des rapports**, et enfin la **gestion du stock des pièces de rechange**. Chacune de ces activités est représentée par un diagramme spécifique afin de mettre en lumière les étapes, les décisions et les flux alternatifs qui les composent, facilitant ainsi leur compréhension et leur implémentation dans le système informatisé.

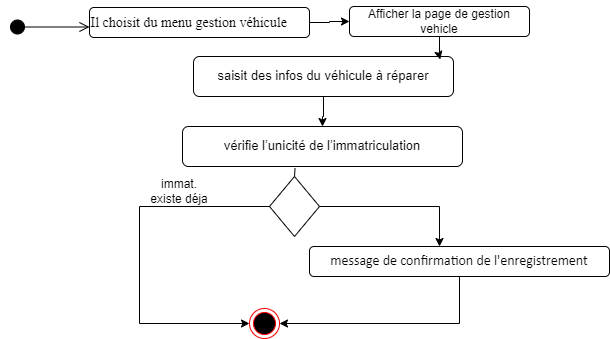
**S’authentifier**

Cette activité permet aux utilisateurs autorisés d'accéder au système en saisissant leurs identifiants (nom d'utilisateur et mot de passe). Le système vérifie les informations et accorde l'accès uniquement si elles sont valides. Cela garantit la sécurité et la confidentialité des données du garage.



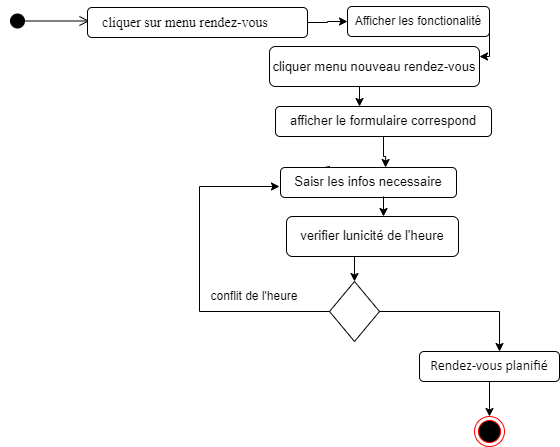
**Enregistrer les véhicules à réparer**

Lorsqu’un client amène un véhicule, le réceptionniste saisit les informations du véhicule (marque, plaque, propriétaire, etc.) ainsi que les pannes ou anomalies signalées. Ces données sont indispensables pour le suivi des réparations et l’élaboration des devis.



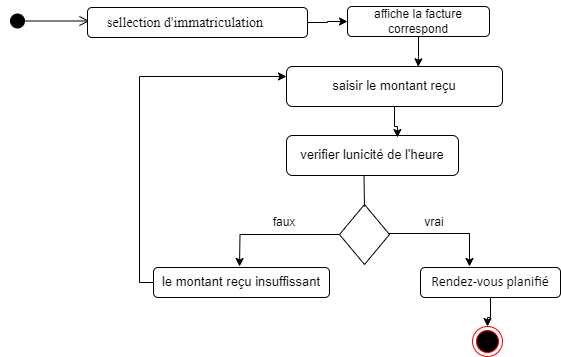
### ****Gérer les rendez-vous****

Ce diagramme montre le déroulement logique de la gestion des rendez-vous, depuis la consultation du planning jusqu’à l’enregistrement du rendez-vous ou son annulation/modification.



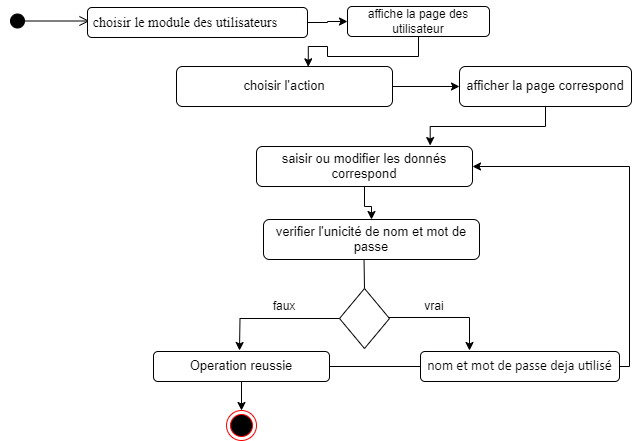
**Gérer le paiement**

Cette activité regroupe l’enregistrement des paiements effectués par les clients pour les services rendus (réparations, pièces remplacées, etc.). Elle comprend la saisie des montants, la génération des reçus, et la mise à jour de l’historique des transactions. Elle est essentielle pour assurer une bonne traçabilité financière.



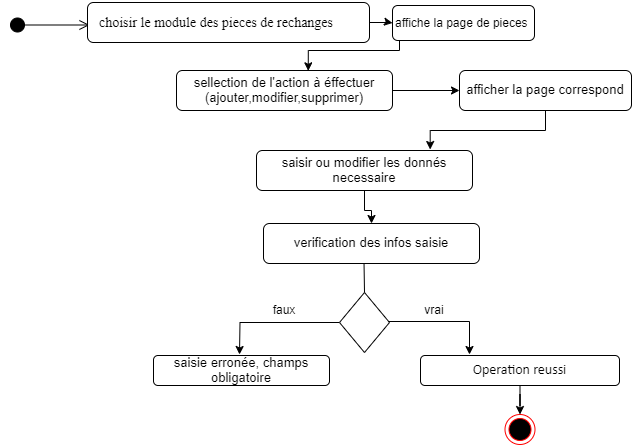
**Gérer les utilisateurs (ajout, modification, suppression)**

Réservée à l’administrateur, cette activité permet de gérer les comptes des différents utilisateurs du système (réceptionniste, caissier, etc.). Elle inclut la création de nouveaux comptes, la mise à jour des droits d’accès et la suppression des comptes inactifs ou non autorisés.



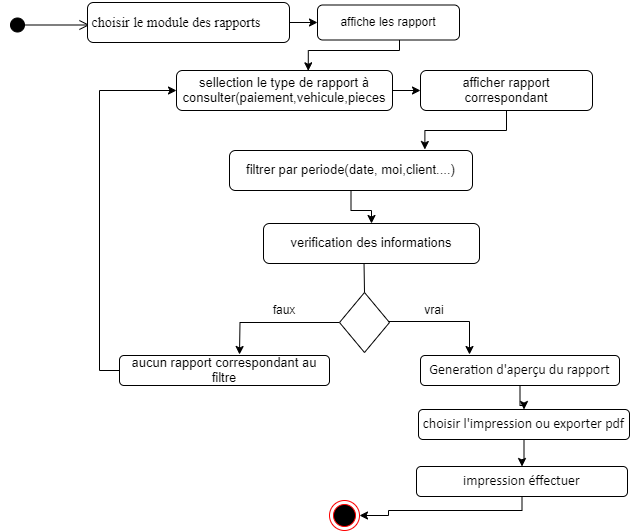
**Gérer des pièces de rechange**

Cette activité permet de suivre l’entrée et la sortie des pièces de rechange. Elle assure un bon niveau de stock, évite les ruptures, et alerte en cas de seuil critique. C’est un élément clé pour la continuité des opérations de réparation.



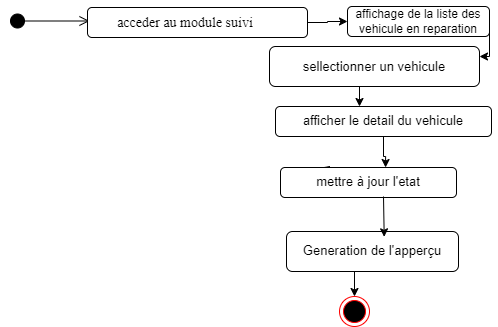
**Cosulter et Imprimer les rapports**

Les rapports générés permettent de visualiser les statistiques ou historiques liés aux activités du garage (ventes, réparations, paiements, etc.). Cette activité donne la possibilité d’afficher ces rapports à l’écran ou de les imprimer pour les besoins administratifs ou d’audit.

****

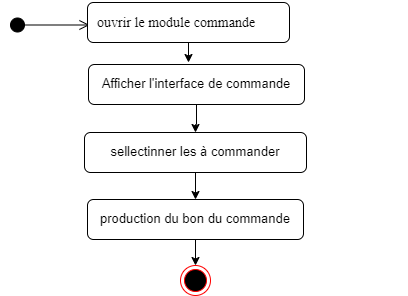
**Suivre les réparations**

Ce diagramme montre la logique de consultation et de suivi d’une réparation, du choix du véhicule à la mise à jour de son état.

****

**Commander les pièces de rechange**

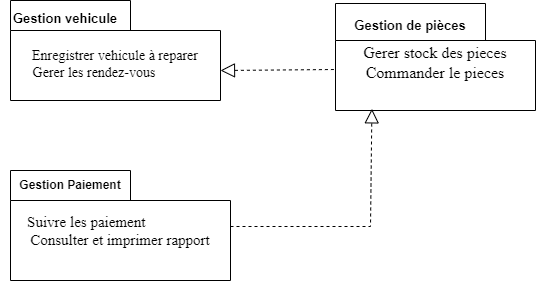
Ce diagramme illustre les étapes du processus de commande : analyse du stock, saisie des pièces à commander, validation, génération et enregistrement du bon de commande.

****

Ces diagrammes offrent une vision opérationnelle et séquentielle des actions à réaliser, tout en tenant compte des règles métiers spécifiques à chaque fonctionnalité du système. Ils permettent d’identifier clairement les enchaînements logiques entre les tâches, les points de décision, ainsi que les interactions entre les acteurs et le système. Grâce à cette modélisation, les développeurs disposent d’un cadre structuré facilitant l’implémentation des processus métier tout en assurant la cohérence avec les exigences fonctionnelles définies en amont.

**III.2.5. Diagramme de Package (Cas d’utilisation)**

Le **diagramme de packages** (ou **diagramme de paquets**) est un diagramme structurel de l’UML qui permet de représenter **l’organisation modulaire** d’un système en regroupant les éléments (comme les cas d’utilisation, les classes, ou les composants) dans des **packages logiques**. Dans le contexte des **cas d’utilisation**, ce diagramme aide à organiser et à structurer les différents cas d’utilisation du système en sous-ensembles cohérents, souvent en fonction des domaines fonctionnels ou des responsabilités métier. Cela facilite la **lisibilité** du modèle et permet une **gestion modulaire** du système. D’après Bhuvaneswari et Sekar (2020), ce type de diagramme « permet de mieux comprendre la décomposition du système en sous-systèmes et les dépendances entre eux ».



### ****III.2.6. Matrice de validation des cas d’utilisations****

La **matrice de validation des cas d’utilisation**, également connue sous le nom de **Requirements Traceability Matrix (RTM)**, est un outil d’analyse essentiel dans le processus de développement logiciel. Elle permet d’assurer une **traçabilité bidirectionnelle** entre les **exigences fonctionnelles exprimées par les utilisateurs** et les **cas d’utilisation définis lors de la conception**. Cet outil garantit que chaque besoin est bien couvert par une ou plusieurs fonctionnalités du système, ce qui facilite la **vérification,** la **validation,** et le **suivi de la couverture des exigences.** Les pratiques actuelles privilégient l’intégration de matrices automatisées, permettant une mise à jour dynamique et réduisant les erreurs humaines, notamment dans des environnements soumis à des contraintes de qualité ou de conformité (Visure Solutions, 2023 ; ValGenesis, 2024). Ainsi, la RTM joue un rôle clé dans le maintien de la cohérence entre les besoins initiaux et les solutions mises en œuvre tout au long du cycle de vie du système.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S’authentifier | Enregistrement des vehicules à réparer | Gestion de rendez vous | Suivi des paiements | Gerer le stock de pieces rechange | Commander les pieces de rechanges | Suivre reparation | Gerer les utilisateurs | Consulter et Imprimer les rapports |
| S’authentifier |  |  |  |  |  |  |  |  |  |
| Enregistrement des véhicules à réparer |  |  |  |  |  |  |  |  |  |
| Gestion de stock des pièces de rechange |  |  |  |  |  |  |  |  |  |
| Gestion de commande de pièces |  |  |  |  |  |  |  |  |  |
| Gestion de rendez-vous |  |  |  |  |  |  |  |  |  |
| Suivi de paiement |  |  |  |  |  |  |  |  |  |
| Gestion des utilisateurs |  |  |  |  |  |  |  |  |  |
| Suivi de réparations |  |  |  |  |  |  |  |  |  |
| Consultation et Impression des rapports |  |  |  |  |  |  |  |  |  |

Tableau : Matrice de validation des cas d’utilisations

**3.3. MODÉLISATION STATIQUE DU SYSTEME**

**3.3.1. Conception du modèle logique (Diagramme de classes)**

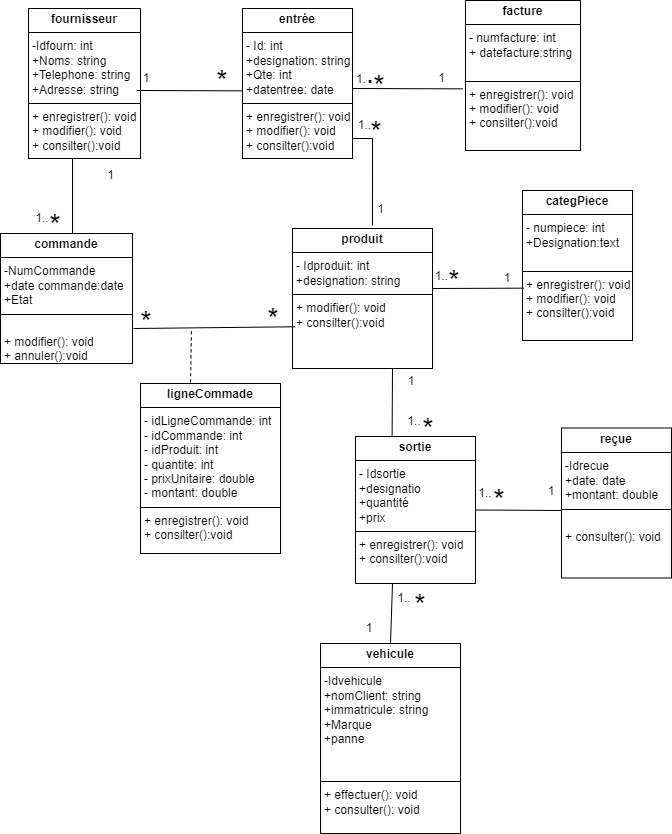
Le **diagramme de classes** est un diagramme structurel du langage UML (Unified Modeling Language) qui permet de représenter la structure statique d’un système orienté objet. Il modélise les **classes**, leurs **attributs**, leurs **méthodes**, ainsi que les **relations** (associations, héritages, dépendances, compositions) qui existent entre elles. Ce diagramme constitue un outil fondamental dans l’analyse et la conception logicielle, facilitant la compréhension, la communication et l’implémentation du système. D’après Chonoles et Schardt (2022), le diagramme de classes sert à « montrer les éléments de base du système et comment ils interagissent à travers des liens structurels ».

Le **diagramme de classes** joue un rôle clé dans le passage de l’analyse à la conception technique. Il permet de **décrire les entités métiers**, leurs **propriétés**, leurs **comportements** et les **liens structurels** qu’elles entretiennent dans le système. Il offre ainsi une vue d’ensemble de l’architecture logique du système, essentielle pour les phases de développement. Comme le précisent Lano et Clark (2021), il s’agit d’un « pont entre les exigences fonctionnelles et les solutions techniques », facilitant la traduction des besoins utilisateurs en structures logicielles exploitables.

Le diagramme de classes permet de :

* Identifier les **objets métiers** du système (ex. : Pièce, Facture, Paiement, Utilisateur) ;
* Définir leurs **caractéristiques** (attributs) et **comportements** (méthodes) ;
* Déterminer les **relations** entre les différentes classes (ex. : un utilisateur peut générer plusieurs factures) ;
* Servir de **modèle de référence** pour le développement de la base de données ou du code source orienté objet.

Il offre une vue **statique et conceptuelle** du système, contrairement aux diagrammes de séquence ou d’activités qui se concentrent sur la dynamique des interactions.



**3.3.2. Conception du Modèle Logique des données (Schéma Relationnel)**

Un schéma relationnel est une représentation logique d’une base de données relationnelle. Il spécifie les différentes relations (ou tables) du système ainsi que leurs attributs, c’est-à-dire les colonnes de chaque table. Chaque relation contient une clé primaire (soulignée), qui permet d’identifier de façon unique chaque enregistrement, et peut contenir une ou plusieurs clés étrangères (précédées du symbole #) qui permettent d’établir des liens avec d’autres relations.

Le schéma relationnel permet de structurer les données de manière normalisée, ce qui garantit la cohérence, l’intégrité et la non-redondance des informations stockées dans la base de données. Il constitue une étape fondamentale dans la modélisation conceptuelle et logique d’un système d’information, car il facilite à la fois la conception de la base et son implémentation sur un système de gestion de base de données(SGBD).  
Comme le soulignent Yunianto, Putra et Rahmad (2023), un schéma relationnel bien normalisé améliore la clarté, la performance et la fiabilité de la base de données relationnelle.

**Fournisseur**(*idFourn*, noms, téléphone, adresse)

**Produit**(*idProduit*, désignation, unité, prix, #numPiece)

**CategoriePiece**(*numPiece*, désignation)

**Vehicule**(*idVehicule*, nomClient, immatricule, marque, panne)

**Commande**(*numCommande*, #idFourn, dateCommande, état)

**LigneCommande**(*idLigneCommande*, #numCommande, #idProduit, quantite, prixUnitaire, montant)

**Entrée**(*idEntree*, #idProduit, qte, dateEntree

**Sortie**(*idSortie*, #idProduit, quantite, prix)

**Facture**(*numFacture*, dateFacture)

### III.3.2. Diagramme d’objets

Le **diagramme d’objets** est un type de diagramme statique de l’UML (Unified Modeling Language) qui illustre une **vue instantanée** des objets du système et de leurs **relations à un moment donné de l'exécution**. Contrairement au diagramme de classes qui représente une vision abstraite et générale, le diagramme d’objets montre des **instances concrètes** des classes avec des **valeurs spécifiques d’attributs.**

Selon Booch, Rumbaugh et Jacobson (2005), le diagramme d’objets est un outil précieux pour analyser les **états d’objets en exécution,** et il aide à valider la conception du modèle de classes en visualisant des **cas réels d’interactions** entre objets.

### 

### III.3.4. Conception des interfaces

La **conception des interfaces utilisateur (UI)** constitue une étape cruciale dans le processus de développement d’un système interactif. Elle vise à garantir une interaction fluide entre l’utilisateur et le système, en tenant compte à la fois des principes **d’ergonomie, d’accessibilité et d’efficacité** (Shneiderman et al., 2016).

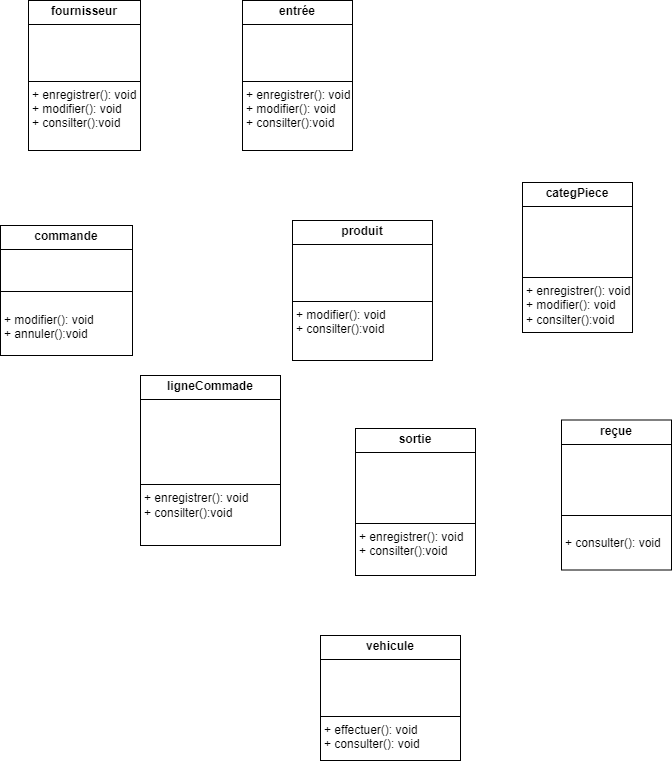
#### a) Définition

Une interface utilisateur est l’ensemble des moyens qui permettent à un utilisateur d’interagir avec un système informatique. Elle comprend généralement les **écrans, boutons, menus, formulaires,** etc. Elle doit être intuitive, fonctionnelle et adaptée au profil des utilisateurs (Galitz, 2007).

#### b) Objectifs de la conception des interfaces

Dans notre système de gestion du garage, la conception des interfaces a pour objectifs de :

* Permettre un enregistrement facile des clients, véhicules, et pièces de rechange ;
* Assurer une facturation rapide et fiable ;
* Simplifier les opérations de caisse ;
* Garantir une navigation simple pour le réceptionniste et le caissier.



### III.3.5. Conception de déploiement (diagramme de déploiement)

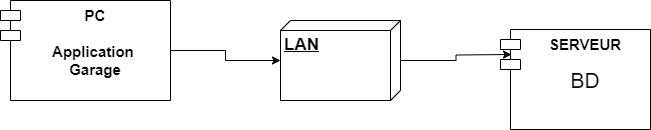
Le **diagramme de déploiement** est un diagramme structurel de l’UML qui permet de représenter la **configuration physique** du système logiciel sur l’infrastructure matérielle. Il décrit la manière dont les **composants logiciels** (applications, bases de données, services, etc.) sont **déployés sur les nœuds matériels** (serveurs, postes clients, imprimantes réseau, etc.) et met en évidence les **liens de communication** entre ces éléments. Ce diagramme offre une vision globale de l’environnement d’exécution du système, facilitant ainsi la **planification du déploiement, l’identification des dépendances réseau** et la **prévision des ressources matérielles nécessaires**. D’après Miro (2023), le diagramme de déploiement « aide les équipes techniques à comprendre comment les logiciels sont distribués à travers une infrastructure matérielle spécifique ». Dans le cadre du projet d’informatisation du **GARAGE PEINT BUTEMBO,** l’architecture retenue est une **architecture client-serveur à deux tiers,** reconnue pour sa simplicité, sa maintenabilité et sa fiabilité, où le client (poste du réceptionniste, comptable, etc.) interagit avec un serveur central hébergeant la base de données et les composants métiers essentiels de l’application.

* **Tier 1 – Client** : Les utilisateurs (réceptionniste, caissier, administrateur) accèdent à l’application via des **postes clients** connectés au réseau local (LAN), avec une interface utilisateur graphique.
* **Tier 2 – Serveur de données** : Un **serveur centralisé** héberge la base de données relationnelle, assurant la persistance des données liées aux pièces détachées, aux paiements, aux utilisateurs, aux services, etc.

Une **imprimante réseau partagée,** connectée au système, est utilisée pour **générer des factures, rapports d’inventaire ou documents de gestion**.

De manière concrète :

* Le **réceptionniste** utilise un **PC client** pour consulter ou enregistrer les véhicules à réparer et les pièces de rechange à utiliser ausii que les rendez-vous;
* Le **comptable** effectue les paiements et imprime les reçus via un autre **poste client** ;
* **Le chef d’atelier** accède au système via un **terminal sécurisé** pour gérer les utilisateurs, surveiller les opérations et gérer les permissions ;
* Une **imprimante réseau** permet l’émission des documents nécessaires à la gestion quotidienne.



### ****Conclusion****

Ce troisième chapitre a exposé de manière détaillée la démarche méthodologique suivie pour atteindre les objectifs de cette étude. Il a permis de justifier le choix de la méthode appliquée, de définir les techniques de collecte de données utilisées, ainsi que les outils et langages informatiques retenus pour le développement du système. Grâce à l’utilisation du langage UML pour la modélisation, combinée à l’environnement de développement C# et au SGBD SQL Server, une base solide a été posée pour la réalisation technique du système. Cette approche a également permis d’assurer la cohérence entre les besoins exprimés par les utilisateurs du garage et la solution informatique envisagée. Ainsi, la méthodologie adoptée constitue un fondement essentiel pour aborder, dans le chapitre suivant, la phase de concrétisation du système à travers son implémentation, ses tests et sa mise à l’épreuve dans un environnement réel.

### Références

1. Object Management Group (OMG). (2017). UML Superstructure Specification, Version 2.5.1. https://www.omg.org/spec/UML
2. Référence : Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2020). Systems Analysis and Design in a Changing World (8th ed.). Cengage Learning.
3. Larman, C. (2005). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.). Prentice Hall.
4. Arlow, J., & Neustadt, I. (2013). UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd ed.). Addison-Wesley.
5. Larman, C. (2005). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.). Pearson Education.
6. Arlow, J., & Neustadt, I. (2013). *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley.
7. Larman, C. (2004). *Applying UML and Patterns*. Prentice Hall.
8. Fowler, M. (2003). *UML Distilled*. Addison-Wesley.
9. Dennis, A., Wixom, B., & Tegarden, D. (2021). *Systems Analysis and Design*. Wiley.
10. Dennis, A., Wixom, B. H., & Roth, R. M. (2021). *Systems Analysis and Design* (7th ed.). Wiley.
11. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.
12. Larman, C. (2019). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Pearson Education.
13. Yunianto, D. R., Putra, Y. E., & Rahmad, C. (2023). *Comparison of relational database modeling performance based on number of normalized entities*. SMARTICS Journal, **9**(1), 42–48. <https://doi.org/10.21067/smartics.v9i1.8390>

Chonoles, M. J., & Schardt, J. A. (2022). UML 2 for Dummies (2nd ed.). Wiley.

Lano, K., & Clark, T. (2021). UML Modeling for Business Analysts and Software Engineers. Springer. <https://doi.org/10.1007/978-3-030-71646-3>

ValGenesis. (2024, 13 février). *The benefits of automating your requirements traceability matrix (RTM)*. https://www.valgenesis.com/blog/automating-requirements-traceability-matrix

Visure Solutions. (2023). *Requirements Traceability Matrix (RTM): How-To Guide*. <https://visuresolutions.com/blog/requirements-traceability-matrix/>