

Cahier des charges

Application

Table des matières

1. Contexte du projet	1
2. Objectifs du projet	1
3. Périmètre fonctionnel	1
4. Maquettes et Wireframe	2
4.1 Maquettes	2
4.2 Wireframes	3
5. Modélisation des données	3
5.1 Dictionnaire de données	3
5.2 Modèle conceptuel de données (MCD)	5
5.3 Modèle logique de données (MLD)	5
5.4 Modèle physique de données (MPD)	6
6. Technologies utilisées	7
7. Diagrammes	7
7.1 Diagramme d'activité	7
7.2 Diagramme de classe	8
7.3 Diagramme cas d'utilisation	8
8. Contraintes techniques	9
8.1. Architecture logicielle	9
8.2. Langage et environnement de développement	9
8.3. Gestion des données	9
8.4. Interface utilisateur	10
8.5. Gestion des erreurs	10
8.6. Interopérabilité et évolutivité	10
9. Sécurité	10
10. Modalités de réalisation	10
10.1 Organisation du travail	10
10.2 Gestion du projet	10
10.3 Validation des étapes	Erreur ! Signet non défini.
11. Planning prévisionnel	11
12. Livrables	11
13. Critères d'acceptation	12

1. Contexte du projet

L'application Java pour la gestion des cartes grises est développée dans le cadre d'un projet académique du BTS Services Informatiques aux Organisations (option SLAM). Le projet a pour objectif de concevoir, développer et sécuriser une solution de gestion des cartes grises pour une préfecture.

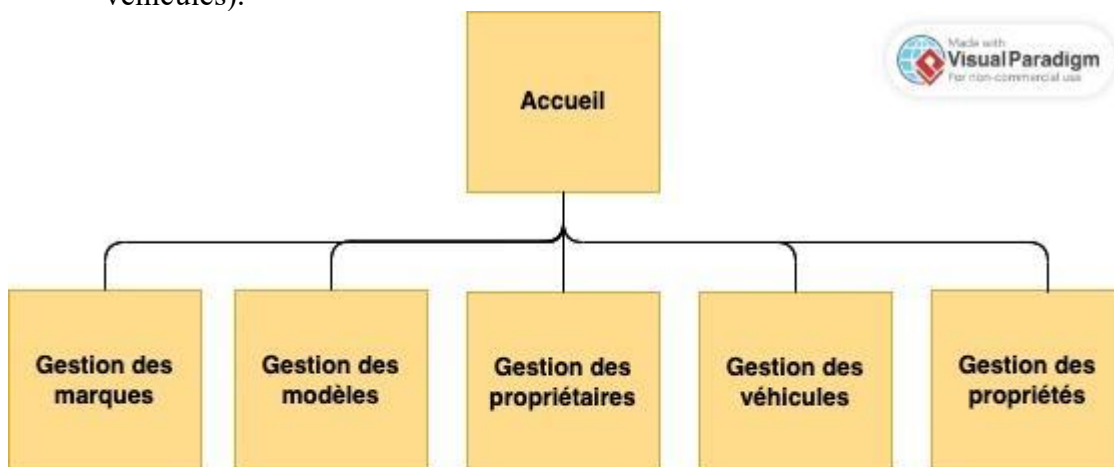
2. Objectifs du projet

- Assurer la gestion de la possession des véhicules par leurs propriétaires.
- Permettre l'ajout, la modification, la suppression et l'affichage des informations des propriétaires et des véhicules.
- Sécuriser les données sensibles des utilisateurs.

3. Périmètre fonctionnel

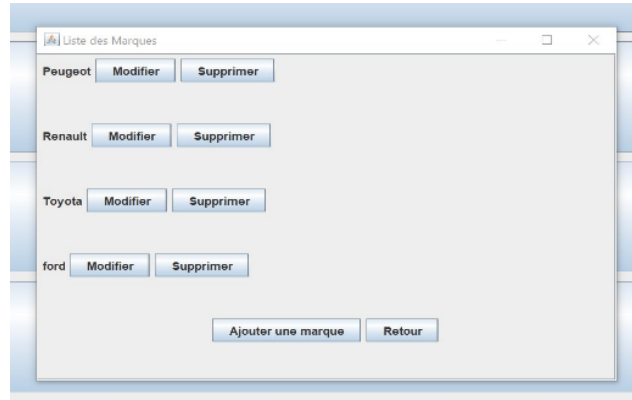
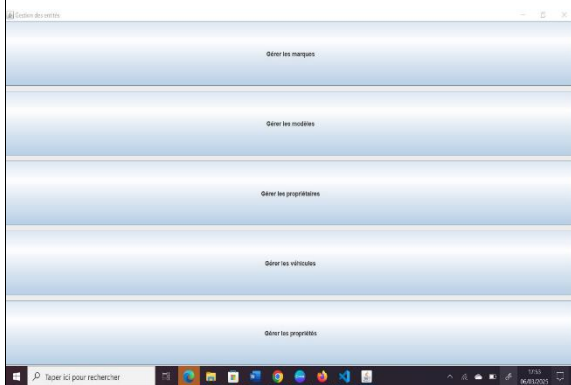
L'application doit couvrir les fonctionnalités suivantes :

- Gestion des propriétaires (ajout, modification, suppression, consultation).
- Gestion des véhicules (ajout, modification, suppression, consultation).
- Gestion des marques et des modèles de véhicules.
- Gestion des relations de possession (un propriétaire peut posséder plusieurs véhicules).




4. Maquettes et Wireframe

4.1 Maquettes



Modifier le Propriétaire

 **Nom :**


Prénom :

Adresse :

Code Postal :

Ville :

Ajouter un Propriétaire

 **Nom :**

Prénom :

Adresse :

Code Postal :

Ville :

4.2 Wireframes



5. Modélisation des données

La modélisation des données est essentielle pour structurer les informations de manière logique et cohérente. Elle permet de comprendre les relations entre les entités principales du système.

5.1 Dictionnaire de données

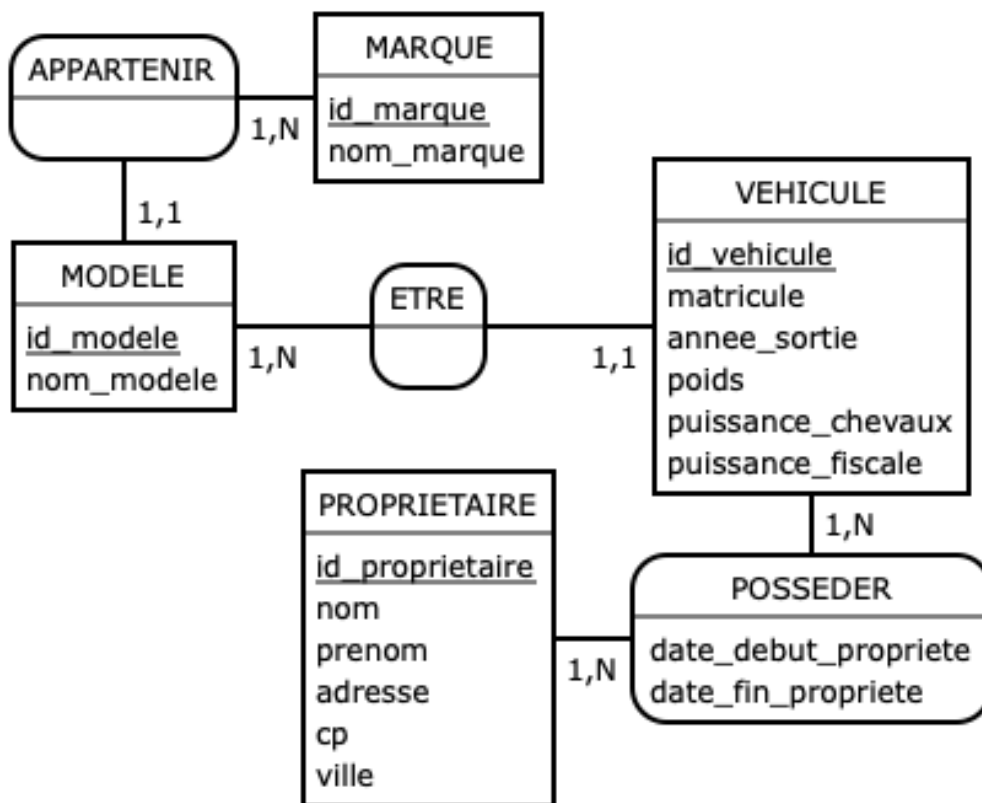
Nom de la table	Nom de l'attribut	Type	Description
MARQUE	id_marque	INT	Identifiant unique, clé primaire, auto-incrémentée
	nom_marque	VARCHAR(255)	Nom de la marque

MODELE	id_modele	INT	Identifiant unique, clé primaire, auto-incrémentée
	nom_modele	VARCHAR(255)	Nom du modèle
	id_marque	INT	Référence à la marque (clé étrangère)
VEHICULE	id_vehicule	INT	Identifiant unique, clé primaire, auto-incrémentée
	matricule	VARCHAR(50)	Numéro d'immatriculation
	annee_sortie	INT	Année de sortie du véhicule
	poids	INT	Poids du véhicule
	puissance_chevaux	INT	Puissance en chevaux
	P uissance_fiscale	INT	Puissance fiscale
	id_modele	INT	Référence au modèle (clé étrangère)
PROPRIETAIRE	id_proprietaire	INT	Identifiant unique, clé primaire, auto-incrémentée
	nom	VARCHAR(100)	Nom du propriétaire
	prenom	VARCHAR(100)	Prénom du propriétaire
	adresse	VARCHAR(255)	Adresse postale
	cp	VARCHAR(10)	Code postal
	ville	VARCHAR(100)	Ville
POSSEDER	id_proprietaire	INT	Référence au propriétaire (clé étrangère)
	id_vehicule	INT	Référence au véhicule (clé étrangère)
	date_debut_propriete	DATE	Date de début de possession
	date_fin_propriete	DATE	Date de fin de possession

5.2 Modèle conceptuel de données (MCD)

Le modèle conceptuel de données (MCD) est une représentation abstraite et simplifiée des entités et de leurs relations. Dans le cadre de cette application, les entités principales sont :

- **MARQUE** : représente les constructeurs automobiles.
- **MODELE** : désigne les différents modèles de véhicules associés à une marque.
- **VEHICULE** : caractérise les informations techniques d'un véhicule spécifique.
- **PROPRIETAIRE** : identifie les personnes physiques ou morales possédant des véhicules.
- **POSSEDER** : est une relation associative permettant de lier les propriétaires et leurs véhicules avec des dates de début et de fin de possession. Le MCD simplifie la compréhension des interactions entre ces entités sans entrer dans les détails techniques.



5.3 Modèle logique de données (MLD)

Le modèle logique de données (MLD) est une formalisation plus détaillée du MCD. Il définit les clés primaires, les clés étrangères ainsi que les contraintes d'intégrité.

MARQUE (*id_marque*, *nom_marque*)

- Clé primaire : *id_marque*

MODELE (*id_modele*, *nom_modele*, *id_marque*)

- Clé primaire : *id_modele*
- Clé étrangère : *id_marque* fait référence à *id_modele* de MARQUE

VEHICULE (*id_vehicule*, *matricule*, *annee_sortie*, *poids*, *puissance_chevaux*, *puissance_fiscale*, *id_modele*)

- Clé primaire : *id_vehicule*
- Clé étrangère : *id_modele* fait référence *id_modele* de MODELE

PROPRIETAIRE (*id_proprietaire*, *nom*, *prenom*, *adresse*, *cp*, *ville*)

- Clé primaire : *id_proprietaire*

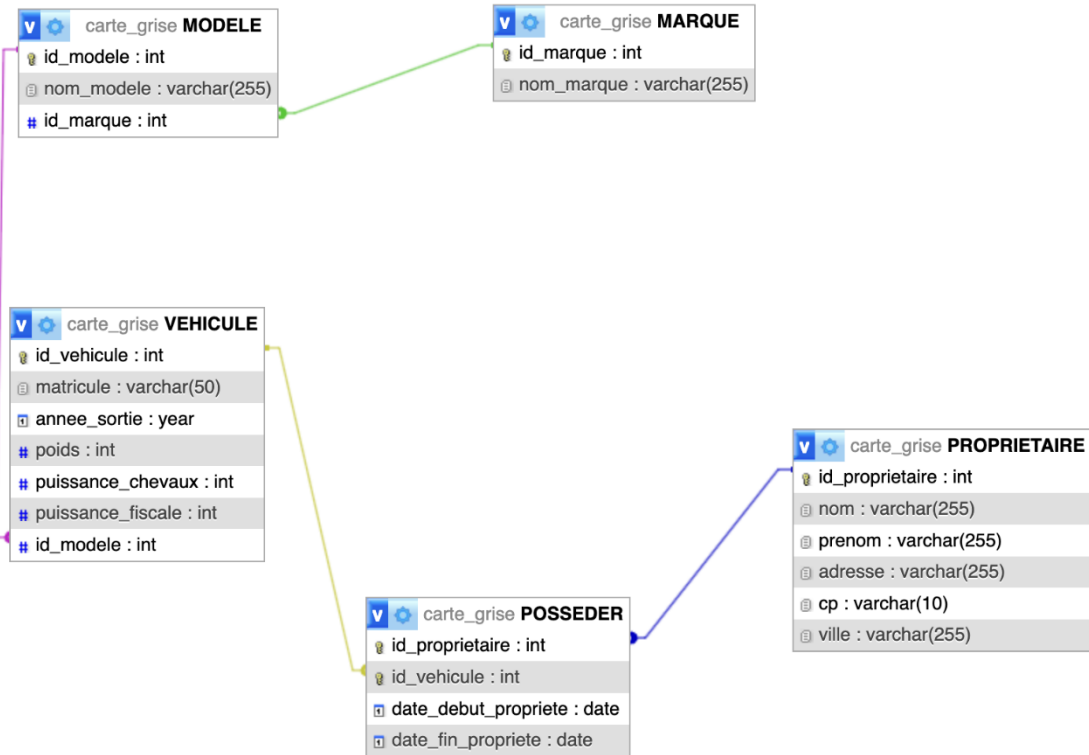
POSSEDER (*id_vehicule*, *#id_proprietaire*, *date_debut_propriete*, *date_fin_propriete*)

- Clé primaire : (*id_vehicule*, *id_proprietaire*, *date_debut_propriete*)
- Clés étrangères : *id_vehicule* fait référence à *id_vehicule* de VEHICULE
id_proprietaire fait référence à *id_proprietaire* de PROPRIETAIRE

5.4 Modèle physique de données (MPD)

Le modèle physique de données (MPD) décrit la structure finale de la base de données telle qu'elle sera implémentée sur le serveur MySQL. Il prend en compte les types de données SQL, les index et les contraintes d'intégrité :

- **Tables et colonnes** : les types SQL comme INT, VARCHAR et DATE sont utilisés.
- **Indexation** : les clés primaires et les clés étrangères assurent une indexation efficace.
- **Contraintes d'intégrité** : les contraintes NOT NULL, PRIMARY KEY, FOREIGN KEY et UNIQUE garantissent la cohérence des données. Le MPD est le guide de référence pour l'implémentation technique de la base de données.

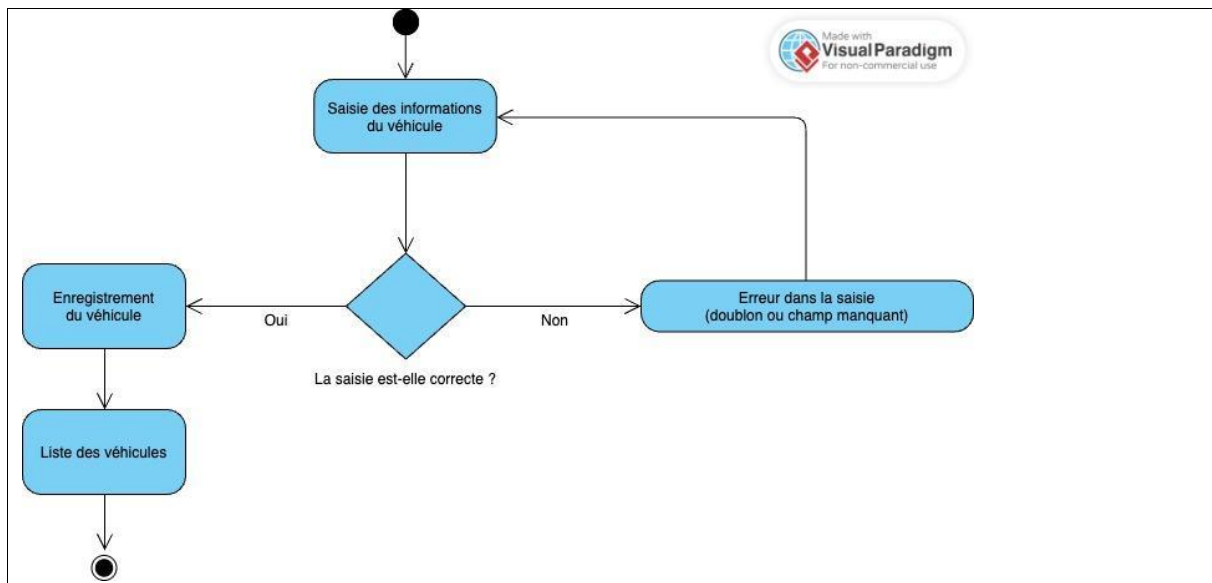


6. Technologies utilisées

- Langage de programmation : Java
- Base de données : MySQL
- IDE : Visual Studio Code
- Serveur Apache via MAMP
- Outil de modélisation : Mocodo, Visual Paradigm
- Gestion de projet : Trello

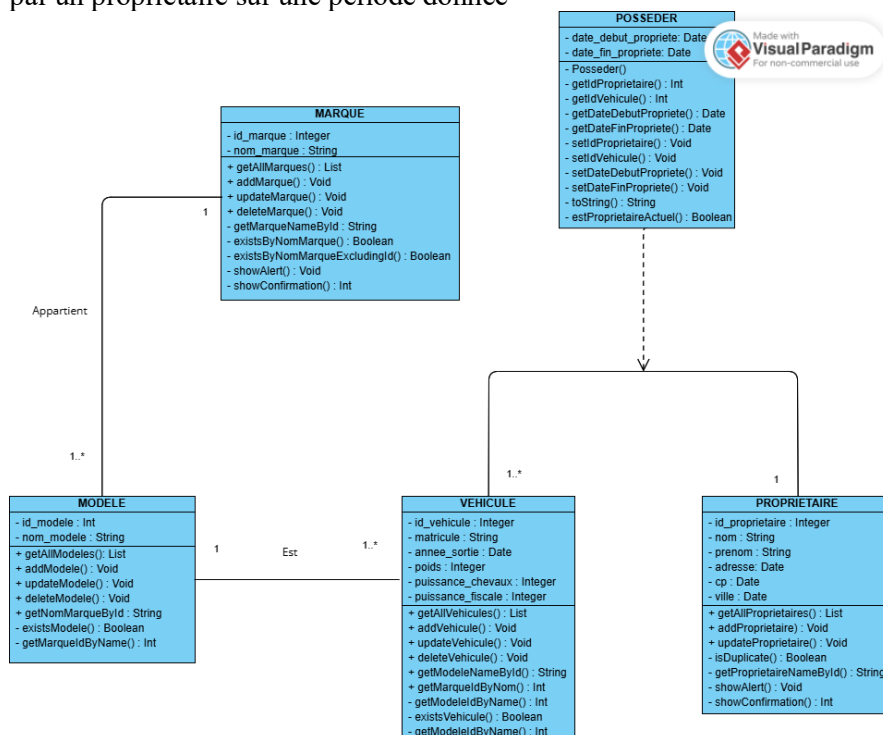
7. Diagrammes

7.1 Diagramme d'activité



7.2 Diagramme de classe

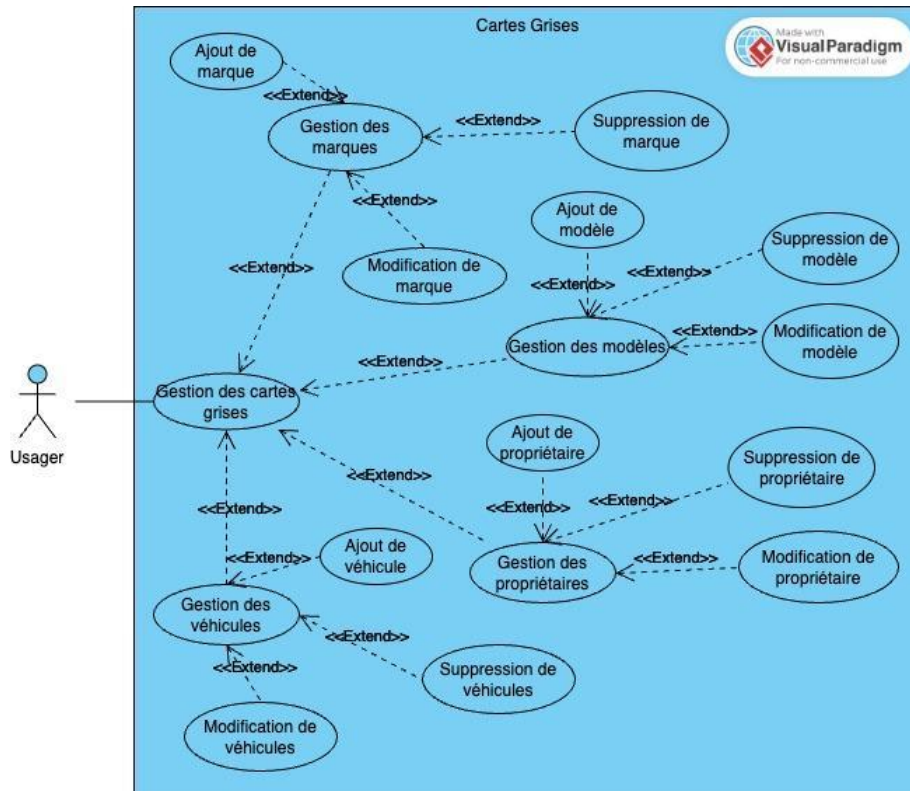
La table **"POSSEDER"** représente la relation entre les entités **"PROPRIETAIRE"** et **"VEHICULE"** dans le modèle de données. Elle est utilisée pour gérer la possession d'un véhicule par un propriétaire sur une période donnée



7.3 Diagramme cas d'utilisation

Ce diagramme de cas d'utilisation du système de gestion des cartes grises. L'utilisateur, en tant qu'acteur principal, peut interagir avec différentes fonctionnalités du système, notamment la gestion des marques, des modèles, des propriétaires et des véhicules. Chaque catégorie comprend des actions spécifiques comme l'ajout, la modification et la suppression. Des relations **<<include>>** indiquent les

dépendances obligatoires, tandis que <<extend>> représente des fonctionnalités supplémentaires accessibles selon le contexte.



8. Contraintes techniques

Les contraintes techniques définissent les règles d'implémentation à respecter pour assurer la qualité et la performance de l'application. Elles couvrent les aspects suivants :

8.1. Architecture logicielle

- Respect du modèle **MVC** (Modèle-Vue-Contrôleur) pour séparer la logique métier, la gestion des données et l'interface utilisateur.
- Utilisation des **DAO** (Data Access Object) pour l'accès aux données, assurant une couche d'abstraction avec la base de données.

8.2. Langage et environnement de développement

- **Java** est utilisé pour le développement côté backend et pour l'interface utilisateur via Java Swing.
- L'IDE principal est **Visual Studio Code**, avec les extensions adaptées pour Java.
- Le serveur **Apache** est configuré via **MAMP** pour héberger la base de données **MySQL**.
- Les outils de modélisation tels que **Mocodo** pour le MCD et **Visual Paradigm** pour les diagrammes UML sont utilisés.

8.3. Gestion des données

- Les requêtes SQL doivent respecter les bonnes pratiques d'optimisation pour éviter les performances dégradées.

- Mise en place de **transactions SQL** pour assurer la cohérence des opérations critiques (ajouts, modifications, suppressions).
- Indexation appropriée sur les clés primaires et les clés étrangères pour améliorer la rapidité des requêtes.

8.4. Interface utilisateur

- Utilisation de **Java Swing** pour l'interface graphique.
- Validation des champs de saisie pour éviter les erreurs de format et les données manquantes.
- Prise en compte de l'ergonomie pour une navigation intuitive et fluide.

8.5. Gestion des erreurs

- Gestion des exceptions Java pour anticiper et traiter les erreurs d'exécution (SQLException, NullPointerException, etc.).
- Messages d'erreurs clairs et compréhensibles pour l'utilisateur.
- Gestion des erreurs liées à la base de données, comme les conflits d'intégrité référentielle.

8.6. Interopérabilité et évolutivité

- Capacité à intégrer de nouvelles fonctionnalités (ex. : ajout d'un module de statistiques ou de gestion des amendes).
- Adaptabilité à d'autres SGBD (Oracle, PostgreSQL) en cas de migration future.

Ces contraintes assurent la stabilité, la sécurité et la maintenabilité de l'application tout au long de son cycle de vie.

- Respecter les normes du modèle MVC.
- Utilisation des requêtes SQL pour la gestion des données.
- Interface utilisateur avec Java Swing.
- Gestion des erreurs et validation des données utilisateur.

9. Sécurité

- Authentification et autorisation des utilisateurs.
- Protection des données sensibles.
- Prévention des injections SQL.

10. Modalités de réalisation

Les modalités de réalisation décrivent l'organisation du travail, les outils collaboratifs utilisés et les étapes nécessaires à la bonne conduite du projet.

10.1 Organisation du travail

- **Travail individuel et collaboratif** : Le développement est réalisé en classe lors des travaux dirigés et à domicile pour les tâches individuelles.
- **Réunions régulières** : Des réunions hebdomadaires permettent de faire le point sur l'avancement du projet, d'identifier les difficultés et de valider les choix techniques.

10.2 Gestion du projet

- **Outil de gestion de projet** : Trello est utilisé pour suivre les tâches, répartir les responsabilités et gérer les priorités.

- **Versionnage du code** : Le code source est hébergé sur GitHub, permettant un suivi rigoureux des modifications et facilitant le travail collaboratif.
- **Documentation** : Une documentation technique détaillée est produite, comprenant les spécifications fonctionnelles, le code commenté et les guides d'utilisation.
- **Validation intermédiaire** : À chaque fin de phase (conception, développement, tests), une validation est effectuée pour s'assurer du respect des objectifs.
- **Tests utilisateurs** : Des tests utilisateurs sont prévus pour valider l'ergonomie et la conformité aux attentes.
- Travail en classe et à domicile.
- Stockage du code source sur GitHub.
- Suivi du projet via Trello.

11. Planning prévisionnel

Le planning prévisionnel se déroule sur plusieurs mois pour assurer une progression cohérente du projet. Voici les différentes phases :

- **Phase de conception** (octobre 2024) : Analyse des besoins, élaboration du cahier des charges, modélisation des données (MCD, MLD, MPD).
- **Phase de développement** (novembre 2024 à janvier 2025) : Développement des fonctionnalités principales, intégration des modèles de données et réalisation de l'interface utilisateur.
- **Phase de tests** (février 2025) : Tests unitaires, tests d'intégration, validation des fonctionnalités et correction des anomalies.
- **Livraison finale** (mars 2025) : Livraison de l'application finale et présentation du projet. Des ajustements peuvent être réalisés en fonction des retours utilisateurs et des imprévus techniques.
- Conception : octobre 2024
- Développement : novembre 2024 à janvier 2025
- Tests : février 2025
- Livraison finale : mars 2025

12. Livrables

Les livrables sont les productions attendues à la fin de chaque phase du projet. Ils incluent :

- **Code source** : Hébergé sur GitHub, documenté et commenté.
- **Modèles de données** : Modèle conceptuel (MCD), logique (MLD) et physique (MPD), disponibles sous forme de schémas UML.
- **Documentation technique** : Contenant les spécifications détaillées, le guide d'installation et les instructions d'utilisation.
- **Rapport de tests** : Compte rendu des tests réalisés (unitaires, d'intégration) et validation finale des fonctionnalités.
- **Présentation finale** : Présentation du projet sous forme de diaporama pour expliquer les choix techniques et les résultats obtenus.
- Code source sur GitHub.
- Modèles de données (MCD, MLD, MPD).
- Interface utilisateur fonctionnelle.
- Rapport de tests.

13. Critères d'acceptation

Les critères d'acceptation permettent d'évaluer la conformité du projet par rapport aux objectifs fixés :

- **Respect des fonctionnalités** : Toutes les fonctionnalités listées dans le cahier des charges doivent être implémentées et fonctionnelles.
- **Qualité du code** : Code clair, bien structuré, respectant les normes Java et les bonnes pratiques de développement.
- **Absence de bugs bloquants** : Tous les bugs majeurs doivent être corrigés avant la livraison finale.
- **Performance et optimisation** : Temps de réponse acceptable pour l'utilisateur final, optimisation des requêtes SQL.
- **Conformité de la sécurité** : Protection des données sensibles, prévention des failles de sécurité (injections SQL, accès non autorisés).
- **Satisfaction des utilisateurs** : Retour positif des tests utilisateurs concernant l'ergonomie et l'efficacité de l'application.
- Respect des fonctionnalités définies.
- Absence de bugs bloquants.
- Respect des normes de sécurité et de performance.