BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2025**

Épreuve E6 - Conception et développement d'applications (option SLAM)
ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle (recto)

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE	**N° réalisation : 02**
Nom, prénom : Nerat Damien	**N° candidat : 02218593860**
Épreuve ponctuelle ☒ **Contrôle en cours de formation** ☐	**Date :** 12 /12 /2024

Organisation support de la réalisation professionnelle
Projet réalisé en séance de TP et à la maison

Intitulé de la réalisation professionnelle
Logiciel de gestion des cartes grises (JAVA)

Période de réalisation : Du 28/11/2024 au 06/03/2025 **Lieu : Lycée Latournelle, la Garenne-Colombes**
Modalité : ☒ **Seul(e)** ☐ **En équipes**

Compétences travaillées
☒ Concevoir et développer une solution applicative
☒ Assurer la maintenance corrective ou évolutive d'une solution applicative
☒ Gérer les données

Conditions de réalisation[1] (ressources fournies, résultats attendus)
Travail en classe et à la maison à partir d'un support étudiant reprenant un contexte professionnel.
L'objectif est de concevoir et développer un logiciel de gestions des cartes grises (ajout, modification et suppression d'informations). Le logiciel est développé en Java.

Description des ressources documentaires, matérielles et logicielles utilisées[6]
Ressources documentaires :

- Supports étudiants (cours), Travaux dirigés et travaux pratiques
- Sites de ressources en ligne : Stackoverflow, Open classroom. Youtube.
- Fiches ressources (Cours)

Ressources matérielles :

- Ordinateur portable
- Connexion internet
- Clavier, souris et écran

Ressources logicielles :

- IDE : Visual Studio Code
- Ensemble de solution logicielle : MAMP (Contient gestionnaire de base de données relationnelles MySQL)
- Outil de modélisation de base de données : Mocodo
- Outil de modélisation UML : Visual Paradigm
- Gestion de projet : Trello
- Outils de wdesign : Figma
- Plateforme de développement collaborative : Github

Langages

- Langages de programmation applicatif : Java
- Langage base de données : SQL

Modalités d'accès aux productions[2] et à leur documentation[3]
Cahier des charges techniques : https://github.com/Ghost-Rouge/JAVA/blob/main/CDC/CDC-JAVA-Damien.pdf
Code : https://github.com/Ghost-Rouge/JAVA

[1] En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO. 6
Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

[2] Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.* ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

[3] Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

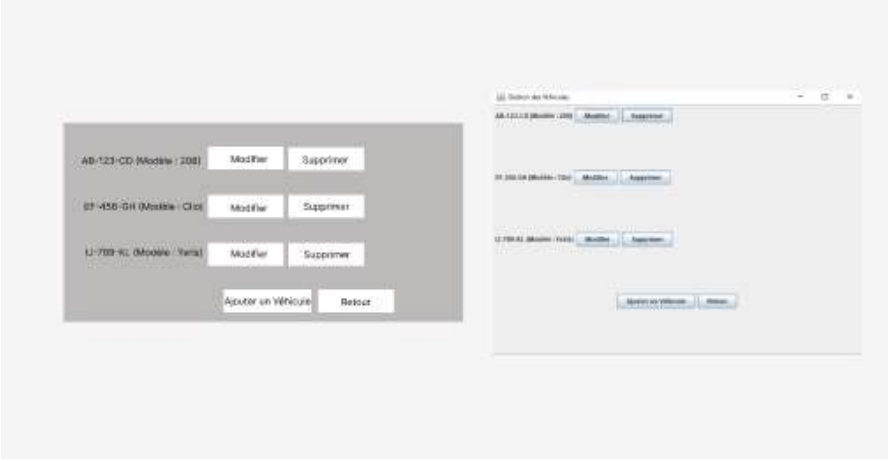
L'objectif est de concevoir et développer un logiciel de gestions des cartes grises (ajout, modification et suppression d'informations). Le logiciel est développé en Java.

Exemple de wireframes

Wireframes des affichages des données POSSEDER :

Wireframes du menu :

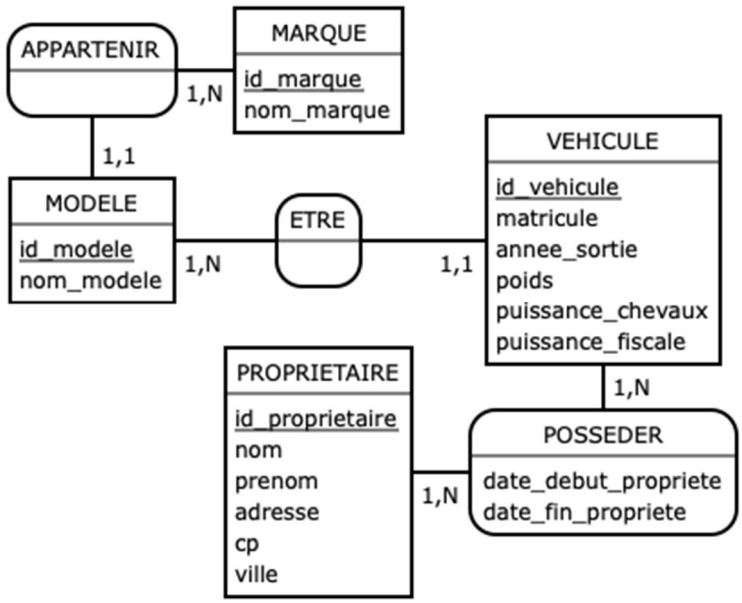
Wireframes des Gestions des Véhicules



Dictionnaire données

Nom	Description	Entité	Type
id_marque	Identifiant unique de la marque	marque	INT, PK
nom_marque	Nom de la marque	marque	VARCHAR(100)
id_modele	Identifiant unique du modèle	modele	INT, PK
nom_modele	Nom du modèle	modele	VARCHAR(100)
id_marque	Référence à la marque du modèle	modele	INT, FK
id_vehicule	Identifiant unique du véhicule	vehicule	INT, PK
matricule	Numéro d'immatriculation	vehicule	VARCHAR(20)
annee_sortie	Année de sortie du véhicule	vehicule	YEAR
poids	Poids du véhicule en kg	vehicule	DECIMAL(6,2)
puissance_chevaux	Puissance en chevaux	vehicule	INT
puissance_fiscale	Puissance fiscale	vehicule	INT
id_modele	Référence au modèle du véhicule	vehicule	INT, FK
id_proprietaire	Identifiant unique du propriétaire	proprietaire	INT, PK
nom	Nom du propriétaire	proprietaire	VARCHAR(100)
prenom	Prénom du propriétaire	proprietaire	VARCHAR(100)
adresse	Adresse complète	proprietaire	VARCHAR(255)
cp	Code postal	proprietaire	VARCHAR(10)
ville	Ville	proprietaire	VARCHAR(100)
id_vehicule	Référence au véhicule possédé	posseder	INT, FK
id_proprietaire	Référence au propriétaire	posseder	INT, FK
date_debut_propriete	Date de début de propriété	posseder	DATE, PK
date_fin_propriete	Date de fin de propriété	posseder	DATE

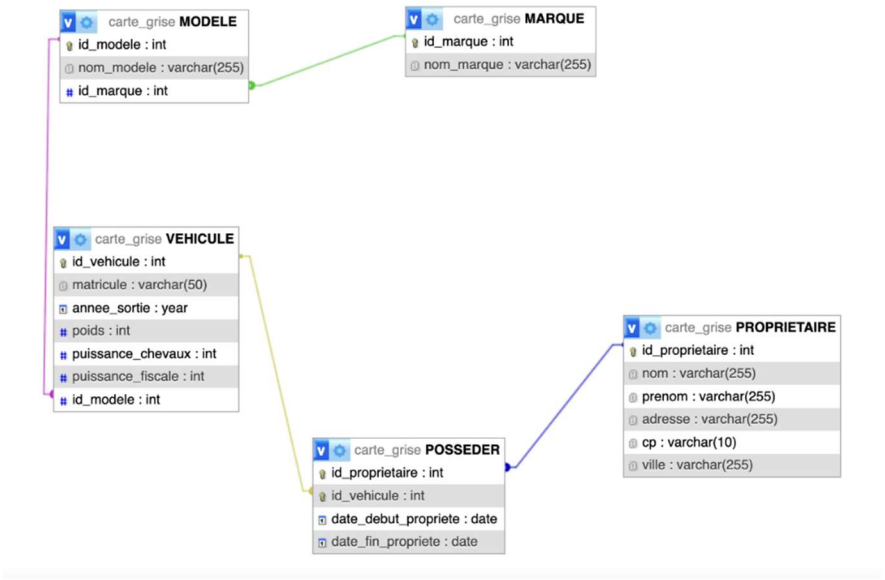
MCD



MLD (format BTS)

- **marque (id_marque, nom_marque)**
  - **Clé primaire : id_marque**
- **modele (id_modele, nom_modele, id_marque)**
  - **Clé primaire : id_modele**
  - **Clé étrangère : id_marque en référence à id_marque de MARQUE**
- **vehicule (id_vehicule, matricule, annee_sortie, poids, puissance_chevaux, puissance_fiscale, id_modele)**
  - **Clé primaire : id_vehicule**
  - **Clé étrangère : id_modele en référence à 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**
  - **Clé étrangère : id_vehicule en référence à id_vehicule de VEHICULE id_proprietaire en référence à id_proprietaire de PROPRIETAIRE**

MDP



DIGRAMME DE CAS D'UTILISATION

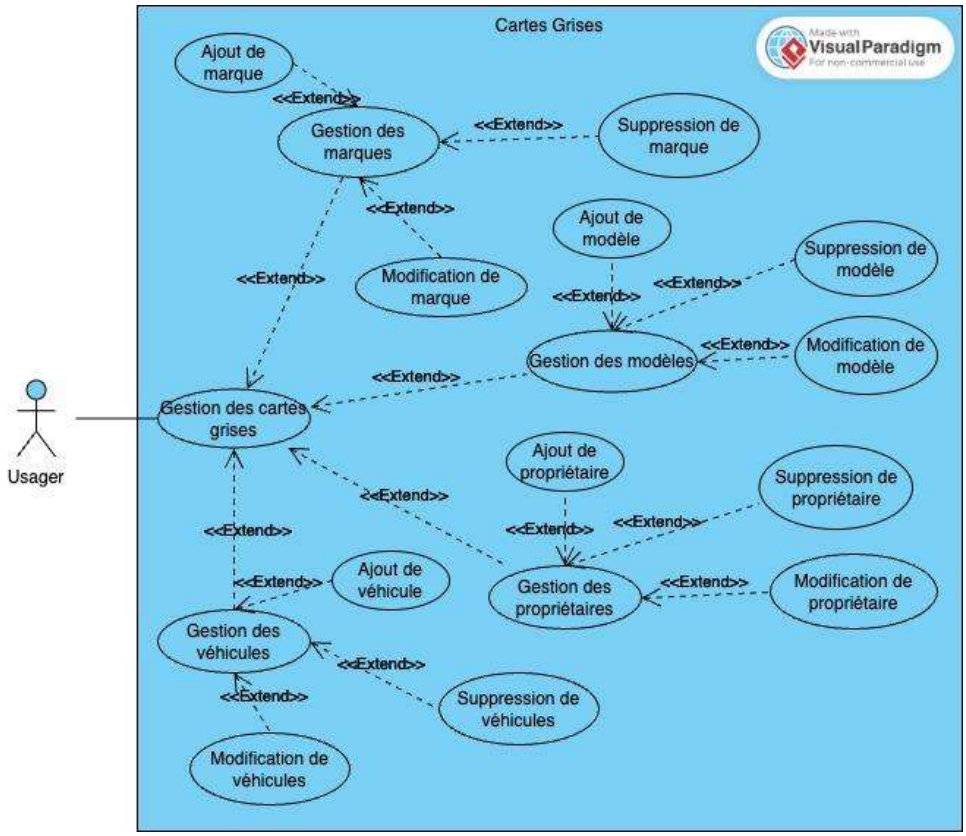
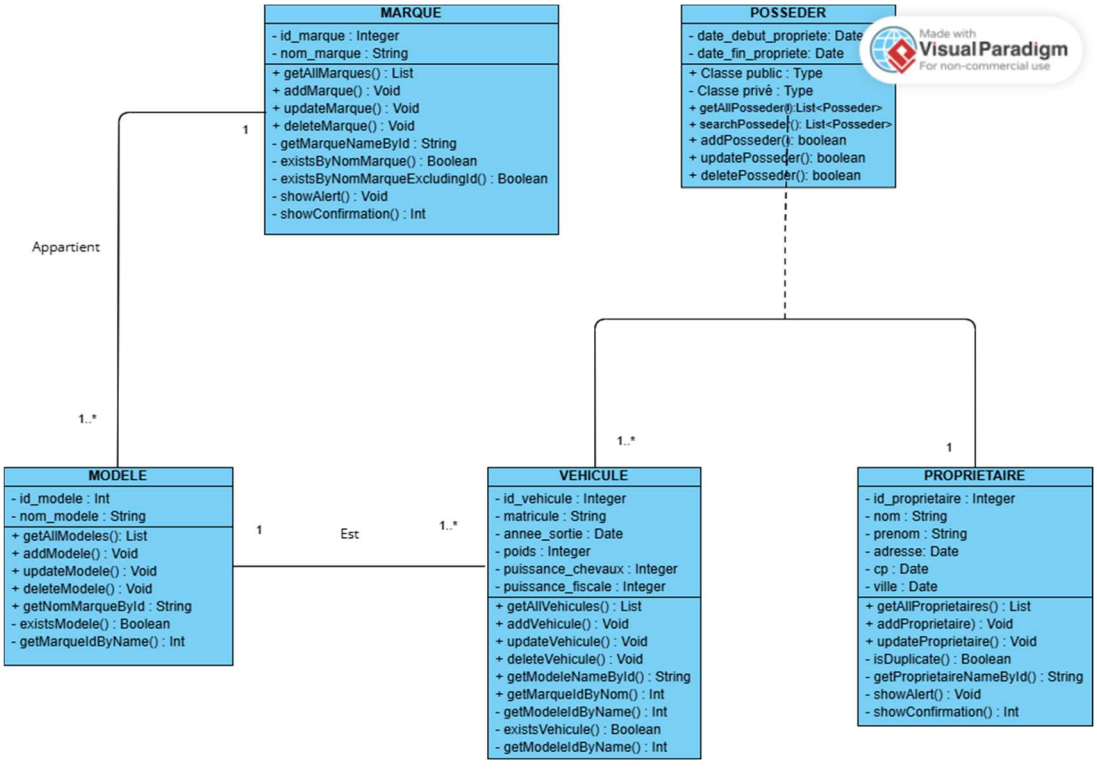


DIAGRAMME DE CLASSE



Exemple de Code

```
package views;

import controllers.PossederController;
import models.Posseder;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

public class PossederView extends JFrame {

    private JTable possessionTable;
    private PossederController possederController;

    public PossederView() {
        // Initialiser le contrôleur
        possederController = new PossederController();

        // Définir le titre de la fenêtre
        setTitle("Gestion des Propriétés");
        // Définir la taille de la fenêtre
        setSize(600, 400);
        // Centrer la fenêtre sur l'écran
        setLocationRelativeTo(null);
        // Utiliser un layout pour organiser les composants
        setLayout(new BorderLayout());

        // Ajouter un panneau de gestion des données
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());

        // Créer un bouton "Ajouter"
        JButton addButton = new JButton("Ajouter");
        addButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Ouvrir la fenêtre d'ajout de données
                showAddDialog();
            }
        });

        // Créer un bouton "Mettre à jour"
        JButton updateButton = new JButton("Mettre à jour");
        updateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Ouvrir la fenêtre de mise à jour de données
                showUpdateDialog();
            }
        });
```

```
    // Créer un bouton "Supprimer"
    JButton deleteButton = new JButton("Supprimer");
    deleteButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Supprimer une ligne sélectionnée dans le tableau
            deletePossession();
        }
    });

    // Ajouter les boutons dans un JPanel au dessus du tableau
    JPanel buttonPanel = new JPanel();
    buttonPanel.add(addButton);
    buttonPanel.add(updateButton);
    buttonPanel.add(deleteButton);

    // Ajouter les boutons et le tableau dans la fenêtre
    add(buttonPanel, BorderLayout.NORTH);

    // Initialiser la table avec les données
    initPossessionTable();

    // Rendre la fenêtre visible
    setVisible(true);
}

private void initPossessionTable() {
    List<Posseder> possessions = possederController.getAllPossessions();

    // Créer un modèle de table avec les données
    String[] columnNames = {"Nom Propriétaire", "Matricule Véhicule", "Date Début", "Date Fin"};
    Object[][] data = new Object[possessions.size()][4];

    for (int i = 0; i < possessions.size(); i++) {
        Posseder possession = possessions.get(i);

        // Récupérer le nom du propriétaire et le matricule du véhicule à partir des ID
        String nomProprietaire = possederController.getNomProprietaire(possession.getIdProprietaire());
        String matriculeVehicule = possederController.getMatriculeVehicule(possession.getIdVehicule());

        // Remplir les données de la table
        data[i][0] = nomProprietaire;  // Nom du propriétaire
        data[i][1] = matriculeVehicule;      // Matricule du véhicule
        data[i][2] = possession.getDateDebutPropriete(); // Date début
        data[i][3] = possession.getDateFinPropriete();    // Date fin
    }

    // Créer une nouvelle table avec les données mises à jour
    possessionTable = new JTable(data, columnNames);
    JScrollPane scrollPane = new JScrollPane(possessionTable);

    // Ajouter la table au centre de la fenêtre
    add(scrollPane, BorderLayout.CENTER);
```

```
    }

    // Afficher la fenêtre d'ajout d'un propriétaire et véhicule
    private void showAddDialog() {
        JTextField idProprietaireField = new JTextField();
        JTextField idVehiculeField = new JTextField();  // Champ pour l'ID du véhicule
        JTextField dateDebutField = new JTextField();
        JTextField dateFinField = new JTextField();

        Object[] message = {
            "ID Propriétaire:", idProprietaireField,
            "ID Véhicule:", idVehiculeField,  // Utiliser l'ID du véhicule
            "Date Début (YYYY-MM-DD):", dateDebutField,
            "Date Fin (YYYY-MM-DD):", dateFinField
        };

        int option = JOptionPane.showConfirmDialog(this, message, "Ajouter une possession",
JOptionPane.OK_CANCEL_OPTION);

        if (option == JOptionPane.OK_OPTION) {
            // Appeler la méthode pour ajouter une nouvelle possession
            possederController.addPosseder(idProprietaireField.getText(), idVehiculeField.getText(),
                    dateDebutField.getText(), dateFinField.getText());
            refreshTable();
        }
    }

    // Afficher la fenêtre de mise à jour d'une possession
    private void showUpdateDialog() {
        int selectedRow = possessionTable.getSelectedRow();
        if (selectedRow != -1) {
            String idProprietaire = possessionTable.getValueAt(selectedRow, 0).toString();
            String idVehicule = possessionTable.getValueAt(selectedRow, 1).toString();
            String dateDebut = possessionTable.getValueAt(selectedRow, 2).toString();
            String dateFin = possessionTable.getValueAt(selectedRow, 3).toString();

            JTextField idProprietaireField = new JTextField(idProprietaire);
            JTextField idVehiculeField = new JTextField(idVehicule);  // Afficher l'ID du véhicule
            JTextField dateDebutField = new JTextField(dateDebut);
            JTextField dateFinField = new JTextField(dateFin);

            Object[] message = {
                "ID Propriétaire:", idProprietaireField,
                "ID Véhicule:", idVehiculeField,
                "Date Début (YYYY-MM-DD):", dateDebutField,
                "Date Fin (YYYY-MM-DD):", dateFinField
            };

            int option = JOptionPane.showConfirmDialog(this, message, "Mettre à jour une possession",
JOptionPane.OK_CANCEL_OPTION);

            if (option == JOptionPane.OK_OPTION) {
                // Appeler la méthode pour mettre à jour la possession
                int idProprietaireInt = Integer.parseInt(idProprietaire); // convertir l'ID en int
```

```
            possederController.updatePosseder(idProprietaireInt, idVehiculeField.getText(),
                dateDebutField.getText(), dateFinField.getText());
            refreshTable();
          }
        } else {
          JOptionPane.showMessageDialog(this, "Veuillez sélectionner une ligne à mettre à jour.", "Erreur",
JOptionPane.ERROR_MESSAGE);
        }
      }

      // Supprimer la possession sélectionnée
      private void deletePossession() {
        int selectedRow = possessionTable.getSelectedRow();
        if (selectedRow != -1) {
          int idProprietaire = Integer.parseInt(possessionTable.getValueAt(selectedRow, 0).toString());
          possederController.deletePosseder(idProprietaire);
          refreshTable();
        } else {
          JOptionPane.showMessageDialog(this, "Veuillez sélectionner une ligne à supprimer.", "Erreur",
JOptionPane.ERROR_MESSAGE);
        }
      }

      // Recharger les données du tableau
      private void refreshTable() {
        // Réinitialiser le tableau en supprimant l'ancien
        remove(getContentPane().getComponent(1)); // Supprimer le JScrollPane contenant la table
        // Mettre à jour les données du tableau avec les dernières informations
        initPossessionTable();
        revalidate();  // Rafraîchir la mise en page du composant
        repaint();     // Redessiner l'écran
      }

      public static void main(String[] args) {
        new PossederView();
      }
}
```