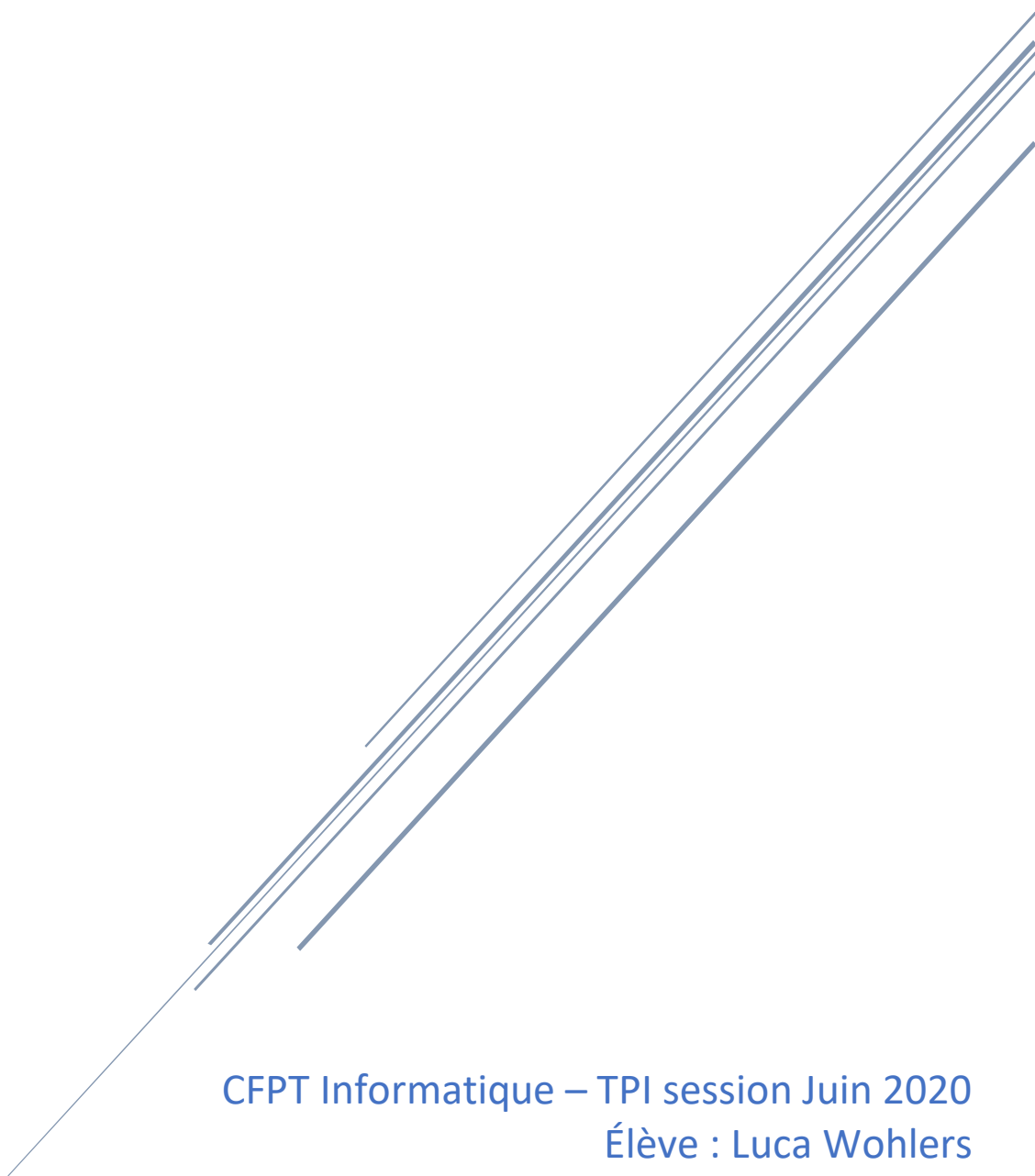


MANUEL TECHNIQUE

Moto Care C#



CFPT Informatique – TPI session Juin 2020

Élève : Luca Wohlers

Maîtresse d'apprentissage : Anne Terrier

Experts : Borys Folomietow et Alain Fontanini

1 Table des matières

2	Table des versions	3
3	Introduction	3
4	Rappel de l'énoncé	3
4.1	Organisation	3
4.2	Livrables.....	3
4.3	Matériel et logiciels à disposition	4
4.4	Description de l'application.....	4
5	Méthodologie.....	6
6	Planification	8
7	Interfaces	11
7.1	Liste de fonctionnalités disponibles.....	11
7.1.1	Véhicules	11
7.1.2	Trajets.....	11
7.1.3	Entretiens/Maintenances.....	12
7.1.4	Points d'intérêts.....	12
7.2	Présentation de l'interface.....	12
7.2.1	L'icône	12
7.2.2	Image de moto.....	13
7.2.3	La fiche frmMain	13
7.2.4	La fiche frmAjoutVehicule	19
7.2.5	La fiche frmModifierVehicule	20
7.2.6	La fiche frmAjoutTrajet	21
7.2.7	La fiche frmModifierTrajet	22
7.2.8	La fiche frmAjoutEntretien	23
7.2.9	La fiche frmModifierEntretien	23
7.2.10	Les labels informatifs	24
7.2.11	Les messages d'informations	25
7.2.12	Mesures de sécurité mises en place	26
8	Généralités concernant l'implémentation	27
8.1	Base de données	27

8.1.1	Contenu	27
8.1.2	Ajout au projet.....	29
8.2	Classes	30
8.2.1	Diagramme de classe	30
8.2.2	BD.cs	31
8.2.3	Vehicule.cs.....	33
8.2.4	Trajet.cs	33
8.2.5	Entretien.cs.....	34
8.2.6	PointInteret.cs	34
8.3	Carte du monde	34
8.4	Exemple de code.....	35
8.4.1	Base de données.....	35
8.4.2	GMap.....	36
8.5	Protocole de Tests.....	37
8.6	Arborescence du Projet.....	44
9	Conclusion.....	45
9.1	Difficultés rencontrées	45
9.2	Variantes de solutions et choix.....	45
9.3	Améliorations possibles	45
9.4	Bilan personnel	45
9.5	Remerciements.....	46
10	Bibliographie	47
11	Table des illustrations.....	48
12	Annexes.....	49

2 Table des versions

N° de version	Date	Auteur	Changements apportés
1.0	09.06.2020	Luca Wohlers <luca.whlrs@eduge.ch>	Version finale du document pour le rendu TPI

3 Introduction

Ce document est un rapport présentant différents aspects de la conception du projet « Moto Care ». Ce projet a été réalisé dans le cadre du « Travail pratique individuel (TPI) » durant fin mai, début juin. Il a pour but de valider mes compétences acquises pendant la formation Informaticien CFC effectuée à l'école d'informatique du CFPT au Petit-Lancy.

Moto Care est une application C# qui permet à l'utilisateur de planifier des entretiens pour sa ou ses motos, d'indiquer les trajets effectués avec cette dernière et de placer sur une carte du monde les lieux favoris de l'utilisateur ou les endroits qu'il souhaite visiter.

4 Rappel de l'énoncé

4.1 Organisation

Élève
Luca Wohlers <luca.whlrs@eduge.ch>

Maîtresse d'apprentissage
Anne Terrier <anne.terrier@edu.ge.ch>

Experts	
Borys Folomietow <borys@folomietow.ch>	Alain Fontanini <alain.fontanini@outlook.com>

4.2 Livrables

- Planning prévisionnel
- Rapport du projet
- Manuel utilisateur

- Journal de bord

4.3 Matériel et logiciels à disposition

Étant donné que le TPI cette année s'effectue lors de la pandémie du Covid-19 le matériel est mon matériel personnel mais avec les outils utilisés à l'école.

- Un PC portable avec Windows 10, 2 écrans
- Visual Studio Community 2019
- La suite Office
- Git avec dépôt sur Github

4.4 Description de l'application

Moto Care est une application C# destinée à un usage privé. Toute personne possédant l'application peut ajouter, modifier ou supprimer son véhicule. Il peut ajouter, modifier ou supprimer les trajets effectués. Ensuite, il peut ajouter, modifier ou supprimer des entretiens à effectuer sur le véhicule. Enfin, il y a une carte du monde pour ajouter les points d'intérêts de l'utilisateur. Il n'y a pas de connexion nécessaire pour gérer les entretiens etc.

Les points suivants doivent être respectés :

- La solution est fonctionnelle
La solution finale fonctionne correctement.
- La base de données est correctement implémentée. Il y a au moins 1 véhicule, 5 maintenances, 2 trajets et 1 point d'intérêt
Dans la base de données toutes les tables doivent être remplies. Le nombre d'enregistrement correspond plus ou moins à l'importance de la table. Au moins, 1 enregistrement pour la table véhicule, 5 enregistrements pour celle des maintenances, 2 pour celle des trajets et 1 pour les points d'intérêts.
- L'utilisateur peut consulter son carnet d'entretien
Dans un des onglets l'utilisateur peut consulter tous les entretiens à venir ou ceux déjà effectués. Il a aussi la possibilité de cocher les entretiens qui viennent d'être faits.
- L'utilisateur peut ajouter/supprimer/modifier les entretiens pour un véhicule
L'utilisateur voit la liste des entretiens et il peut pour chaque entretien, le modifier ce qui va modifier les valeurs dans la base de données. Il peut le supprimer, ce qui enlèvera l'enregistrement de la base de données. Finalement il peut créer un nouvel entretien.

- L'utilisateur peut ajouter/supprimer/modifier un véhicule
L'utilisateur peut ajouter un nouveau véhicule en renseignant les champs nécessaires. Il peut le modifier, les champs sont préremplis avec les valeurs actuellement dans la base de données. Il peut également supprimer un véhicule de la base de données ce qui supprimera aussi tous les autres enregistrements liés à ce véhicule. Tous les trajets et entretiens appartenant au véhicule supprimé, seront également supprimés.
- L'utilisateur peut ajouter/supprimer/modifier un trajet
L'utilisateur peut ajouter un trajet qu'il a effectué avec un véhicule. Il a la possibilité de le modifier en ayant les champs préremplis avec les valeurs actuellement dans la base de données. Il peut aussi le supprimer définitivement. Tout ajout/suppression/modification de la distance du trajet engendre une modification du kilométrage du véhicule.
- La solution respecte une implémentation MV
La façon dont le code a été écrit est en Modèle-Vue, c'est-à-dire qu'on sépare le modèle de la vue. Tout ce qui concerne l'affichage s'effectue à un endroit et ce qui concerne le programme se fait de l'autre côté.

5 Méthodologie

Pour planifier mon projet de TPI, je me suis basé sur la méthode en 6 étapes.

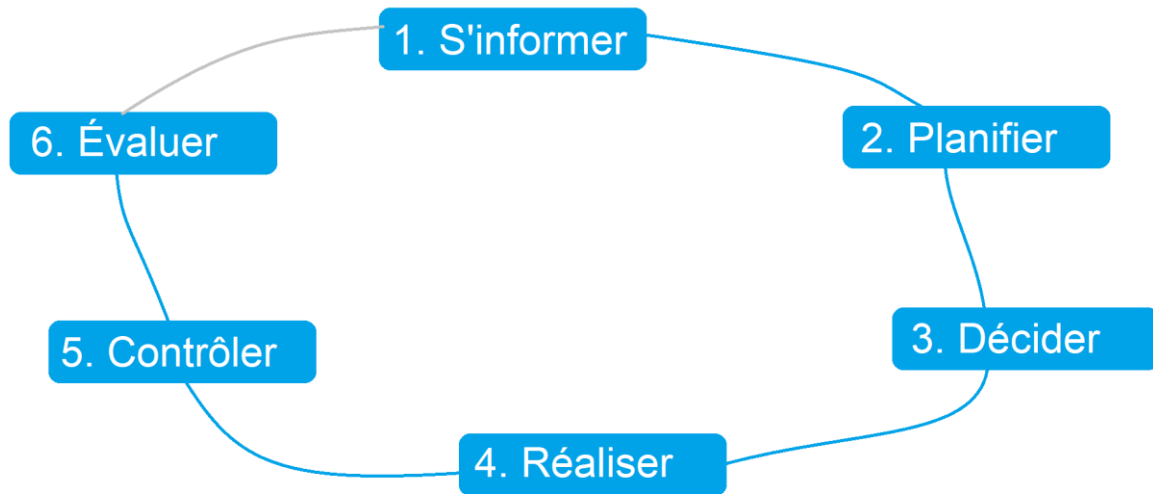


Figure 1 - Méthodologie

1. S'informer

La toute première étape de mon projet était la lecture en profondeur de mon énoncé pour comprendre toutes les fonctionnalités qu'il était nécessaire d'implémenter. J'ai également demandé à ma formatrice quelques clarifications sur certains détails lorsque c'était nécessaire.

2. Planifier

Dès le début du projet, j'ai préparé un planning de travail pour savoir ce qu'il fallait faire et quand le faire. J'ai donc séparé mon travail en sous-tâches importantes. Une fois les tâches importantes isolées j'ai créé un diagramme de Gantt pour visualiser au quotidien ma progression dans le travail ainsi que les différences entre ma planification et mon avancement effectif.

3. Décider

Au cours de l'avancement de mon travail, j'ai dû prendre de nombreuses décisions sur la manière de le réaliser. Lorsque je faisais des choix que je jugeais importants pour le projet, j'en parlais dans mon journal de bord en expliquant les raisons qui m'ont poussé à faire ce choix en question.

4. Réaliser

Une fois les bonnes décisions prises, je les implémentais dans le code.

5. Contrôler

À chaque fois que je terminais une fonctionnalité, je la testais dans différents cas d'usage pour être sûr qu'elle fonctionnait comme prévu.

6. Évaluer

La dernière étape de la méthodologie en 6 étapes est l'évaluation, pour faire une rétrospective de ce que j'ai fait et avoir un regard critique sur ce qui pourrait être amélioré. Pour ce faire, j'ai écrit des bilans journaliers à chaque fin de journée de travail dans mon journal de bord.

6 Planification

Pour choisir dans quel ordre effectuer mon projet j'ai tout d'abord relevé toutes les tâches importantes :

- Tout d'abord, ce qui concerne la base de données
 - Création de la base de données
 - Insertion de données
- Ensuite, ce qui est en lien avec le design
 - Création de la page principale
 - Création des 4 onglets
 - Trajet
 - Carnet
 - Gestion
 - Points d'intérêts
 - Affichage de la carte du monde
- Puis, le développement
 - Gestion des véhicules
 - Gestion des trajets
 - Modification du kilométrage du véhicule en fonction des trajets
 - Validation des entretiens effectués
 - Gestion des entretiens
 - Consultation des points d'intérêts sur la carte
 - Ajout de points d'intérêts sur la carte

Après avoir analysé les tâches importantes j'ai créé un diagramme de Gantt en les positionnant dans un ordre logique afin de voir plus grossièrement ce qu'il fallait que je fasse. Ensuite, j'ai estimé la durée que chaque tâche devrait me prendre pour ne pas prendre de retard. Le résultat est le suivant :

Planning Prévisionnel

Tâches à réaliser	1er jour		2e jour		3e jour		4e jour		5e jour		6e jour		7e jour		8e jour		9e jour		10e jour		11e jour	
	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.
PREPARATION																						
Lecture, analyse de l'énoncé																						
Définition des tâches et du planning prévisionnel																						
Rédactions des scénarios de tests																						
BASE DE DONNÉES																						
Création de la base de données																						
Insertion de données dans la base																						
DESIGN [FRONT-END]																						
Création de la page d'accueil avec les véhicules																						
Création des 4 onglets (Trajets, Carnet, Gestion, Points d'intérêts)																						
Afficher une carte du monde pour les points d'intérêts																						
DEVELOPPEMENT [BACK-END]																						
Gestion des véhicules (CRUD)																						
Gestion des trajets des véhicules (CRUD)																						
Modification du kilométrage du véhicule en fonction des trajets																						
Validation des entretiens effectués																						
Gestion des entretiens (CRUD)																						
Consultation des points d'intérêts sur la carte																						
Ajout de point d'intérêts à l'aide de la carte																						
ADMINISTRATION																						
Conception du document technique																						
Conception du manuel utilisateur																						
Debugging, tests, résolutions																						
Finitions, optimisations, révisions																						

Figure 2 - Planning Prévisionnel

Le diagramme ci-dessus est le planning prévisionnel, c'est-à-dire qu'il s'agit du diagramme que j'ai créé avant même de commencer mon travail. Mais au fur et à mesure où j'avais dans mon travail certaines tâches n'ont pas été effectuées au moment où je l'avais prévu. Par exemple j'ai fini le CRUD des trajets une demi-journée en avance. J'ai donc commencé la modification du kilométrage du véhicule pas le 5^{ème} jour dans l'après-midi mais le matin. Cela m'a aussi pris plus de temps que prévu car non seulement il m'a fallu me familiariser avec les checkBox et le dataGridView mais aussi réfléchir à une interface optimum pour le carnet d'entretien.

Ensuite, je me suis rendu compte que pour faire la validation des entretiens il fallait tout d'abord que je commence le CRUD des entretiens, au moins le « Read », autrement dit la lecture des entretiens à partir de la base de données.

Mais encore, la consultation des points d'intérêts et leur ajout a été nettement plus rapide que ce que j'avais défini. J'ai terminé les deux durant l'après-midi de la 8^{ème} journée et non pas au cours de la 9^{ème}.

Finalement, en ce qui concerne la documentation, je n'ai pas travaillé sur le manuel technique et utilisateur chaque jour, j'ai cependant évidemment mis chaque jour mon journal de bord à jour.

Voici donc le planning effectif de mon travail :

Planning Effectif

Tâches à réaliser	1er jour		2e jour		3e jour		4e jour		5e jour		6e jour		7e jour		8e jour		9e jour		10e jour		11e jour	
	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.	Mat.	Ap.
PREPARATION																						
Lecture, analyse de l'énoncé																						
Définition des tâches et du planning provisionnel																						
Rédactions des scénarios de tests																						
BASE DE DONNÉES																						
Création de la base de données																						
Insertion de données dans la base																						
DESIGN [FRONT-END]																						
Création de la page d'accueil avec les véhicules																						
Création des 4 onglets (Trajets, Carnet, Gestion, Points d'intérêts)																						
Afficher une carte du monde pour les points d'intérêts																						
DEVELOPPEMENT [BACK-END]																						
Gestion des véhicules (CRUD)																						
Gestion des trajets des véhicules (CRUD)																						
Modification du kilométrage du véhicule en fonction des trajets																						
Validation des entretiens effectués																						
Gestion des entretiens (CRUD)																						
Consultation des points d'intérêts sur la carte																						
Ajout de point d'intérêts à l'aide de la carte																						
ADMINISTRATION																						
Conception du document technique																						
Conception du manuel utilisateur																						
Debugging, tests, résolutions																						
Finitions, optimisations, révisions																						

Figure 3 - Planning Effectif

7 Interfaces

Cette section de la documentation traite la partie visible de l'application, vue par l'utilisateur.

Tout d'abord, sont énumérées les fonctionnalités disponibles dans l'application. Ensuite viennent les présentations des fenêtres de l'interface utilisateur.

7.1 Liste de fonctionnalités disponibles

7.1.1 Véhicules

- Ajouter un nouveau véhicule
 - Nom
 - Description
 - Nb. de kilomètre lors de l'achat
 - Photo
- Modifier un véhicule existant
 - Nom
 - Description
 - Nb. de kilomètre lors de l'achat
 - Photo
- Supprimer un véhicule existant
- Choisir quel véhicule afficher
 - Son nom s'affiche
 - Sa description s'affiche
 - Son nombre de kilomètre lors de l'achat s'affiche
 - Son nombre de kilomètre actuelle s'affiche
 - Sa photo s'affiche

7.1.2 Trajets

- Ajouter un nouveau trajet
 - Départ
 - Arrivée
 - Distance parcourue (en km)
 - Date lors du trajet
- Modifier un trajet existant
 - Départ
 - Arrivée
 - Distance
 - Date du trajet
- Supprimer un trajet existant

7.1.3 Entretiens/Maintenances

- Cocher ses entretiens pour indiquer qu'ils ont été faits
- Ajouter un nouvel entretien
 - Description
 - Le kilométrage auquel faudra effectuer le premier entretien en qu'on ajoute
 - La fréquence en kilomètre à laquelle faudra répéter l'entretien
- Modifier un entretien, seulement ceux pas encore effectués
 - Description
 - Fréquence
- Supprimer un entretien

7.1.4 Points d'intérêts

- Ajouter un point d'intérêt
 - Nom
 - Lieux déjà visité
 - Description
- Visualiser sur la carte tous les points d'intérêts

7.2 Présentation de l'interface

L'interface utilisateur comprend un total de sept formulaires. Une seule d'entre elle est la principale et c'est sur celle-ci qu'on voit toutes les informations. Les six autres sont pour modifier et ajouter de nouveaux éléments, comme les véhicules, les trajets et les entretiens.

7.2.1 L'icône



Figure 4 - Icône application

Cette icône récupérée à l'URL suivante n'était pas protégée par d'éventuels droits d'auteurs :

<http://www.iconarchive.com/download/i91836/icons8/windows-8/Transport-Motorcycle.ico>

Une fois l'image téléchargée elle a été ajoutée à l'application par le biais des propriétés du projet dans Visual Studio 2019. Il s'agit simplement d'une image représentant un dessin tout noir d'une moto. L'extension du fichier de l'icône est « .ico ».

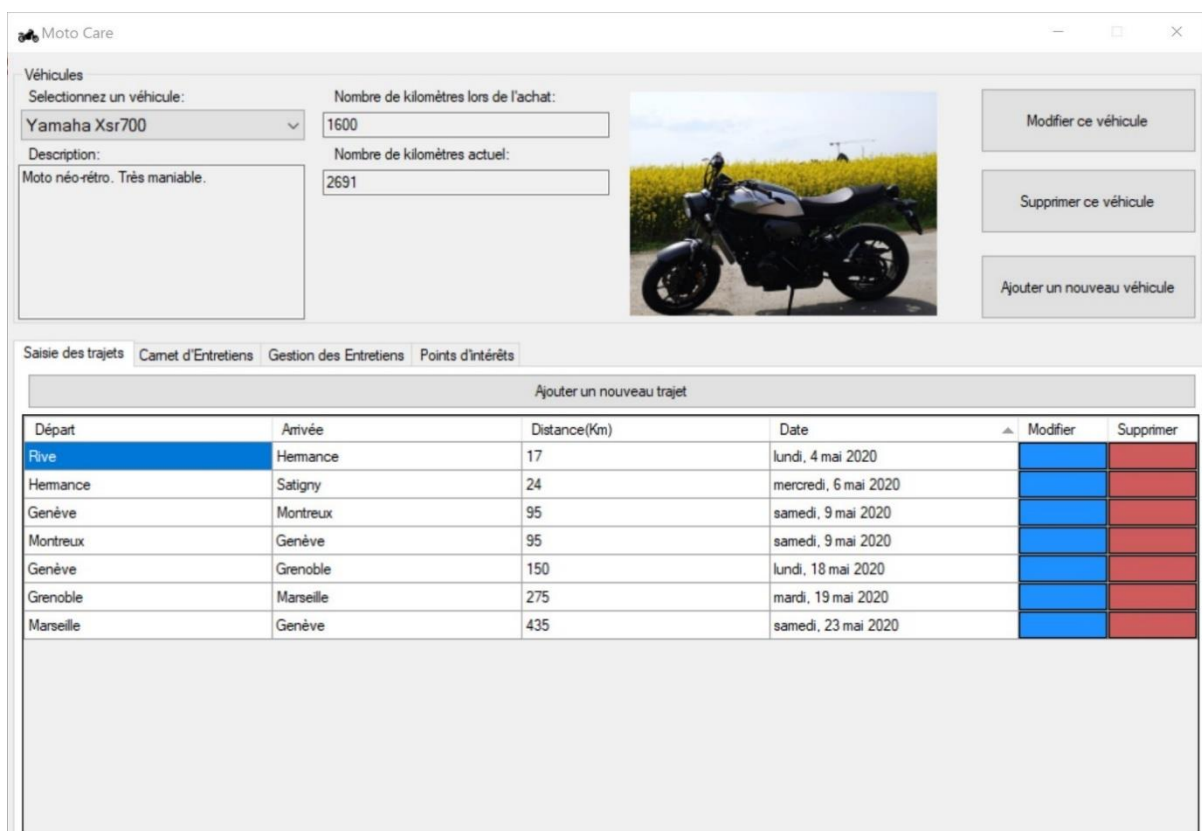
7.2.2 Image de moto



Figure 5 - Photo moto

Cette photo de moto n'est pas soumise à de droits d'auteurs car il s'agit d'une image personnelle. Elle apparaît dans frmMain grâce à la lecture de l'image se trouvant dans la base de données. Elle apparaît aussi dans frmModifierVehicule lors de la modification du véhicule pour avoir un aperçu de l'image sélectionnée.

7.2.3 La fiche frmMain



Départ	Arrivée	Distance(Km)	Date	Modifier	Supprimer
Rive	Hemance	17	lundi, 4 mai 2020		
Hemance	Satigny	24	mercredi, 6 mai 2020		
Genève	Montreux	95	samedi, 9 mai 2020		
Montreux	Genève	95	samedi, 9 mai 2020		
Genève	Grenoble	150	lundi, 18 mai 2020		
Grenoble	Marseille	275	mardi, 19 mai 2020		
Marseille	Genève	435	samedi, 23 mai 2020		

Figure 6 - La fiche frmMain

Cette fiche est la fiche principale de l'application. L'utilisateur a accès à tout à partir de cette fenêtre. Il peut sélectionner à l'aide du composant comboBox le véhicule sur lequel il compte faire des modifications. Il peut voir dans la section Véhicules se trouvant dans le

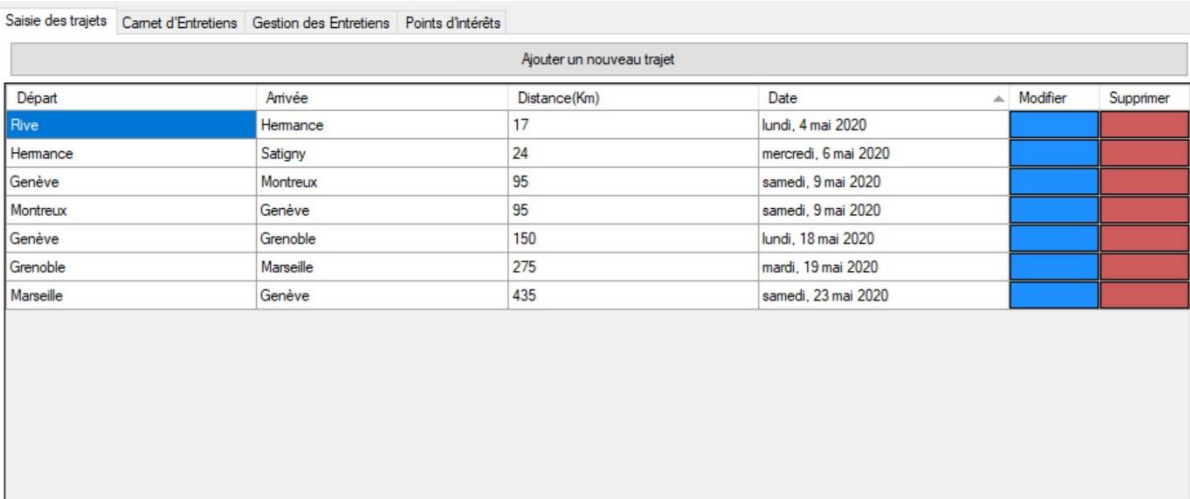
premier tiers de la fenêtre, toutes les informations sur son véhicule. Les informations visibles du véhicule sont :

- Son nom, à l'aide d'un ComboBox qui permet la sélection d'un véhicule parmi ceux qu'on possède.
- Sa « Description » à l'aide d'un TextBox en lecture seule.
- Son « Nombre de kilomètres lors de l'achat » grâce à un TextBox en lecture seule.
- Puis finalement son « Nombre de kilomètres actuel » grâce à un TextBox en lecture seule. Cette valeur est calculé avec tous les trajets du véhicule et son kilométrage lors de l'achat.

La section véhicule est séparée du reste grâce à un composant GroupBox. L'utilisateur peut aussi dans cette section, voir les différentes actions qu'il peut effectuer sur son véhicule grâce aux différents boutons, comme par exemple le modifier, le supprimer ou encore ajouter un nouveau véhicule.

Dans la partie inférieure de cette fenêtre, l'utilisateur peut gérer plusieurs choses. Tel que les trajets qu'il a effectué avec le véhicule actuellement sélectionné, les entretiens qu'il a effectué sur son véhicule ou ceux qui sont à venir et les points d'intérêts de l'utilisateur, indépendant du véhicule sélectionné. Tout ceci est séparé à l'aide d'un TabControl qui est composé de différents onglets.

7.2.3.1 Onglet Saisie des trajets



Ajouter un nouveau trajet					
Départ	Arrivée	Distance(Km)	Date	Modifier	Supprimer
Rive	Hernance	17	lundi, 4 mai 2020		
Hernance	Satigny	24	mercredi, 6 mai 2020		
Genève	Montreux	95	samedi, 9 mai 2020		
Montreux	Genève	95	samedi, 9 mai 2020		
Genève	Grenoble	150	lundi, 18 mai 2020		
Grenoble	Marseille	275	mardi, 19 mai 2020		
Marseille	Genève	435	samedi, 23 mai 2020		

Figure 7 - Onglet Saisie des trajets

Dans cet onglet apparaît la liste des trajets du véhicule sélectionné. Chaque trajet est affiché dans une ligne du composant DataGridView. La première colonne est le départ du trajet, ensuite son lieu d'arrivée, la distance parcourue entre le départ et l'arrivée, la date lorsque le trajet a été effectué, finalement un bouton bleu pour modifier le trajet et un rouge pour le supprimer. La liste des trajets est triée par leur date, le plus récent en dernier dans le tableau. Les colonnes n'ont pas toutes le même type de donnée :

- La colonne Départ
 - Est de type DataGridViewTextBoxColumn.
 - Sans format spécial.
- La colonne Arrivée
 - Est de type DataGridViewTextBoxColumn.
 - Sans format spécial.
- La colonne Distance
 - Est de type DataGridViewTextBoxColumn.
 - Sans format spécial.
- La colonne Date
 - Est de type DataGridViewTextBoxColumn.
 - Avec un format date pour afficher la date d'une telle manière. Exemple : samedi, 23 mai 2020.
- La colonne Modifier
 - Est de type DataGridViewButtonColumn.
 - Sans format spécial.
- La colonne Supprimer
 - Est de type DataGridViewButtonColumn.
 - Sans format spécial.

Un composant de type Button se trouve en haut de l'onglet pour ajouter un nouveau trajet.

L'utilisateur peut à partir de cet onglet :

- Ajouter un trajet
- Modifier un trajet
- Supprimer un trajet

7.2.3.2 Onglet Carnet d'Entretiens

Effectué	Description	Date Dernière Maintenance	Km Dernière Maintenance	Fréquence	Prochaine Maintenance dans (km):
<input type="checkbox"/>	Vidange			8000	5 309
<input type="checkbox"/>	Graissage chaîne	vendredi, 5 juin 2020	2011	750	70
<input checked="" type="checkbox"/>	Service des 2000Km			0	--
<input checked="" type="checkbox"/>	Graissage chaîne			750	--

Figure 8 - Onglet Carnet d'Entretiens (Jaune-Vert)

Dans cet onglet, l'utilisateur peut voir tous les entretiens effectués et ceux à venir. Les entretiens effectués sont affichés en vert, ceux qui doivent être effectués dans les 100 prochains kilomètres sont en jaune comme affiché sur l'image ci-dessus. Puis ceux en retard ou qui doivent être effectués absolument sont en rouge comme sur l'image ci-dessous.

Saisie des trajets					
Carnet d'Entretiens					
Gestion des Entretiens					
Points d'intérêts					
Effectué	Description	Date Dernière Maintenance	Km Dernière Maintenance	Fréquence	Prochaine Maintenance dans (km):
<input type="checkbox"/>	Vidange			8000	5 239
<input checked="" type="checkbox"/>	Graissage chaîne	vendredi, 5 juin 2020	2011	750	0
<input checked="" type="checkbox"/>	Service des 2000Km			0	--
<input checked="" type="checkbox"/>	Graissage chaîne			750	--

Figure 9 - Onglet Carnet d'Entretiens (Rouge-Vert)

Cet onglet est composé d'un seul composant, un DataGridView. Il contient six différentes colonnes visibles, la première est une pour savoir si l'entretien a été effectué ou non, la suivante décrit de quel entretien il s'agit, celle d'après indique la date de la dernière même maintenance s'il y en a une, la suivante indique le nombre de kilomètre de la moto lors de la même dernière maintenance, puis la prochaine indique la fréquence de la maintenance, finalement la dernière colonne montre dans combien de kilomètre il faudra effectuer la maintenance en question. Dans ce dataGridView il y a aussi une colonne cachée en première position pour enregistrer l'id de l'entretien pour savoir facilement quel entretien on coche comme étant effectué. Cette liste d'entretien est trié par le km de la prochaine maintenance, ceux étant fait se retrouvent plus bas dans la liste et le reste est trié en décroissant. Voici les différentes données en fonction des colonnes.

- La colonne Id
 - Est de type DataGridViewTextBoxColumn
 - Sans format spécial mais caché
- La colonne Effectué
 - Est de type DataGridViewCheckBoxColumn
 - Sans format spécial
- La colonne Description
 - Est de type DataGridViewTextBoxColumn
 - Sans format spécial
- La colonne Date Dernière Maintenance
 - Est de type DataGridViewTextBoxColumn
 - Avec un format date
- La colonne Km Dernière Maintenance
 - Est de type DataGridViewTextBoxColumn
 - Sans format spécial
- La colonne Fréquence
 - Est de type DataGridViewTextBoxColumn
 - Sans format spécial
- La colonne Prochaine Maintenance dans (km)
 - Est de type DataGridViewTextBoxColumn
 - Avec un format numérique

L'utilisateur peut à partir de cet onglet :

- Coché un trajet comme étant effectué (qui créera automatiquement le prochain entretien)

7.2.3.3 Onglet Gestion des Entretien

Ajouter un nouvel Entretien					
Effectué	Description	Km lors de l'entretien	Fréquence	Modifier	Supprimer
<input checked="" type="checkbox"/>	Service des 2000Km	2000	0		
<input checked="" type="checkbox"/>	Graissage chaîne	2000	750		
<input type="checkbox"/>	Graissage chaîne	2761	750		
<input type="checkbox"/>	Vidange	8000	8000		

Figure 10 - Onglet Gestion des Entretien

Dans cet onglet, l'utilisateur peut ajouter, modifier ou supprimer un entretien. L'onglet contient deux composants, un de type Button pour ajouter un entretien et un DataGridView pour afficher la liste des entretiens et en fonction de la colonne il est possible d'en modifier ou supprimer un. Il faut savoir qu'il est possible de modifier seulement un entretien qui n'a pas encore été effectué.

Le DataGridView est trié dans l'ordre croissant du kilométrage du véhicule lors de l'entretien. Il est composé de six différentes colonnes visibles, la première est celle intitulé Effectué pour savoir si l'entretiens a déjà été effectué, la deuxième contient la description de l'entretien, la troisième indique le kilométrage lorsque l'entretien doit ou a été effectué, la quatrième montre la fréquence de l'entretien, la suivante contient un bouton bleu pour modifier l'entretien et la dernière un bouton rouge pour le supprimer. Voici les différentes données en fonction des colonnes :

- La colonne Id
 - Est de type DataGridViewTextBoxColumn.
 - Sans format spécial mais caché.
- La colonne Effectué
 - Est de type DataGridViewCheckBoxColumn.
 - Sans format spécial.
- La colonne Description
 - Est de type DataGridViewTextBoxColumn.
 - Sans format spécial.
- La colonne Km lors de l'entretien

- Est de type DataGridViewTextBoxColumn.
 - Sans format spécial.
- La colonne Fréquence
 - Est de type DataGridViewTextBoxColumn.
 - Sans format spécial.
- La colonne Modifier
 - Est de type DataGridViewButtonColumn.
 - Sans format spécial.
- La colonne Supprimer
 - Est de type DataGridViewButtonColumn.
 - Sans format spécial.

L'utilisateur peut à partir de cet onglet :

- Ajouter un entretien
- Modifier un entretien
- Supprimer un entretien

7.2.3.4 Onglet Points d'intérêts

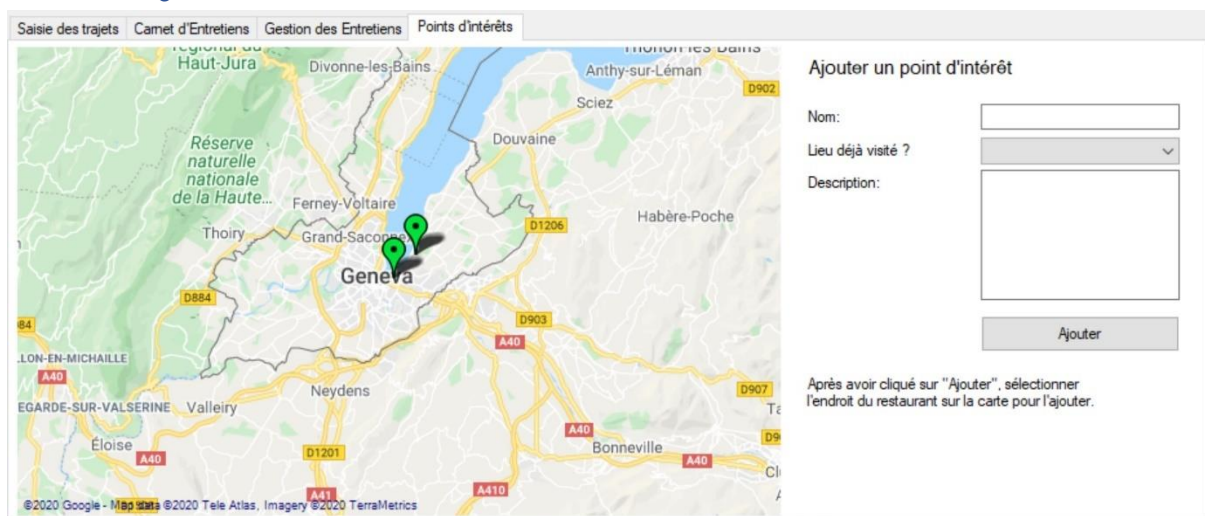


Figure 11 - Onglet Points d'intérêts

Dans cet onglet, l'utilisateur peut voir sur la carte tous les points d'intérêts qu'il a ajoutés. Ces points d'intérêts peuvent être des lieux où l'utilisateur a aimé se rendre ou encore des lieux où il aimerait aller. Il peut toujours ajouter et voir ses points d'intérêts même s'il ne possède pas de véhicule car ils sont dépendant de l'utilisateur et non pas des véhicules.

Dans cet onglet il y a plusieurs contrôles, sur la gauche se trouve un composant GMapControl inséré à l'aide d'une librairie GMap.Net.Windows. Sur la droite se trouvent les composants pour l'ajout d'un marqueur. Un TextBox pour le nom que l'utilisateur souhaite donner au marqueur, un ComboBox pour indiquer si oui on a déjà visité le lieu ou non, puis un autre

TextBox multiligne pour laisser à l'utilisateur de la place pour écrire une description du lieu.
Un composant de type Button pour que l'utilisateur puisse enregistrer ce qu'il a donné comme caractéristique au marqueur.

7.2.4 La fiche frmAjoutVehicule

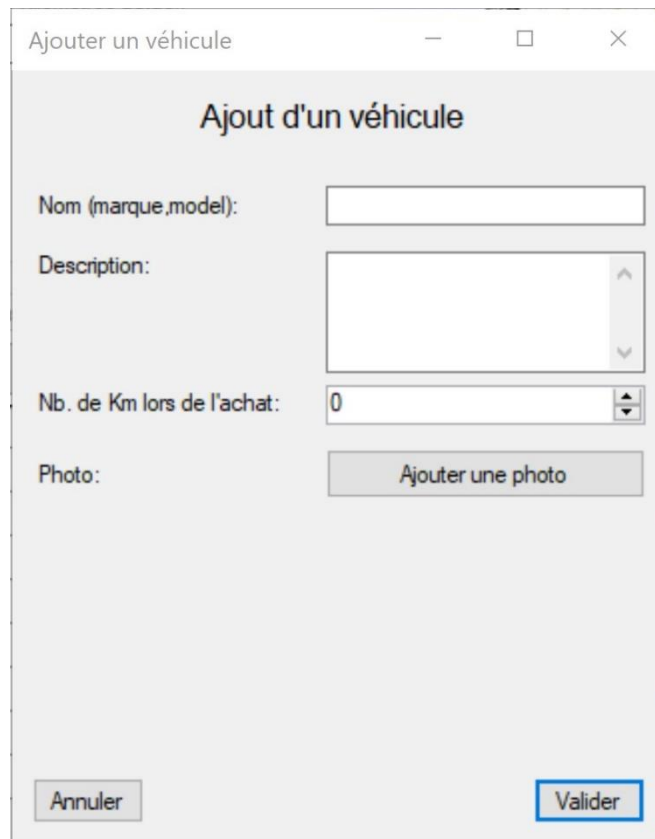


Figure 12 - La fiche frmAjoutVehicule

Cette fiche s'ouvre lorsqu'on souhaite ajouter un véhicule dans la section supérieur de l'interface principale.

Les champs :

- Dans « Nom », le composant correspondant est une TextBox, l'utilisateur peut entrer n'importe quel nom tant qu'il sait reconnaître le véhicule avec le nom entré.
- Dans « Description », le composant correspondant est une TextBox, l'utilisateur peut entrer ce qui lui plaît comme description de son véhicule. Comme par exemple le style, la puissance, la maniabilité, etc.
- Dans « Nb. de Km lors de l'achat », le composant correspondant est un NumericUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur doit simplement à combien de kilomètre était la moto lorsqu'il l'a achetée.
- Pour « Photo », l'utilisateur doit cliquer sur le bouton « Ajouter une photo », ce qui lui ouvrira l'explorateur de fichier en créant un nouvel objet OpenFileDialog, ce qui permet d'obtenir un fichier se trouvant sur l'ordinateur, dans ce cas-là une image. Il

peut alors sélectionner une seule photo de son choix. Une fois l'image sélectionné, elle s'affiche dans le PictureBox en dessous du bouton « Ajouter une photo » ici non visible car il n'y a pas d'image.

Une fois tous les champs remplis seulement, l'utilisateur peut cliquer sur le bouton « Valider » pour ajouter son véhicule ou sur celui « Annuler » s'il souhaite annuler l'opération.

7.2.5 La fiche frmModifierVehicule



Figure 13 - La fiche frmModifierVehicule

Cette fiche s'ouvre lorsqu'on souhaite modifier un véhicule dans la section supérieur de l'interface principale. Il faut pour cela avoir sélectionné un véhicule à partir du ComboBox se trouvant dans la même section.

Les champs sont les même que pour l'ajout d'un véhicule sauf que les champs sont préremplis avec les valeurs du véhicule que l'utilisateur est en train de modifier :

- Dans « Nom », le composant correspondant est une TextBox, l'utilisateur peut entrer n'importe quel nom tant qu'il sait reconnaître le véhicule avec le nom entré.
- Dans « Description », le composant correspondant est une TextBox, l'utilisateur peut entrer ce qui lui plaît comme description de son véhicule. Comme par exemple le style, la puissance, la maniabilité, etc.

- Dans « Nb. de Km lors de l'achat », le composant correspondant est un NumericUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur doit simplement à combien de kilomètre était la moto lorsqu'il l'a achetée.
- Pour « Photo », l'utilisateur doit cliquer sur le bouton « Ajouter une photo », ce qui lui ouvrira l'explorateur de fichier en créant un nouvel objet OpenFileDialog, ce qui permet d'obtenir un fichier se trouvant sur l'ordinateur, dans ce cas-là une image. Il peut alors sélectionner une seule photo de son choix. Une fois l'image sélectionné, elle s'affiche dans le PictureBox en dessous du bouton « Ajouter une photo » ici non visible car il n'y a pas d'image.

Une fois tous les champs remplis seulement, l'utilisateur peut cliquer sur le bouton « Valider » pour ajouter son véhicule ou sur celui « Annuler » s'il souhaite annuler l'opération.

7.2.6 La fiche frmAjoutTrajet

Pour ajouter un trajet il faut cliquer sur le bouton « Ajouter un nouveau trajet » se trouvant dans l'onglet « Saisie des trajets ».

Une fois cliqué sur le bouton cette fenêtre apparaît :

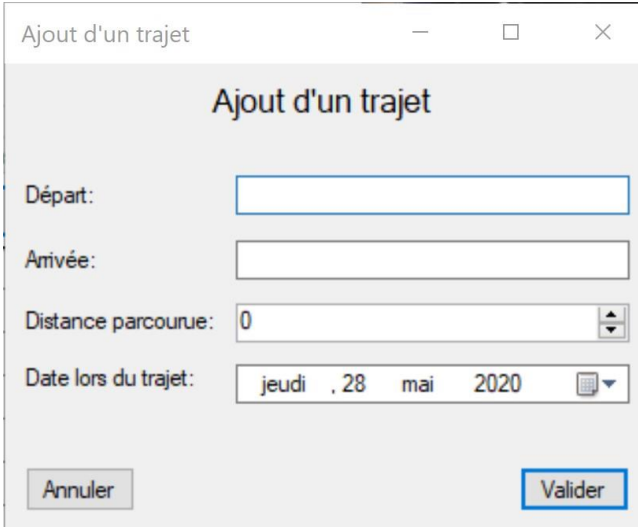


Figure 14 - La fiche frmAjoutTrajet

Les champs :

- Dans « Départ », le composant correspondant est une TextBox, l'utilisateur peut entrer n'importe quel nom de lieu d'où il est parti.
- Dans « Arrivée », le composant correspondant est une TextBox, l'utilisateur peut entrer n'importe quel nom de lieu d'où il est arrivé.
- Dans « Distance parcourue », le composant correspondant est un NumericUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur doit entrer le nombre de kilomètre qu'il a parcouru lors de ce trajet.

- Pour « Date lors du trajet », le composant correspondant est un DateTimePicker qui sert à sélectionner une date, l'utilisateur doit entrer la date à laquelle le trajet a été effectué.

Une fois tous les champs remplis seulement, l'utilisateur peut cliquer sur le bouton « Valider » pour ajouter son trajet ou sur celui nommé « Annuler » s'il souhaite annuler l'opération. La distance du trajet sera alors ajoutée au kilométrage réel de la moto.

7.2.7 La fiche frmModifierTrajet

Pour modifier un trajet il faut cliquer sur le bouton bleu se trouvant sur la même ligne que le trajet que l'on souhaite modifier. Une fois cliqué sur le bouton cette fenêtre doit apparaître :



Figure 15 - La fiche frmModifierTrajet

Les champs sont les mêmes que pour l'ajout d'un trajet sauf qu'ils sont déjà préremplis avec les valeurs actuelles du trajet que l'utilisateur est en train de modifier :

- Dans « Départ », le composant correspondant est une TextBox, l'utilisateur peut entrer n'importe quel nom de lieu d'où il est parti.
- Dans « Arrivée », le composant correspondant est une TextBox, l'utilisateur peut entrer n'importe quel nom de lieu d'où il est arrivé.
- Dans « Distance parcourue », le composant correspondant est un NumericUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur doit entrer le nombre de kilomètre qu'il a parcouru lors de ce trajet.
- Pour « Date lors du trajet », le composant correspondant est un DateTimePicker qui sert à sélectionner une date, l'utilisateur doit entrer la date à laquelle le trajet a été effectué.

Une fois tous les champs remplis seulement, l'utilisateur peut cliquer sur le bouton « Valider » pour ajouter son trajet ou sur celui nommé « Annuler » s'il souhaite annuler l'opération. La distance du trajet sera alors ajoutée au kilométrage réel de la moto.

7.2.8 La fiche frmAjoutEntretien

Pour ajouter un entretien il faut tout d’abord cliquer sur le bouton « Ajouter un nouvel Entretien » se trouvant dans l’onglet « Gestion des Entretiens », la fenêtre suivante doit ensuite s’afficher :

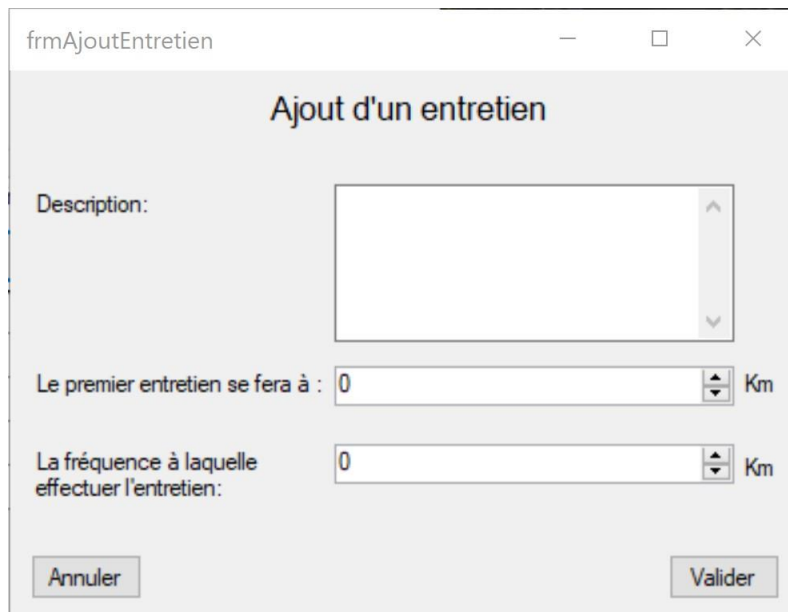


Figure 16 - La fiche frmAjoutEntretien

Les champs :

- Dans « Description », le composant correspondant est une TextBox, l'utilisateur doit entrer quel est l'entretien qui va devoir être effectué.
- Dans « Le premier entretien se fera à », le composant correspondant est un NumeriqueUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur doit indiquer à quel kilométrage de la moto le premier entretien aura lieu.
- Dans « La fréquence à laquelle effectuer l'entretien », le composant correspondant est un NumeriqueUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur doit entrer la fréquence en kilomètre à laquelle l'entretien devra être effectué à nouveau.

Une fois tous les champs remplis seulement, l'utilisateur peut cliquer sur le bouton « Valider » pour ajouter son entretien ou sur celui nommé « Annuler » s'il souhaite annuler l'opération.

7.2.9 La fiche frmModifierEntretien

Pour modifier un entretien il faut cliquer sur le bouton bleu sur la même ligne que l'entretien que l'on souhaite modifier. Il est seulement possible de modifier un entretien non effectué. Une fois cliqué sur le bouton bleu d'un entretien pas encore effectué la fenêtre frmModifierEntretien apparaît :

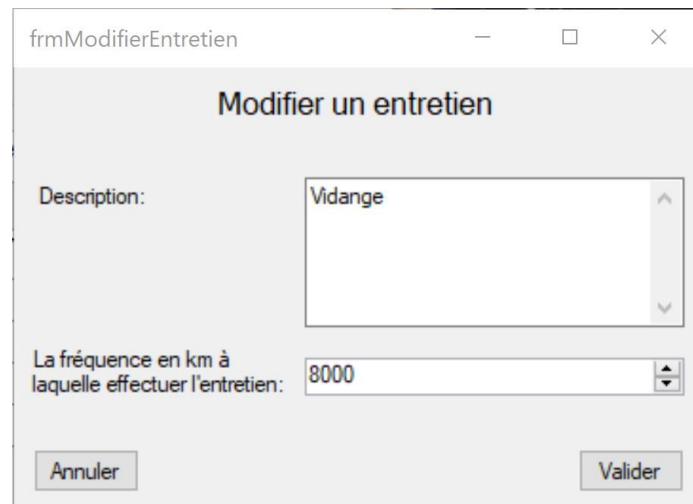
The screenshot shows a Windows-style window titled 'frmModifierEntretien'. The main heading is 'Modifier un entretien'. Below this, there are two labels: 'Description:' and 'La fréquence en km à laquelle effectuer l'entretien:'. The 'Description:' label is followed by a text box containing the word 'Vidange'. The 'La fréquence en km...' label is followed by a numeric up-down control showing the value '8000'. At the bottom of the form, there are two buttons: 'Annuler' on the left and 'Valider' on the right.

Figure 17 - La fiche frmModifierEntretien

Les champs :

- Dans « Description », le composant correspondant est un TextBox, l'utilisateur peut modifier la description de l'entretien.
- Dans « La fréquence en km à laquelle effectuer l'entretien », le composant correspondant est un NumericUpDown qui permet d'entrer seulement des valeurs numériques, l'utilisateur peut modifier la fréquence en kilomètre à laquelle l'entretien devra être effectué à nouveau.

Une fois le ou les champs modifiés, l'utilisateur peut cliquer sur le bouton « Valider » pour valider les changements ou sur celui nommé « Annuler » pour annuler l'opération.

7.2.10 Les labels informatifs

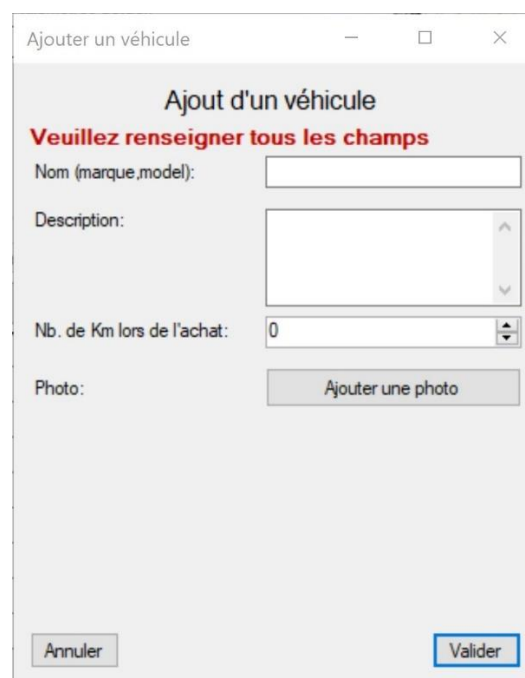
The screenshot shows a Windows-style window titled 'Ajouter un véhicule'. The main heading is 'Ajout d'un véhicule'. Below this, there is a red instruction: 'Veuillez renseigner tous les champs'. There are four labels: 'Nom (marque,model):', 'Description:', 'Nb. de Km lors de l'achat:', and 'Photo:'. The 'Nom (marque,model):' label is followed by a text box. The 'Description:' label is followed by a text box. The 'Nb. de Km lors de l'achat:' label is followed by a numeric up-down control showing the value '0'. The 'Photo:' label is followed by a button labeled 'Ajouter une photo'. At the bottom of the form, there are two buttons: 'Annuler' on the left and 'Valider' on the right.

Figure 18 - Label informatif

Pour chaque formulaire de modification ou d'ajout. Que ce soit pour les véhicules, les trajets, les entretiens ou les points d'intérêts un message rouge apparaît dans un label si les champs ne sont pas renseignés.

7.2.11 Les messages d'informations

7.2.11.1 Suppression d'un véhicule

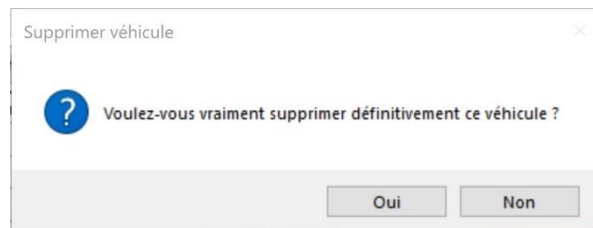


Figure 19 - Message suppression véhicule

Ce message de confirmation de suppression de véhicule apparaît lorsque l'utilisateur clique sur le bouton pour supprimer un véhicule. Il a pour but d'éviter de faire une fausse manipulation.

7.2.11.2 Suppression d'un trajet

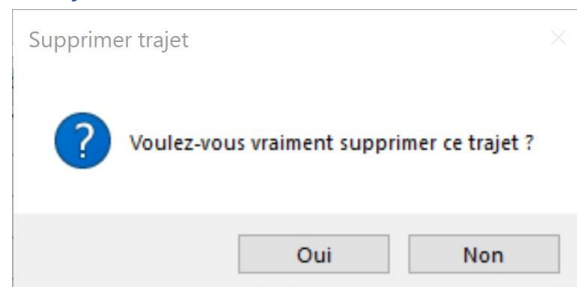


Figure 20 - Message suppression trajet

Ce message de confirmation de suppression d'un trajet apparaît lorsque l'utilisateur clique sur un bouton rouge, servant à supprimer un trajet, se trouvant dans l'onglet des trajets. Ce message a pour but d'éviter de faire une fausse manipulation.

7.2.11.3 Suppression d'un entretien

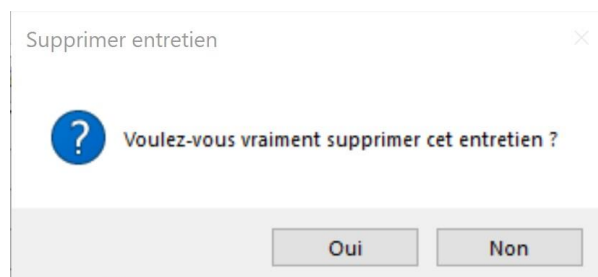


Figure 21 - Message suppression entretien

Ce message de confirmation de suppression d'un trajet apparaît lorsque l'utilisateur clique sur un bouton rouge, servant à supprimer un entretien, se trouvant dans l'onglet de gestion des trajets. Il sert à éviter les fausses manipulations.

7.2.11.4 Impossible de modifier entretien

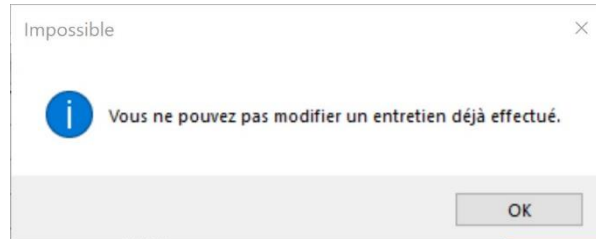


Figure 22 - Message modification entretien impossible

Ce message informatif apparaît lorsque l'utilisateur clique sur un bouton bleu pour modifier un entretien mais l'entretien a déjà été effectué, il est donc impossible de modifier cet entretien. Seul les entretiens pas encore effectués peuvent être modifiés.

7.2.11.5 Mauvais format de fichier

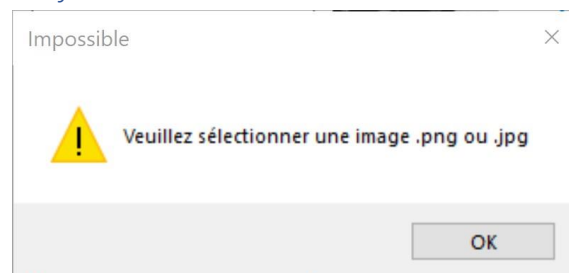


Figure 23 - Message mauvais format de fichier

Ce message apparaît lorsque l'utilisateur essaie d'ajouter un fichier autre que en .png ou .jpg lors de l'ajout ou modification d'un véhicule.

7.2.12 Mesures de sécurité mises en place

Voici une liste des contrôles mis en place afin d'éviter les mauvaises manipulations des utilisateurs et ainsi empêcher un maximum de soulèvement d'exceptions dans le code :

- Les fenêtres ne sont pas redimensionnables afin d'éviter les problèmes d'affichage
- On ne peut pas maximiser l'affichage de la fenêtre principale pour ne pas avoir de problème d'affichage
- Uniquement des chiffres entre 0 et 999999 sont possibles d'entrer dans les NumericUpDown
- Si l'utilisateur n'entre aucune valeur dans un TextBox et que c'est nécessaire un message rouge dans un label apparaît et lui dit de bien remplir tous les champs.

8 Généralités concernant l'implémentation

8.1 Base de données

8.1.1 Contenu

Pour stocker tous les véhicules, trajets, entretiens et points d'intérêts j'ai dû utiliser une base de données SQLite, comme mentionné dans le cahier des charges.

Voici le modèle conceptuel fourni dans le cahier des charges :

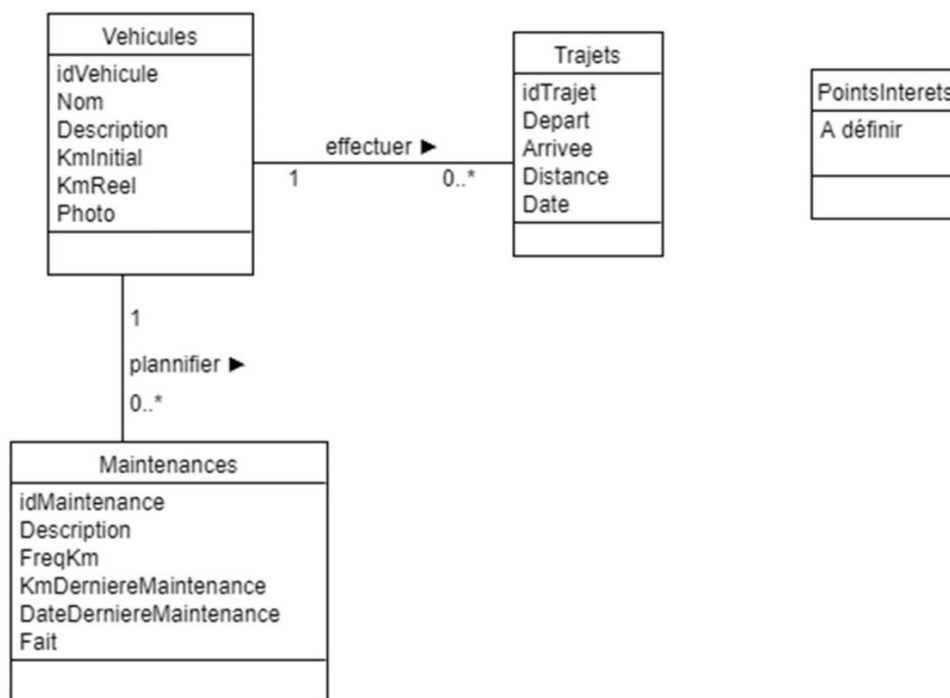


Figure 24 - Modèle conceptuel

J'ai donc créé cette base de données à l'aide de l'outil « DB Browser ». Pour la table pointsInterets il m'a fallu définir les différents champs. J'ai choisi d'ajouter les champs suivants :

- idPointInteret → pour pouvoir identifier chaque points d'intérêts
- lat → pour enregistrer la latitude du marqueur
- lng → pour enregistrer la longitude du marqueur
- nom → pour pouvoir donner un nom au marqueur. Par exemple un nom de lieu
- visite → pour indiquer lors de l'ajout si le lieu a déjà été visité ou non
- description → pour donner une description au marqueur

Voici la table :

pointInteret

Nom	Type	Clef primaire	Auto-Incrément	Non Null
idPointInteret	INTEGER	TRUE	TRUE	FALSE
lat	REAL	FALSE	FALSE	TRUE
lng	REAL	FALSE	FALSE	TRUE
nom	TEXT	FALSE	FALSE	FALSE
visite	TEXT	FALSE	FALSE	FALSE
description	TEXT	FALSE	FALSE	FALSE

Pour la table maintenance j'ai décidé d'ajouter le champ « kmPremiereMaintenance » pour être capable de différencier les tout premiers entretiens, lorsqu'on les crée, de ceux qui se répètent simplement. Cela permet lors de l'ajout d'un entretien, de définir à combien de kilomètre veut-on effectuer la première maintenance et avec quelle fréquence en kilomètre. Aussi pour le champ « fait », dans DB Browser il n'y a pas de type booléen donc j'ai fais avec du INTEGER où j'entre 0 ou 1.

Voici ce que cela donne :

maintenance				
Nom	Type	Clef primaire	Auto-Incrément	Non Null
idMaintenance	INTEGER	TRUE	TRUE	FALSE
description	TEXT	FALSE	FALSE	TRUE
freqKm	INTEGER	TRUE	FALSE	FALSE
kmPremiereMaintenance	INTEGER	FALSE	FALSE	FALSE
kmDerniereMaintenance	INTEGER	FALSE	FALSE	FALSE
dateDerniereMaintenance	TEXT	FALSE	FALSE	FALSE
fait	INTEGER	FALSE	FALSE	FALSE
idVehicule	INTEGER	FALSE	FALSE	FALSE

Pour le reste je n'ai rien changé.

Les photos sont stocké en type BLOB, il s'agit de stocker des données binaires. Ca a l'avantage de pouvoir stocker n'importe quoi comme donnée mais ça prend du temps à les récupérer.

vehicule				
Nom	Type	Clef primaire	Auto-Incrément	Non Null

idVehicule	INTEGER	TRUE	TRUE	FALSE
nom	TEXT	FALSE	FALSE	TRUE
description	TEXT	FALSE	FALSE	FALSE
kmInitial	INTEGER	FALSE	FALSE	FALSE
kmReel	INTEGER	FALSE	FALSE	FALSE
photo	BLOB	FALSE	FALSE	FALSE

trajet				
Nom	Type	Clef primaire	Auto-Incrément	Non Null
idTrajet	INTEGER	TRUE	TRUE	FALSE
depart	TEXT	FALSE	FALSE	FALSE
arrivee	TEXT	FALSE	FALSE	FALSE
distance	INTEGER	FALSE	FALSE	TRUE
date	TEXT	FALSE	FALSE	FALSE
idVehicule	INTEGER	FALSE	FALSE	FALSE

8.1.2 Ajout au projet

Dans mon projet, j'ai dû faire une base de données SQLite.

Pour lier la base de données au projet j'ai commencé par importer la librairie « System.Data.SQLite.dll ».

Ensuite, j'ai ajouté la référence « System.Data.SQLite » au projet.

Puis, j'ai ajouté le using « System.Data.SQLite » dans la classe BD pour pouvoir utiliser la base de données dans le code du projet.

Puis, j'ai placé la base de données motoCare.sqlite dans le dossier « MotoCare/bin/Debug » du projet pour qu'elle soit accessible rapidement.

Ensuite, j'ai créé une nouvelle connexion à la base de données en indiquant où se trouvait la base de données et quel était la version. :

```
const string CHAINE_CONNEXION = "Data Source=motoCare.sqlite;Version=3;";
public SQLiteConnection maConnexion = new SQLiteConnection(CHAINE_CONNEXION);
```

Figure 25 - Connexion à la base de données

8.2 Classes

Pour pouvoir réaliser le projet correctement j'ai divisé le projet en plusieurs classes.

8.2.1 Diagramme de classe

Le projet étant en modèle-vue, voici les classes concernant le modèle :

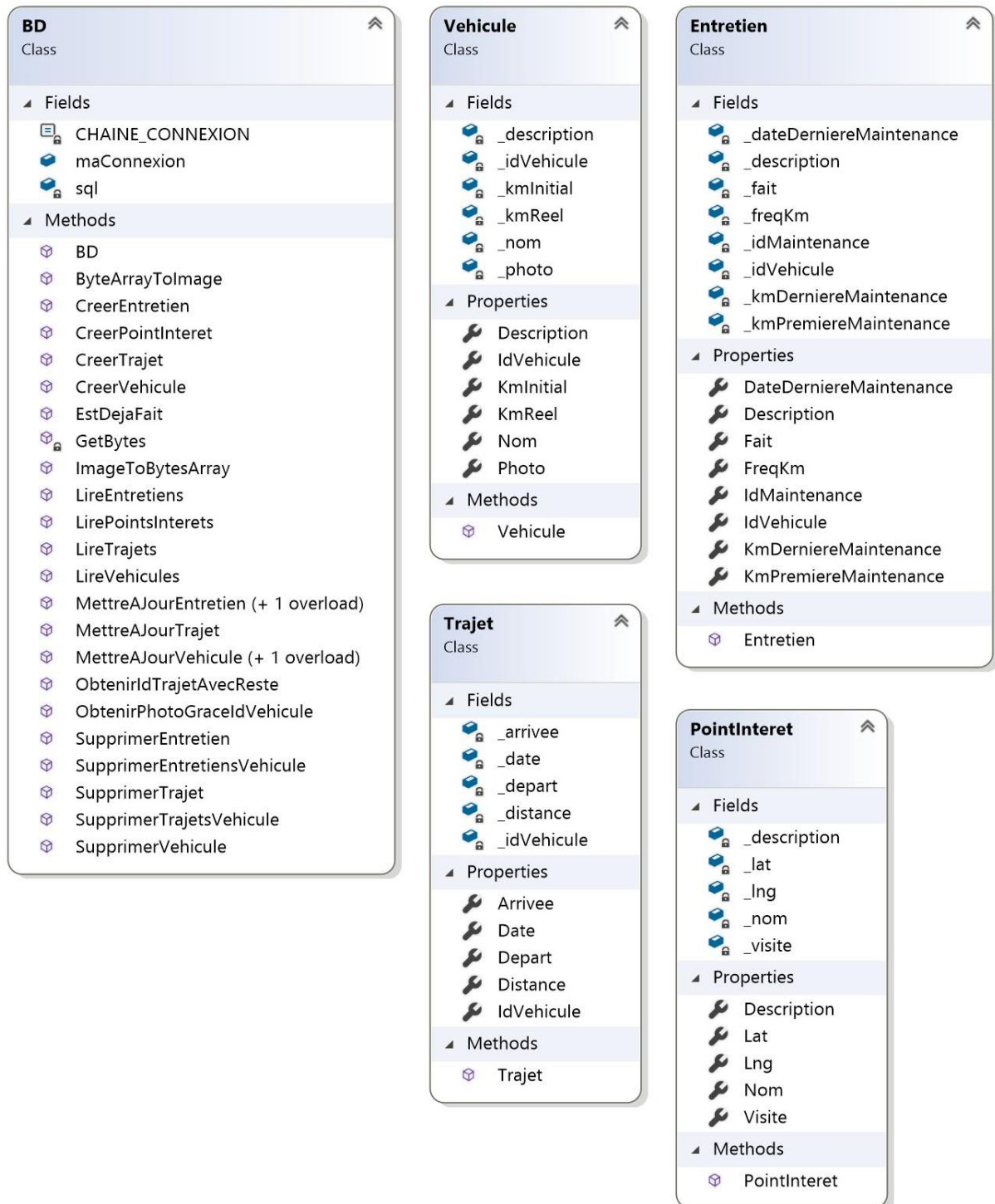


Figure 26 - Diagramme de classe du modèle

Puis voici les classes concernant la vue :

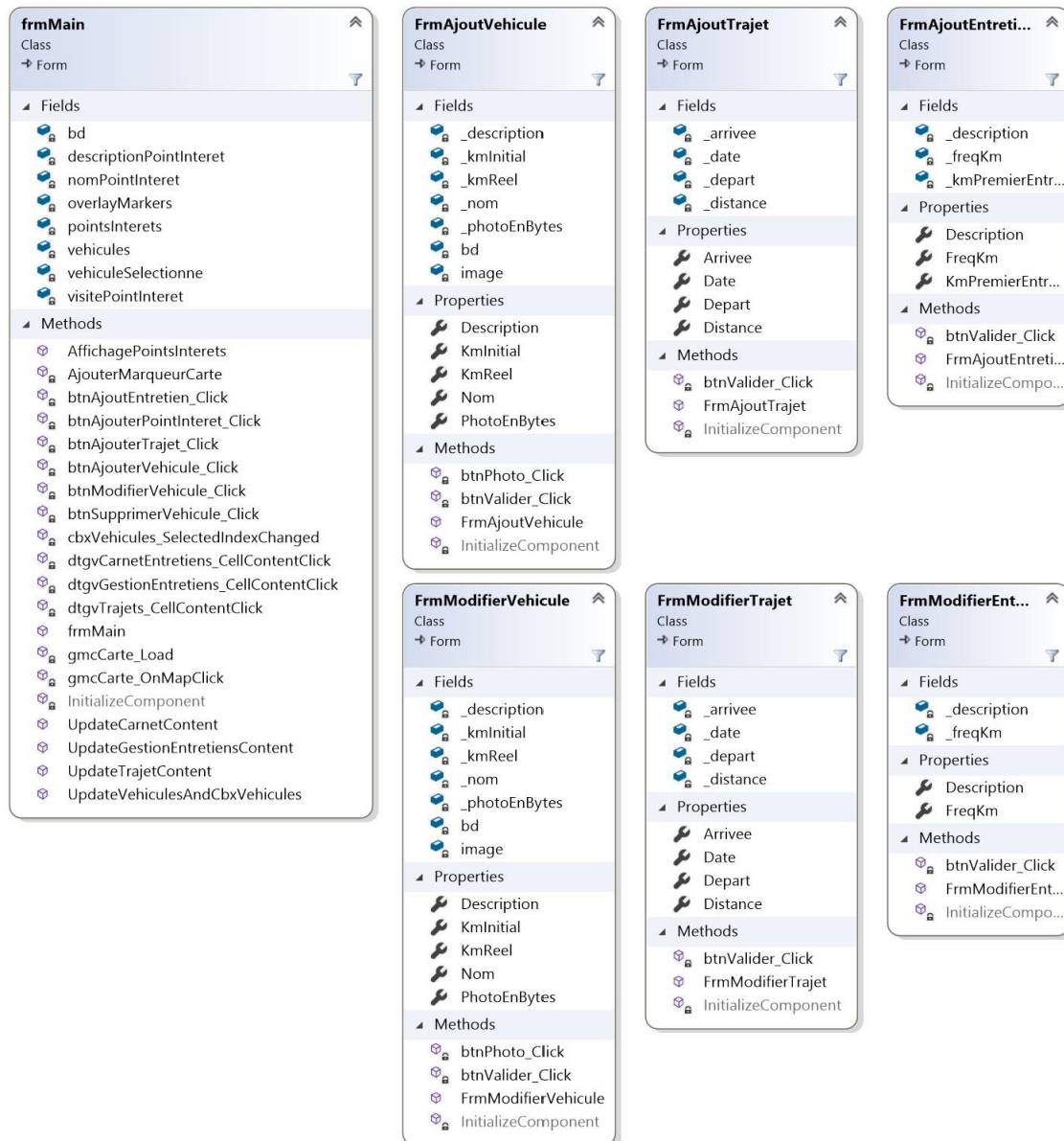


Figure 27 - Diagramme de classe de la vue

8.2.2 BD.cs

Cette classe contient toutes les méthodes qui effectuent une action sur la base de données et quelques méthodes pour la gestion des images.

- Méthode **ByteArrayToImage**
 - Cette méthode a pour but de convertir un tableau d'octet en une Image.
- Méthode **GetBytes**
 - Cette méthode récupère un tableau d'octet à partir de n'importe quelle donnée.
- Méthode **ImageToByteArray**

- Cette méthode convertit une image en un tableau d'octet.
- Méthode ObtenirPhotoGraceldVehicule
 - Cette méthode effectue une requête sur la base de données qui récupère les données binaires du champ photo puis les convertit en tableau d'octet et enfin en une image.
- Méthode LireVehicules
 - Cette méthode effectue une requête sur la base de données qui récupère toutes les informations de tous les véhicules et crée une liste d'objet Vehicule. Elle retourne cette liste.
- Méthode CreerVehicule
 - Cette méthode ajoute un véhicule à la base de données.
- Méthode SupprimerVehicule
 - Cette méthode supprime le véhicule donné en paramètre de la base de données.
- Méthode MettreAJourVehicule
 - Cette première surcharge met à jour dans la base de données les informations d'un véhicule fournit en paramètre.
 - Cette deuxième surcharge met à jour dans la base de données le kilométrage réel d'un véhicule fourni en paramètre.
- Méthode CreerTrajet
 - Cette méthode crée un nouveau trajet dans la base de données.
- Méthode LireTrajets
 - Cette méthode lit tous les trajets d'un véhicule se trouvant dans la base de données. Elle retourne le résultat sous forme de liste d'objet Trajet.
- Méthode SupprimerTrajet
 - Cette méthode supprime un trajet de la base de données simplement avec le numéro d'identification du trajet.
- Méthode SupprimerTrajetsVehicule
 - Cette méthode supprime tous les trajets de la base de données correspondant au numéro d'identification du véhicule fournit en paramètre.
- Méthode MettreAJourTrajet
 - Cette méthode met à jour dans la base de données les nouvelles informations d'un trajet.
- Méthode ObtenirIdTrajetAvecReste
 - Cette méthode récupère le numéro d'identification du trajet correspondant aux paramètres fournis.
- Méthode LireEntretiens
 - Cette méthode lit tous les entretiens d'un véhicule se trouvant dans la base de données. Elle retourne le résultat sous forme de liste d'objet Entretiens.
- Méthode MettreAJourEntretien

- Cette méthode met à jour le champ « fait » d'un entretien se trouvant dans la base de données. Elle met à jour le bon entretien grâce à son numéro d'identification fourni.
- Cette deuxième surcharge met à jour dans la base de données plusieurs informations de l'entretien fourni en paramètre.
- Méthode CreerEntretien
 - Cette méthode crée un nouvel entretien dans la base de données.
- Méthode SupprimerEntretien
 - Cette méthode supprime de la base de données l'entretien correspondant au numéro d'identification de l'entretien fourni en paramètre.
- Méthode SupprimerEntretiensVehicule
 - Cette méthode supprime tous les entretiens correspondant au numéro d'identification du véhicule fourni en paramètre.
- Méthode EstDejaFait
 - Cette méthode indique si l'entretien correspondant au numéro d'identification fourni en paramètre a déjà été effectué ou non. Elle retourne True si oui et False si ce n'est pas le cas.
- Méthode LirePointsInterets
 - Cette méthode récupère absolument tous les points d'intérêts de la base de données et en retourne une liste d'objet PointInteret.
- Méthode CreerPointInteret
 - Cette méthode ajoute dans la base de données un nouveau point d'intérêt.

8.2.3 Vehicule.cs

Cette classe permet de simplifier la manipulation des véhicules. Elle contient des champs, leurs propriétés et un constructeur. Elle permet aussi d'enregistrer durant le programme le véhicule étant sélectionné, cela évite de toujours effectuer une requête sur la base de données ce qui fait perdre beaucoup de temps. Le véhicule sélectionné contient toutes les mêmes informations que celui dans la base de données.

J'appelle le constructeur de cette classe seulement lors de la lecture des véhicules à partir de la classe BD.cs. Cela sert à retourner facilement toutes les informations des véhicules.

8.2.4 Trajet.cs

Cette classe permet de simplifier la manipulation des Trajets. Elle contient des champs, leurs propriétés et un constructeur.

Le constructeur de cette classe est appelé une seule fois lors de la lecture des trajets dans BD.cs. Cela sert à retourner facilement toutes les informations des trajets.

8.2.5 Entretien.cs

Cette classe permet de simplifier la manipulation des Entretiens. Elle contient des champs, leurs propriétés et un constructeur.

Dans le constructeur, il adapte les paramètres fournis avec les valeurs correctes. Par exemple pour la propriété Fait qui est un booléen, on fournit au constructeur un string « 0 » ou « 1 » et en fonction de ce que l'on fournit il va définir à true ou false la propriété.

Le constructeur de cette classe est appelé une seule fois lors de la lecture des entretiens dans BD.cs. Cela sert à nouveau à faciliter la manipulation des informations des entretiens pour en retourner une simple liste d'objet Entretien.

8.2.6 PointInteret.cs

Cette classe permet de simplifier la manipulation des Points d'intérêts. Elle contient des champs, leurs propriétés et un constructeur.

Ce constructeur est appelé deux fois dans le programme. Une fois lors de la lecture des points d'intérêts dans BD.cs. Puis une deuxième fois lors de l'ajout d'un point d'intérêt, à ce moment-là on va ajouter un PointInteret à la liste des points d'intérêts tout en l'ajoutant dans la base de données. Mais on ne va pas effectuer une relecture des points d'intérêts de la base de données. On effectue cela juste une seule fois au lancement de l'application. Ensuite on les ajoute juste dans la liste d'objet et sur la carte, cela rend l'application légèrement plus rapide.

8.3 Carte du monde

Pour ajouter une carte du monde dans le projet j'ai procédé avec la librairie GMap.

J'ai commencé par ajouter un paquet NuGet « GMap.Net.Windows » qui installe ensuite plusieurs paquets nécessaires, j'ai obtenu après l'installation les paquets suivants :

- GMap.Net.Core
- GMap.Net.Windows
- GMap.Net.WinForms
- GMap.Net.WinPresentation

Puis pour pouvoir utiliser GMap et les composants j'ai eu besoins d'ajouter dans la fiche où se trouve la carte, les using suivants :

- Gmap.Net.MapProviders
- Gmap.Net
- Gmap.Net.WindowsForms
- Gmap.Net.WindowsForms.Markers

Il est maintenant possible de travailler avec les composants de GMap. Pour savoir comment utiliser la carte, voir Exemple de code de ce document.

8.4 Exemple de code

8.4.1 Base de données

L'exemple de code suivant sélectionne tous les véhicules de la base de données et les retourne sous forme de liste de véhicules.

Mais avant de pouvoir effectuer une action sur la base de données il faut s'assurer avoir bien effectué la connexion, en dehors de la méthode dans mon cas.

```
const string CHAINE_CONNEXION = "Data Source=motoCare.sqlite;Version=3;";  
public SQLiteConnection maConnexion = new SQLiteConnection(CHAINE_CONNEXION);
```

Figure 28 - Connexion à la base de données

Une fois cela fait, il faut ouvrir la connexion à la base de données puis la refermer une fois le résultat de la requête obtenu :

```
bd.maConnexion.Open();  
vehicules = bd.LireVehicules();  
bd.maConnexion.Close();
```

Figure 29 - Ouverture et fermeture de la connexion à la base de données

Voici ce que contient la méthode LireVehicules ci-dessus :

```
public List<Vehicule> LireVehicules()  
{  
    List<Vehicule> vehicules = new List<Vehicule>();  
    sql = "SELECT idVehicule, nom, description, kmInitial, kmReel, photo FROM vehicule";  
    SQLiteCommand command = new SQLiteCommand(sql, maConnexion);  
    //Pour récupérer toutes les lignes et colonnes  
    SQLiteDataReader dtReader = command.ExecuteReader();  
  
    while (dtReader.Read())  
    {  
        vehicules.Add(new Vehicule(  
            dtReader["idVehicule"].ToString(),  
            dtReader["nom"].ToString(),  
            dtReader["description"].ToString(),  
            dtReader["kmInitial"].ToString(),  
            dtReader["kmReel"].ToString(),  
            ObtenirPhotoGraceIdVehicule(dtReader["idVehicule"].ToString())  
        ));  
    }  
    return vehicules;  
}
```

Figure 30 - Méthode LireVehicules

Au début de cette méthode, une nouvelle liste d'objet Vehicule est créé.

Ensuite on prépare la requête SQL qui va récupérer toutes les informations de chaque véhicule. On établit ensuite le lien avec la requête et la base de données grâce à la « maConnexion ». Puis on exécute la requête et enregistre le contenu des lignes et colonnes retournées dans un SqlDataReader.

Ensuite on parcourt chaque ligne de dtReader. On ajoute pour chaque ligne un nouveau véhicule avec les informations correspondantes à la bonne colonne du résultat retourné par la requête. Pour l'image on va faire appel à une autre méthode qui va nous retourner une image de type Image en fonction de l'id du véhicule

Puis finalement on retourne la liste de véhicules.

8.4.2 GMap

Pour créer la carte GMap il faut commencer par ajouter le composant « GMapControl » dans la fiche.

Ensuite, dans les propriétés j'ai apporté quelques modifications. J'ai modifié la propriété MaxZoom à 18 et MinZoom à 2 pour que l'utilisateur puisse zoomer et dézoomer sur la carte. Ensuite, j'ai mis MouseWheelZoomType à MousePositionWithoutCenter pour que l'utilisateur puisse zoomer et dézoomer à l'endroit où la souris se trouve et non avec le centre de la carte.

Ensuite, lors de l'événement Load de la carte il faut lui donner différentes informations.

```
private void gmcCarte_Load(object sender, EventArgs e)
{
    //Indique quelle carte afficher
    gmcCarte.MapProvider = GoogleMapProvider.Instance;
    //Enlever la croix rouge
    gmcCarte.ShowCenter = false;
    //La où débute la map (sur Genève)
    gmcCarte.Position = new PointLatLng(46.2, 6.1667);
}
```

Figure 31 - Evènement Load de la carte

- MapProvider → Ici j'ai choisi de lui fournir une carte de Google
- ShowCenter → C'est purement esthétique, j'ai décidé de ne pas montrer la croix rouge au centre de la carte
- Position → Positionne la carte aux coordonnées latitude et longitude indiquées. Ici il s'agit de Genève.

Puis pour que la carte se positionne bien proche de Genève il faut attribuer un zoom à la carte dès le début. J'ai décidé de faire cela dans l'interface de Visual Studio avec les propriétés de l'objet. J'ai donc défini la propriété Zoom à 10.

8.5 Protocole de Tests

Plan de tests				
ID	TEST	RESULTAT ATTENDU	RESULTAT OBTENU	STATUT (OK/KO)
Tests en lien avec la base de données				
B1	Affichage des véhicules sur la première page.	Etant donné que je suis un simple utilisateur, je suis sur la page d'accueil de l'application et donc mes véhicules enregistrés dans la base de données sont affichés.	Lorsque je lance l'application et que je possède des véhicules dans la base de données, ils sont affichés dans la liste déroulante.	OK
B2	Création de véhicule sur la première page.	Etant donné que je suis un simple utilisateur, je suis sur la page d'accueil de l'application et j'ai la possibilité d'ajouter un nouveau véhicule. Une fois les champs complétés et validés, le véhicule s'ajoute à la base de données et l'utilisateur retourne sur la page d'accueil.	Lorsque je suis sur la page d'accueil, j'ai la possibilité d'ajouter un nouveau véhicule. Une fois que j'ai remplis tous les champs, le véhicule s'ajoute à la base de données et je retourne automatiquement sur la page d'accueil	OK
B3	Modification de véhicule sur la première page.	Etant donné que je suis un simple utilisateur, je suis sur la page d'accueil de l'application et j'ai la possibilité de modifier un véhicule. Une fois les champs modifiés et mis à jour, l'enregistrement est mis à jour dans la base de données, l'utilisateur retourne alors sur la page d'accueil.	Lorsque je suis sur la page d'accueil, j'ai la possibilité de modifier un véhicule. Une fois que j'ai remplis tous les champs le véhicule est modifié dans la base de données et je retourne automatiquement sur la page d'accueil.	OK

B4	Suppression des véhicules sur la première page.	Etant donné que je suis un simple utilisateur, je suis sur la page d'accueil de l'application et j'ai la possibilité de supprimer un véhicule. Une fois cliqué sur supprimer il y a un message de confirmation, si l'utilisateur valide la suppression, l'enregistrement est donc supprimé de la base de données ainsi que toutes les données liées.	Lorsque je suis sur la page d'accueil, j'ai la possibilité de supprimer un véhicule. Une fois que j'ai confirmé grâce au message de confirmation que je souhaitais supprimer le véhicule, le véhicules et toutes les données liées sont supprimées de la base de données.	OK
B5	Ajout d'un trajet.	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule je peux ensuite ajouter un trajet. Une fois tous les champs entrés et validés, le trajet s'ajoute à la base de données et l'utilisateur retourne sur la liste des trajets.	Lorsque j'ai déjà un véhicule qui est sélectionné, j'ai la possibilité d'ajouter un trajet pour le véhicule. Une fois tous les champs entrés et validés je retourne sur la liste des trajets.	OK
B6	Affichage des trajets.	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule je peux donc voir la liste de tous les trajets effectués avec celui-ci.	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux observer tous les trajets effectués avec mon véhicule.	OK
B7	Modification d'un trajet.	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule je peux donc mettre à jour un trajet. Une fois tous les champs mis à jour, l'enregistrement dans la base de données est	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux mettre à jour un trajet. Une fois tous les champs remplis et validé l'enregistrement dans la base de	OK

		modifié et l'utilisateur retourne sur la liste des trajets.	données est modifié et je retourne sur la liste des trajets.	
B8	Suppression d'un trajet.	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule je peux donc supprimer un des trajets souhaités de la liste. Une fois cliqué sur supprimer il y a un message de confirmation, si l'utilisateur valide la suppression, l'enregistrement est donc supprimé de la base de données.	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux supprimer un trajet de la liste. Une fois cliqué sur supprimer il y a un message de confirmation, si je valide la suppression, l'enregistrement est donc supprimé de la base de données.	OK
B9	Modification du kilométrage du véhicule en fonction des trajets.	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, lors de toute modification/suppression/ajout de trajet, le kilométrage de mon véhicule est automatiquement mis à jour.	Lorsque j'ai déjà un véhicule qui est sélectionné, lorsque je modifie/je supprime ou j'ajoute un trajet, le kilométrage réel de mon véhicule est automatiquement mis à jour.	OK
B10	Affichage des entretiens faits et ceux dans un avenir proche.	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, je peux voir tous les entretiens effectués et ceux à venir prochainement.	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux voir dans l'onglet carnet d'entretiens tous les entretiens effectués et les prochains.	OK

B11	Ajout d'un entretien	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, je peux ajouter un entretien. Une fois les champs remplis l'entretien s'ajoute à la base de données.	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux à partir de l'onglet gestion des entretiens je peux ajouter un nouvel entretien. Une fois les champs remplis l'entretien s'ajoute à la base de données.	OK
B12	Modification d'un entretien	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, je peux modifier un entretien parmi la liste de TOUS les entretiens. Une fois les champs remplis et mis à jour, je retourne sur la liste de tous les entretiens.	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux modifier un entretien parmi tous les entretiens. Une fois les champs remplis et mis à jour, je retourne sur la liste de tous les entretiens.	OK
B13	Suppression d'un entretien	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, je peux supprimer un entretien parmi la liste de TOUS les entretiens. Une fois cliqué sur supprimer il y a un message de confirmation, si l'utilisateur valide la suppression, l'enregistrement est donc supprimé de la base de données.	Lorsque j'ai déjà un véhicule qui est sélectionné, je peux supprimer un entretien parmi la liste de tous les entretiens du véhicule. Une fois cliqué sur supprimer il y a un message de confirmation, quand je valide la suppression, l'enregistrement est donc supprimé de la base de données.	OK

B14	Ajout points d'intérêts	Etant donné que je suis un simple utilisateur, je peux ajouter sur la carte un point d'intérêts. Une fois les champs remplis et validé, le point d'intérêt s'ajoute à la base de données et s'affiche sur la carte.	Lorsque j'arrive sur l'interface principale je peux ajouter un point d'intérêt à partir de l'onglet points d'intérêts. Une fois les champs remplis et validé et la carte cliquée, le point d'intérêt s'ajoute à la base de données.	OK
B15	Affichage des points d'intérêts	Etant donné que je suis un simple utilisateur, je peux visualiser tous mes points d'intérêts se trouvant dans la base de données sur la carte.	Lorsque j'arrive sur l'interface principale je peux visualiser tous mes points d'intérêts se trouvant dans la base de données, sur la carte.	OK
Tests de Design				
D1	Maintenances en rouge	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, lorsque je suis dans le carnet d'entretien les maintenances en retard (en fonction du nombre de kilomètres) sont en rouge.	Lorsque j'ai déjà un véhicule qui est sélectionné et que je suis dans le carnet d'entretien, les maintenances en retard ou qui doivent être faites maintenant sont en rouge.	OK
D2	Maintenance en vert	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, lorsque je suis dans le carnet d'entretien les maintenances effectuées sont en vert.	Lorsque j'ai déjà un véhicule qui est sélectionné et que je suis dans le carnet d'entretien, les maintenances déjà effectuées sont en vert.	OK

D3	Maintenance en orange	Etant donné que je suis un simple utilisateur, ayant déjà sélectionné un véhicule, lorsque je suis dans le carnet d'entretien les maintenances à effectuer bientôt sont en orange.	Lorsque j'ai déjà un véhicule qui est sélectionné et que je suis dans le carnet d'entretien, les maintenances à effectuer bientôt sont en rouge.	OK
Tests de Filtrage				
F1	Incorrectes données entrées lors de l'ajout d'un véhicule	Etant donné que je suis un simple utilisateur, lorsque je suis sur la page d'accueil et que je souhaite ajouter un véhicule mais que j'entre de mauvaises données, un message m'avertissant que je me suis trompé apparaît.	Lorsque j'essaie d'ajouter un véhicule mais que j'entre de mauvaises informations, un message rouge apparaît.	OK
F2	Incorrectes données entrées lors de la modification d'un véhicule	Etant donné que je suis un simple utilisateur, lorsque je suis sur la page d'accueil et que je souhaite mettre à jour un véhicule mais que j'entre de mauvaises données, un message m'avertissant que je me suis trompé apparaît.	Lorsque j'essaie de modifier un véhicule mais que j'entre de mauvaises informations, un message rouge apparaît.	OK
F3	Incorrectes données entrées lors de l'ajout d'un trajet	Etant donné que je suis un simple utilisateur, lorsque je suis sur l'onglet des trajets et que je souhaite ajouter un trajet mais que j'entre de mauvaises données, un message m'avertissant que je me suis trompé apparaît.	Lorsque j'essaie d'ajouter un trajet mais que j'entre de mauvaises informations, un message rouge apparaît.	OK

F4	Incorrectes données entrées lors de la modification d'un trajet	Etant donné que je suis un simple utilisateur, lorsque je suis sur l'onglet des trajets et que je souhaite mettre à jour un trajet mais que j'entre de mauvaises données, un message m'avertissant que je me suis trompé apparaît.	Lorsque j'essaie de modifier un trajet mais que j'entre de mauvaises informations, un message rouge apparaît.	OK
F5	Incorrectes données entrées lors de l'ajout d'un entretien	Etant donné que je suis un simple utilisateur, lorsque je suis sur l'onglet de la gestion des entretiens et que je souhaite ajouter un entretien mais que j'entre de mauvaises données, un message m'avertissant que je me suis trompé apparaît.	Lorsque j'essaie d'ajouter un entretien mais que j'entre de mauvaises informations, un message rouge apparaît.	OK
F6	Incorrectes données entrées lors de la modification d'un entretien	Etant donné que je suis un simple utilisateur, lorsque je suis sur l'onglet de la gestion des entretiens et que je souhaite mettre à jour un entretien mais que j'entre de mauvaises données, un message m'avertissant que je me suis trompé apparaît.	Lorsque j'essaie de modifier un entretien mais que j'entre de mauvaises informations, un message rouge apparaît.	OK

8.6 Arborescence du Projet

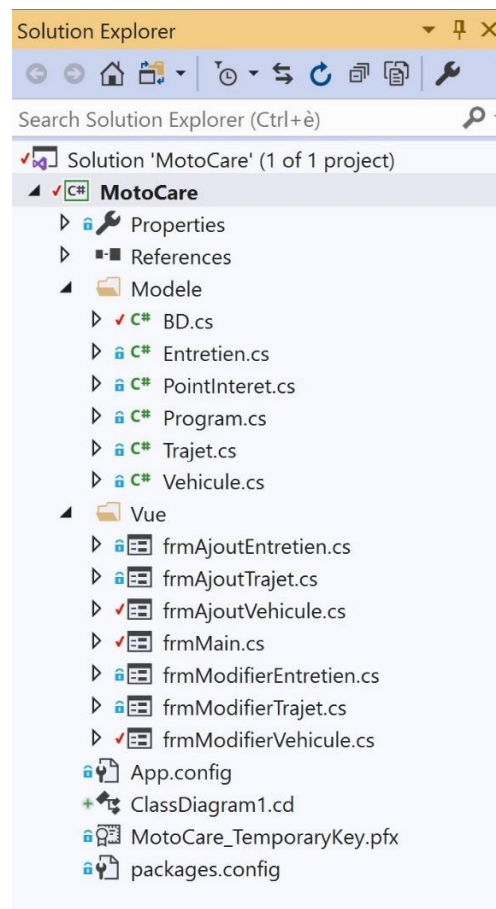


Figure 32 - Arborescence du projet

Le programme étant en modèle-vue j'ai séparé par des dossier l'affichage du modèle.

9 Conclusion

9.1 Difficultés rencontrées

Durant mon projet j'ai rencontré quelques petits problèmes, mais qui ne m'ont pas pour autant trop ralenti sur l'avancement de mon projet. Les soucis rencontrés sont les suivants :

- Un des problèmes rencontrés était de savoir comment faire l'affichage du carnet d'entretien et de la gestion des entretiens. Je ne voulais pas simplement afficher toutes les informations sur les entretiens mais des informations pertinentes et minimiser la quantité d'information sur l'écran ou dans les tableaux.
- Mon problème principal n'était pas majoritairement au niveau du code mais au niveau de la base de données, plus précisément de savoir quel champ je devrais rajouter pour pouvoir différencier ceux qui se répétaient avec la fréquence ou les tout premiers entretiens. Car les premiers entretiens ne possèdent pas de kilomètre de dernier entretien ni de date de dernier entretien.
- J'ai aussi eu quelques soucis en ce qui concerne la lecture et l'ajout d'image dans la base de données. Pour la récupération d'image j'obtenais une erreur provenant de la base de données qui a fini par disparaître simplement. Je me suis dit que c'était peut-être parce que j'avais la base de données ouverte sur le côté, bien que j'aie tout de même réussi à travailler comme ça durant le reste du projet. Il devait y avoir une erreur d'enregistrement avec la base de données.
- J'ai dû aussi découvrir comment gérer les checkbox dans un DataGridView ce qui m'a pris un peu plus de temps que prévu.

9.2 Variantes de solutions et choix

Durant la réalisation de mon projet, j'ai rencontré quelques problématiques et cela m'a amené à faire des choix. Certaines décisions, plus critiques, m'ont demandé plus de réflexions que d'autres. Mais, compte tenu du résultat obtenu, je suis particulièrement satisfait des orientations que j'ai prises. Si vous souhaitez plus de détails sur les problématiques et décisions, je vous invite à consulter les bilans journaliers qui se trouvent dans mon journal de bord.

9.3 Améliorations possibles

Comme amélioration à ce projet il serait intéressant de pouvoir gérer totalement les points d'intérêts. De pouvoir les modifier et les supprimer.

9.4 Bilan personnel

J'ai beaucoup apprécié travailler sur ce projet car c'est un projet qui peut m'être utile personnellement, étant donné que je possède une moto, il serait même possible de gérer d'autres véhicules avec cette application. Je me vois sûrement encore améliorer l'application de mon côté après ce travail. Durant ce travail, j'ai eu beaucoup de plaisir à mettre en

pratique mes connaissances en programmation mais, je dois bien avouer, que la rédaction de la documentation m'a demandé beaucoup d'effort. Cela dit, je suis content de la documentation que j'ai réalisée. Je pense qu'elle est assez complète et permettra à une personne voulant reprendre mon projet d'en comprendre toutes les subtilités.

9.5 Remerciements

Je tiens à remercier

- Mme. Terrier pour son suivi de près de mon TPI
- M. Folomietow et M. Fontanini pour s'assurer que je n'étais pas trop en retard et que j'avais bien compris tout ce qu'il fallait que je fasse dans ce projet.

10Bibliographie

Lors du déroulement de mon projet, j'ai utilisé les ressources suivantes pour obtenir de l'aide technique :

- Pour trier le DataGridView
 - <https://stackoverflow.com/questions/806725/sort-datagridview-columns-in-c-sharp-windows-form>
- Pour les couleurs dans le DataGridView
 - <https://docs.microsoft.com/en-us/dotnet/framework/winforms/controls/how-to-set-font-and-color-styles-in-the-windows-forms-datagridview-control?view=netcore-3.1>
- Pour l'OpenFileDialog
 - <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.openfiledialog?view=netcore-3.1>
- La gestion des images et la bdd
 - <https://www.codeproject.com/Articles/196618/C-SQLite-Storing-Images>
- Découper une chaîne de caractères
 - <https://codes-sources.commentcamarche.net/forum/affich-352126-comment-tronquer-une-chaine-en-c>
- Les checkBox dans le DataGridView
 - <https://stackoverflow.com/questions/13338837/check-uncheck-a-checkbox-on-datagridview>
 - <https://www.c-sharpcorner.com/UploadFile/deveshomar/adding-checkbox-column-in-datagridview-in-C-Sharp-window-forms/>

11 Table des illustrations

Figure 1 - Méthodologie	6
Figure 2 - Planning Prévisionnel.....	9
Figure 3 - Planning Effectif.....	10
Figure 4 - Icône application	12
Figure 5 - Photo moto	13
Figure 6 - La fiche frmMain.....	13
Figure 7 - Onglet Saisie des trajets.....	14
Figure 8 - Onglet Carnet d'Entretiens (Jaune-Vert)	15
Figure 9 - Onglet Carnet d'Entretiens (Rouge-Vert)	16
Figure 10 - Onglet Gestion des Entretiens	17
Figure 11 - Onglet Points d'intérêts	18
Figure 12 - La fiche frmAjoutVehicule.....	19
Figure 13 - La fiche frmModifierVehicule.....	20
Figure 14 - La fiche frmAjoutTrajet	21
Figure 15 - La fiche frmModifierTrajet	22
Figure 16 - La fiche frmAjoutEntretien.....	23
Figure 17 - La fiche frmModifierEntretien.....	24
Figure 18 - Label informatif	25
Figure 19 - Message suppression véhicule.....	25
Figure 20 - Message suppression trajet	25
Figure 21 - Message suppression entretien	25
Figure 22 - Message modification entretien impossible.....	26
Figure 23 - Message mauvais format de fichier	26
Figure 24 - Modèle conceptuel	27
Figure 25 - Connexion à la base de données	29
Figure 26 - Diagramme de classe du modèle	30
Figure 27 - Diagramme de classe de la vue	31
Figure 28 - Connexion à la base de données	35
Figure 29 - Ouverture et fermeture de la connexion à la base de données	35
Figure 30 - Méthode LireVehicules	35
Figure 31 - Evènement Load de la carte.....	36
Figure 32 - Arborescence du projet	44

12 Annexes

1. Enoncé
2. Résumé TPI
3. Manuel Utilisateur
4. Code Source