TPI 2020 - Moto Care

Luca
I.FA-P3A
École d'informatique (CFPT-I)

19 Mai 2020

Table des matières

Chapitre 1

MotoCare

1.1 Modele

1.1.1 Classe

```
1   /*
2  *  Auteur      :   Luca Wohlers
3  *  Professeur  :   Mme. Anne Terrier
4  *  Experts     :   M. Borys Folomietow et M. Alain Fontanini
5  *  Date        :   08 Juin 2020
6  *  Projet      :   Moto Care
7  *  Version     :   1.0
8  *  Description :   Moto Care est une application développé dans le cadre
9  *                 d'un travail pratique individuel (TPI) que les é↩
       lèves d'informatique
10 *                 doivent effectuer à la fin de leur CFC.
11 *                 Cette application consiste à gérer des entretiens ↩
       et trajets en fonction
12 *                 de véhicules. On peut aussi voir et ajouter des ↩
       points d'intérêts sur une carte.
13 *  Fichier     :   BD.cs
14 */
15 using System;
16 using System.Collections.Generic;
17 using System.Data;
18 using System.Data.SQLite;
19 using System.Drawing;
20 using System.IO;
21 using System.Linq;
22 using System.Text;
23 using System.Threading.Tasks;
24
25 namespace MotoCare
26 {
27     /// <summary>
28     /// BD est une classe qui permet d'effectuer beaucoup de requêtes ↩
           envers la base de données motoCare.sqlite
29     /// Comme par exemple:
30     /// -Lire tous les véhicules
31     /// -Créer un véhicule
32     /// -Modifier un véhicule
33     /// -Supprimer un véhicule
34     /// -Créer un trajet
35     /// -etc.
36     /// </summary>
37     public class BD
38     {
39         const string CHAINE_CONNEXION = "Data␣↩
               Source=motoCare.sqlite;Version=3;";
40         public SQLiteConnection maConnexion = new ↩
               SQLiteConnection(CHAINE_CONNEXION);
41
42         string sql = string.Empty;
```

```
43
44          public BD()
45          {
46
47          }
48          /// <summary>
49          /// Converti un tableau d'octet en une Image
50          /// </summary>
51          /// <param name="imageBytes"></param>
52          /// <returns>Une Image</returns>
53          public Image ByteArrayToImage(byte[] imageBytes)
54          {
55              MemoryStream ms = new MemoryStream(imageBytes);
56              //Crée l'image à partir de la donnée fournie
57              Image returnImage = Image.FromStream(ms);
58              return returnImage;
59          }
60          /// <summary>
61          /// Obtient un tableau d'octet à partir de données binaires
62          /// </summary>
63          /// <param name="reader"></param>
64          /// <returns>Le tableau d'octet</returns>
65          static byte[] GetBytes(SQLiteDataReader reader)
66          {
67              const int CHUNK_SIZE = 2 * 1024;
68              byte[] buffer = new byte[CHUNK_SIZE];
69              long bytesRead;
70              long fieldOffset = 0;
71              using (MemoryStream stream = new MemoryStream())
72              {
73                  if (reader.IsDBNull(0))
74                  {
75                      return null;
76                  }
77                  else
78                  {
79                      while ((bytesRead = reader.GetBytes(0, fieldOffset, ↩
                           buffer, 0, buffer.Length)) > 0)
80                      {
81                          stream.Write(buffer, 0, (int)bytesRead);
82                          fieldOffset += bytesRead;
83                      }
84                      return stream.ToArray();
85                  }
86
87              }
88          }
89          /// <summary>
90          /// Convertit une image en tableau d'octet
91          /// </summary>
92          /// <param name="image"></param>
93          /// <returns>Un tableau d'octet</returns>
94          public byte[] ImageToBytesArray(Image image)
95          {
96              MemoryStream ms = new MemoryStream();
97              image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
98              return ms.ToArray();
99          }
100         /// <summary>
101         /// Obtient la photo du véhicule grâce à l'id du véhicule
102         /// </summary>
103         /// <param name="idVehicule"></param>
104         /// <returns>Une Image</returns>
105         public Image ObtenirPhotoGraceIdVehicule(string idVehicule)
106         {
107             Image photo = null;
108
109             string query = string.Format("SELECT␣photo␣FROM␣vehicule␣↩
                    WHERE␣idVehicule␣=␣'{0}';", idVehicule);
110             SQLiteCommand command = new SQLiteCommand(query, maConnexion);
111             SQLiteDataReader dtReader = command.ExecuteReader();
112
```

```
113             while (dtReader.Read())
114             {
115                 //Convertit en tableau d'octet les données binaires ↩
                       obtenus à partir de la requête SQL
116                 byte[] buffer = GetBytes(dtReader);
117                 //Convertit le tableau d'octet temporairement obtenu en ↩
                       Image
118                 photo = ByteArrayToImage(buffer);
119             }
120 
121             return photo;
122         }
123         /// <summary>
124         /// Obtient une liste d'objet Vehicule parmit tous les ↩
               véhicules de la base de données
125         /// </summary>
126         /// <returns>Une liste d'objet Vehicule de tous les ↩
               véhicules</returns>
127         public List<Vehicule> LireVehicules()
128         {
129             List<Vehicule> vehicules = new List<Vehicule>();
130             sql = "SELECT␣idVehicule,␣nom,␣description,␣kmInitial,␣↩
                   kmReel,␣photo␣FROM␣vehicule";
131             SQLiteCommand command = new SQLiteCommand(sql, maConnexion);
132             //Pour récupérer toutes les lignes et colonnes
133             SQLiteDataReader dtReader = command.ExecuteReader();
134 
135             while (dtReader.Read())
136             {
137                 vehicules.Add(new Vehicule(
138                     dtReader["idVehicule"].ToString(),
139                     dtReader["nom"].ToString(),
140                     dtReader["description"].ToString(),
141                     dtReader["kmInitial"].ToString(),
142                     dtReader["kmReel"].ToString(),
143                     ObtenirPhotoGraceIdVehicule(dtReader["idVehicule"].ToString())
144                     ));
145             }
146             return vehicules;
147         }
148         /// <summary>
149         /// Ajoute un véhicule à la base de données
150         /// </summary>
151         /// <param name="nom"></param>
152         /// <param name="description"></param>
153         /// <param name="kmInitial"></param>
154         /// <param name="kmReel"></param>
155         /// <param name="photoEnByte"></param>
156         public void CreerVehicule(string nom, string description, ↩
               string kmInitial, string kmReel, byte[] photoEnByte)
157         {
158             //Remplacer les ' par des double '' pour éviter les erreurs ↩
                   avec le SQL
159             string query = string.Format("INSERT␣INTO␣vehicule␣(nom,␣↩
                   description,␣kmInitial,␣kmReel,␣photo)␣" +
160                                          "VALUES␣('{0}',␣'{1}',␣↩
                                              '{2}',␣'{3}',␣@photo);", ↩
                                              nom.Replace("'", "''"), ↩
                                              description.Replace("'", ↩
                                              "''"), kmInitial, kmReel);
161 
162             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                   maConnexion);
163             //Ajoute à un nouveau paramètre de type binaire la photo en ↩
                   octet
164             sQLiteCommand.Parameters.Add("@photo", DbType.Binary, ↩
                   20).Value = photoEnByte;
165             sQLiteCommand.ExecuteNonQuery();
166         }
167         /// <summary>
168         /// Supprime le véhicule fournit
169         /// </summary>
```

```
170         /// <param name="vehicule">Vehicule que à supprimer</param>
171         public void SupprimerVehicule(Vehicule vehicule)
172         {
173             string query = string.Format("DELETE␣FROM␣vehicule␣WHERE␣↩
                idVehicule␣=␣'{0}';", vehicule.IdVehicule);
174             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                maConnexion);
175             sQLiteCommand.ExecuteNonQuery();
176         }
177         /// <summary>
178         /// Met à jour le véhicule dans la base de données
179         /// </summary>
180         /// <param name="idVehicule">Pour savoir quel véhicule mettre à ↩
            jour</param>
181         /// <param name="nom">Nouveau nom</param>
182         /// <param name="description">Nouvelle description</param>
183         /// <param name="kmInitial">Nouveau kmInitial</param>
184         /// <param name="kmReel">Nouveau kmReel</param>
185         /// <param name="photoEnByte">Nouvelle photo en tableau ↩
            d'octet</param>
186         public void MettreAJourVehicule(string idVehicule, string nom, ↩
            string description, string kmInitial, string kmReel, byte[] ↩
            photoEnByte)
187         {
188             //Remplacer les ' par des double '' pour éviter les erreurs ↩
                avec le SQL
189             string query = string.Format("UPDATE␣vehicule␣SET␣nom␣=␣↩
                '{0}',␣description␣=␣'{1}',␣kmInitial␣=␣'{2}',␣kmReel␣=␣↩
                '{3}',␣photo␣=␣@photo␣" +
190                 "WHERE␣idVehicule␣=␣'{4}'", nom.Replace("'", "''"), ↩
                    description.Replace("'", "''"), kmInitial, kmReel, ↩
                    idVehicule);
191 
192             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                maConnexion);
193             sQLiteCommand.Parameters.Add("@photo", DbType.Binary, ↩
                20).Value = photoEnByte;
194             sQLiteCommand.ExecuteNonQuery();
195         }
196         /// <summary>
197         /// Met à jour le kmReel d'un véhicule
198         /// </summary>
199         /// <param name="kmReel"></param>
200         /// <param name="vehicule"></param>
201         public void MettreAJourVehicule(string kmReel, Vehicule vehicule)
202         {
203             string query = string.Format("UPDATE␣vehicule␣SET␣kmReel␣=␣↩
                '{0}'␣WHERE␣idVehicule␣=␣'{1}';", kmReel, ↩
                vehicule.IdVehicule);
204 
205             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                maConnexion);
206             sQLiteCommand.ExecuteNonQuery();
207         }
208         /// <summary>
209         /// Crée un nouveau trajet
210         /// </summary>
211         /// <param name="idVehicule">l'id du véhicule avec lequel le ↩
            trajet a été effectué</param>
212         /// <param name="depart"></param>
213         /// <param name="arrivee"></param>
214         /// <param name="distance"></param>
215         /// <param name="date"></param>
216         public void CreerTrajet(string idVehicule, string depart, ↩
            string arrivee,string distance, string date)
217         {
218             //Remplacer les ' par des double '' pour éviter les erreurs ↩
                avec le SQL
219             string query = string.Format("INSERT␣INTO␣trajet␣(depart,␣↩
                arrivee,␣distance,␣date,␣idVehicule)" +
220                 "VALUES␣('{0}',␣'{1}',␣'{2}',␣'{3}',␣'{4}')", ↩
                    depart.Replace("'", "''"), arrivee.Replace("'", ↩
```

```
                "''"), distance, date, idVehicule);
221
222             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                    maConnexion);
223             sQLiteCommand.ExecuteNonQuery();
224         }
225         /// <summary>
226         /// Obtient tous les trajets effectué par le véhicule en question
227         /// </summary>
228         /// <param name="idVehicule"></param>
229         /// <returns>Une liste d'objet Trajet</returns>
230         public List<Trajet> LireTrajets(string idVehicule)
231         {
232             List<Trajet> trajets = new List<Trajet>();
233             string select = string.Format("SELECT depart, arrivee, ↩
                    distance, date, idVehicule FROM trajet WHERE idVehicule ↩
                    = '{0}';", idVehicule);
234             SQLiteCommand command = new SQLiteCommand(select, ↩
                    maConnexion);
235             SQLiteDataReader dtReader = command.ExecuteReader();
236
237             while (dtReader.Read())
238             {
239                 trajets.Add(new Trajet(
240                     dtReader["depart"].ToString(),
241                     dtReader["arrivee"].ToString(),
242                     dtReader["distance"].ToString(),
243                     dtReader["date"].ToString(),
244                     dtReader["idVehicule"].ToString()
245                     ));
246             }
247             return trajets;
248         }
249         /// <summary>
250         /// Supprime le trajet avec l'id indiqué en paramètre
251         /// </summary>
252         /// <param name="idTrajet">id du trajet à supprimer</param>
253         public void SupprimerTrajet(string idTrajet)
254         {
255             string query = string.Format("DELETE FROM trajet WHERE ↩
                    idTrajet = '{0}';", idTrajet);
256             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                    maConnexion);
257             sQLiteCommand.ExecuteNonQuery();
258         }
259         /// <summary>
260         /// Supprime tous les trajet du véhicule indiqué en paramètre
261         /// </summary>
262         /// <param name="idVehicule">id du véhicule dont faut supprimer ↩
                tous les trajets</param>
263         public void SupprimerTrajetsVehicule(string idVehicule)
264         {
265             string query = string.Format("DELETE FROM trajet WHERE ↩
                    idVehicule = '{0}';", idVehicule);
266             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                    maConnexion);
267             sQLiteCommand.ExecuteNonQuery();
268         }
269         /// <summary>
270         /// Met à jour le trajet correspondant à l'id indiqué
271         /// </summary>
272         /// <param name="depart"></param>
273         /// <param name="arrivee"></param>
274         /// <param name="distance"></param>
275         /// <param name="date"></param>
276         /// <param name="idVehicule"></param>
277         /// <param name="idTrajet">id du trajet à modifier</param>
278         public void MettreAJourTrajet(string depart, string arrivee, ↩
                string distance, string date, string idVehicule, string ↩
                idTrajet)
279         {
```

```
280             //Remplacer les ' par des double '' pour éviter les erreurs avec le SQL
281             string query = string.Format("UPDATE trajet SET depart = '{0}', arrivee = '{1}', distance = '{2}', date = '{3}', idVehicule = '{4}' " +
282                 "WHERE idTrajet = '{5}'", depart.Replace("'", "''"), arrivee.Replace("'", "''"), distance, date, idVehicule, idTrajet);
283
284             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, maConnexion);
285             sQLiteCommand.ExecuteNonQuery();
286         }
287         /// <summary>
288         /// Obtient l'id du trajet dans la base de données en fonction des paramètres indiqués
289         /// </summary>
290         /// <param name="depart"></param>
291         /// <param name="arrivee"></param>
292         /// <param name="distance"></param>
293         /// <param name="date"></param>
294         /// <param name="idVehicule"></param>
295         /// <returns>l'id du trajet</returns>
296         public string ObtenirIdTrajetAvecReste(string depart, string arrivee, string distance, string date, string idVehicule)
297         {
298             string idTrajet = string.Empty;
299
300             string select = string.Format("SELECT idTrajet FROM trajet WHERE depart = '{0}' AND arrivee = '{1}' AND distance = '{2}' AND date = '{3}' AND idVehicule = '{4}' LIMIT 1;", depart, arrivee, distance, date, idVehicule);
301             SQLiteCommand command = new SQLiteCommand(select, maConnexion);
302             SQLiteDataReader dtReader = command.ExecuteReader();
303             while (dtReader.Read())
304             {
305                 idTrajet = dtReader["idTrajet"].ToString();
306             }
307
308             return idTrajet;
309         }
310         /// <summary>
311         /// Lit tous les entretiens de l'id du véhicule indiqué en paramètre
312         /// </summary>
313         /// <param name="idVehicule"></param>
314         /// <returns>Une liste d'objet Entretien</returns>
315         public List<Entretien> LireEntretiens(string idVehicule)
316         {
317             List<Entretien> entretiens = new List<Entretien>();
318             string select = string.Format("SELECT idMaintenance, description, freqKm, kmPremiereMaintenance, kmDerniereMaintenance, dateDerniereMaintenance, fait, idVehicule " +
319                 "FROM maintenance WHERE idVehicule = '{0}'" +
320                 "ORDER BY kmDerniereMaintenance ASC;", idVehicule);
321             SQLiteCommand command = new SQLiteCommand(select, maConnexion);
322             SQLiteDataReader dtReader = command.ExecuteReader();
323
324             while (dtReader.Read())
325             {
326                 entretiens.Add(new Entretien(
327                     dtReader["idMaintenance"].ToString(),
328                     dtReader["description"].ToString(),
329                     dtReader["freqKm"].ToString(),
330                     dtReader["kmPremiereMaintenance"].ToString(),
331                     dtReader["kmDerniereMaintenance"].ToString(),
332                     dtReader["dateDerniereMaintenance"].ToString(),
333                     dtReader["fait"].ToString(),
334                     dtReader["idVehicule"].ToString()
```

```
335                     ));
336             }
337             return entretiens;
338         }
339         /// <summary>
340         /// Met à jour dans la base de données le champ fait. Pour savoir si l'entretien a été fait ou pas encore.
341         /// </summary>
342         /// <param name="idMaintenance">l'id de l'entretien à modifier</param>
343         /// <param name="fait">"0" s'il n'est pas fait. "1" s'il est fait.</param>
344         public void MettreAJourEntretien(string idMaintenance, string fait)
345         {
346             string query = string.Format("UPDATE maintenance SET fait = '{0}' " +
347                 "WHERE idMaintenance = '{1}'", fait, idMaintenance);
348 
349             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, maConnexion);
350             sQLiteCommand.ExecuteNonQuery();
351         }
352         /// <summary>
353         /// Crée un nouvel entretien en fonction des paramètres fournis
354         /// </summary>
355         /// <param name="description">Description de l'entretien</param>
356         /// <param name="freqKm">Fréquence à laquelle effectuer l'entretien</param>
357         /// <param name="kmPremiereMaintenance">Kilométrage du véhicule lors du premier entretien. "-" s'il se répète, que ce n'est pas le premier.</param>
358         /// <param name="kmDerniereMaintenance">Kilométrage de la dernière maintenance effectué du même type</param>
359         /// <param name="dateDerniereMaintenance">Date lors de la dernière maintenance effectué du même type</param>
360         /// <param name="fait">"0" si elle n'est pas effectué. "1" si elle déjà faite</param>
361         /// <param name="idVehicule">L'id du véhicule sur lequel effectuer la maintenance</param>
362         public void CreerEntretien(string description, string freqKm, string kmPremiereMaintenance, string kmDerniereMaintenance, string dateDerniereMaintenance, string fait, string idVehicule)
363         {
364             //Remplacer les ' par des double '' pour éviter les erreurs avec le SQL
365             string query = string.Format("INSERT INTO maintenance (description, freqKm, kmPremiereMaintenance, kmDerniereMaintenance, dateDerniereMaintenance, fait, idVehicule)" +
366                 "VALUES ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}')", description.Replace("'", "''"), freqKm, kmPremiereMaintenance, kmDerniereMaintenance, dateDerniereMaintenance, fait, idVehicule);
367 
368             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, maConnexion);
369             sQLiteCommand.ExecuteNonQuery();
370         }
371         /// <summary>
372         /// Supprime l'entretien indiqué
373         /// </summary>
374         /// <param name="idEntretien"></param>
375         public void SupprimerEntretien(string idEntretien)
376         {
377             string query = string.Format("DELETE FROM maintenance WHERE idMaintenance = '{0}';", idEntretien);
378             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, maConnexion);
379             sQLiteCommand.ExecuteNonQuery();
380         }
381         /// <summary>
```

```
382         /// Supprime tous les entretiens d'un véhicule
383         /// </summary>
384         /// <param name="idVehicule"></param>
385         public void SupprimerEntretiensVehicule(string idVehicule)
386         {
387             string query = string.Format("DELETE FROM maintenance WHERE idVehicule = '{0}';", idVehicule);
388             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, maConnexion);
389             sQLiteCommand.ExecuteNonQuery();
390         }
391         /// <summary>
392         /// Met à jour un entretien avec de nouvelles valeurs
393         /// </summary>
394         /// <param name="description"></param>
395         /// <param name="freqKm"></param>
396         /// <param name="idMaintenance">l'entretien à mettre à jour</param>
397         public void MettreAJourEntretien(string description, string freqKm, string idMaintenance)
398         {
399             string query = string.Format("UPDATE maintenance SET description = '{0}', freqKm = '{1}' " +
400                 "WHERE idMaintenance = '{2}'", description, freqKm, idMaintenance);
401 
402             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, maConnexion);
403             sQLiteCommand.ExecuteNonQuery();
404         }
405         /// <summary>
406         /// Indique en fonction de l'id de l'entretien s'il a déjà été effectué ou non
407         /// </summary>
408         /// <param name="idEntretien"></param>
409         /// <returns>True s'il à déjà été fait. False si pas encore</returns>
410         public bool EstDejaFait(string idEntretien)
411         {
412             bool estDejaFait = false;
413 
414             string select = string.Format("SELECT fait FROM maintenance WHERE idMaintenance = '{0}';", idEntretien);
415 
416             SQLiteCommand command = new SQLiteCommand(select, maConnexion);
417             SQLiteDataReader dtReader = command.ExecuteReader();
418             while (dtReader.Read())
419             {
420                 estDejaFait = Convert.ToBoolean(dtReader["fait"]);
421             }
422 
423             return estDejaFait;
424         }
425         /// <summary>
426         /// Obtient tous les points d'intérêts de la base de données
427         /// </summary>
428         /// <returns>Une liste d'objet PointInteret</returns>
429         public List<PointInteret> LirePointsInterets()
430         {
431             List<PointInteret> pointsInterets = new List<PointInteret>();
432             sql = "SELECT lat, lng, nom, visite, description FROM pointInteret";
433             SQLiteCommand command = new SQLiteCommand(sql, maConnexion);
434             SQLiteDataReader dtReader = command.ExecuteReader();
435 
436             while (dtReader.Read())
437             {
438                 pointsInterets.Add(new PointInteret(
439                     dtReader["lat"].ToString(),
440                     dtReader["lng"].ToString(),
441                     dtReader["nom"].ToString(),
```

```
442                     dtReader["visite"].ToString(),
443                     dtReader["description"].ToString()
444                     ));
445             }
446             return pointsInterets;
447         }
448         /// <summary>
449         /// Crée un point d'intérêt dans la base de données
450         /// </summary>
451         /// <param name="pointInteret"></param>
452         public void CreerPointInteret(PointInteret pointInteret)
453         {
454             //Remplacer les ' par des double '' pour éviter les erreurs ↩
                avec le SQL
455             string query = string.Format("INSERT␣INTO␣pointInteret␣↩
                (lat,␣lng,␣nom,␣visite,␣description)␣" +
456                                          "VALUES␣('{0}',␣'{1}',␣↩
                                             '{2}',␣'{3}',␣'{4}');", ↩
                                             pointInteret.Lat, ↩
                                             pointInteret.Lng, ↩
                                             pointInteret.Nom.Replace("'", ↩
                                             "''"), ↩
                                             pointInteret.Visite ↩
                                             ,pointInteret.Description.Replace("'
                                             "''"));
457
458             SQLiteCommand sQLiteCommand = new SQLiteCommand(query, ↩
                maConnexion);
459             sQLiteCommand.ExecuteNonQuery();
460         }
461     }
462 }
```

Listing 1.1 – ./MotoCare/MotoCare/Modele/BD.cs

1.1.2 State.cs

```
1    /*
2  *  Auteur      :   Luca Wohlers
3  *  Professeur  :   Mme. Anne Terrier
4  *  Experts     :   M. Borys Folomietow et M. Alain Fontanini
5  *  Date        :   08 Juin 2020
6  *  Projet      :   Moto Care
7  *  Version     :   1.0
8  *  Description :   Moto Care est une application développé dans le cadre
9  *                  d'un travail pratique individuel (TPI) que les é↩
       lèves d'informatique
10 *                  doivent effectuer à la fin de leur CFC.
11 *                  Cette application consiste à gérer des entretiens ↩
       et trajets en fonction
12 *                  de véhicules. On peut aussi voir et ajouter des ↩
       points d'intérêts sur une carte.
13 *  Fichier     :   Entretien.cs
14 */
15 using System;
16 using System.Collections.Generic;
17 using System.Linq;
18 using System.Security.Cryptography;
19 using System.Text;
20 using System.Threading.Tasks;
21
22 namespace MotoCare
23 {
24     /// <summary>
25     /// Entretien est une classe qui mémorise quelques informations sur ↩
          des entretiens/maintenances.
26     /// -L'id de l'entretien
27     /// -Description
28     /// -Frequence en km des prochaines maintenances
```

```
29      /// -Kilométrage lors de la première maintenance
30      /// -Kilométrage de la dernière même maintenance effectuée
31      /// -Fait ou pas encore
32      /// -L'id du véhicule auquel la maintenance doit ou a été faite
33      /// </summary>
34      public class Entretien
35      {
36          //Champs
37          private string _idMaintenance;
38          private string _description;
39          private string _freqKm;
40          private string _kmPremiereMaintenance;
41          private string _kmDerniereMaintenance;
42          private DateTime _dateDerniereMaintenance;
43          private bool _fait;
44          private string _idVehicule;
45
46          //Propriétés
47          public string Description { get => _description; set => ↩
                _description = value; }
48          public string FreqKm { get => _freqKm; set => _freqKm = value; }
49          public string KmPremiereMaintenance { get => ↩
                _kmPremiereMaintenance; set => _kmPremiereMaintenance = ↩
                value; }
50          public string KmDerniereMaintenance { get => ↩
                _kmDerniereMaintenance; set => _kmDerniereMaintenance = ↩
                value; }
51          public DateTime DateDerniereMaintenance { get => ↩
                _dateDerniereMaintenance; set => _dateDerniereMaintenance = ↩
                value; }
52          public bool Fait { get => _fait; set => _fait = value; }
53          public string IdVehicule { get => _idVehicule; set => ↩
                _idVehicule = value; }
54          public string IdMaintenance { get => _idMaintenance; set => ↩
                _idMaintenance = value; }
55
56          //Constructeur
57          /// <summary>
58          /// Crée un nouvel Entretien/Maintenance
59          /// </summary>
60          /// <param name="idMaintenance">l'id de la maintenance</param>
61          /// <param name="description">Une description de la ↩
                maintenance</param>
62          /// <param name="freqKm">A quel fréquence elle devra être ↩
                effectuée à nouveau</param>
63          /// <param name="kmPremiereMaintenance">A quel kilométrage du ↩
                véhicule faudra faire la première maintenance</param>
64          /// <param name="kmDerniereMaintenance">Le kilométrage du ↩
                véhicule lors de la dernière même maintenance</param>
65          /// <param name="dateDerniereMaintenance">La date de la ↩
                dernière maintenance pour celles qui se répètent</param>
66          /// <param name="fait">Indique si la maintenance a été ↩
                effectué. ("0" --> Non, "1" --> Oui)</param>
67          /// <param name="idVehicule">Indique pour quel véhicule ↩
                appartient cette maintenance</param>
68          public Entretien(string idMaintenance, string description, ↩
                string freqKm, string kmPremiereMaintenance, string ↩
                kmDerniereMaintenance, string dateDerniereMaintenance, ↩
                string fait, string idVehicule)
69          {
70              IdMaintenance = idMaintenance;
71              Description = description;
72              FreqKm = freqKm;
73              KmPremiereMaintenance = kmPremiereMaintenance;
74              KmDerniereMaintenance = kmDerniereMaintenance;
75
76              //S'il s'agit d'une maintenance qui se répète elle possède ↩
                    une date de dernier entretien sinon non
77              if (kmPremiereMaintenance == "0")
78                  DateDerniereMaintenance = new ↩
                        DateTime(Convert.ToInt32(dateDerniereMaintenance.Substring(6, ↩
                        4)), ↩
```

```
                    Convert.ToInt32(dateDerniereMaintenance.Substring(3, 2)),
                    Convert.ToInt32(dateDerniereMaintenance.Substring(0, 2)));
            else
                DateDerniereMaintenance = new DateTime();

            //Si la maintenance a déjà été faite on met Fait à true sinon à false
            if (fait == "1")
                Fait = true;
            else
                Fait = false;

            IdVehicule = idVehicule;
        }
    }
}
```

Listing 1.2 – ./MotoCare/MotoCare/Modele/Entretien.cs

1.1.3 HomeMenu.cs

```
  /*
 *  Auteur      :   Luca Wohlers
 *  Professeur  :   Mme. Anne Terrier
 *  Experts     :   M. Borys Folomietow et M. Alain Fontanini
 *  Date        :   08 Juin 2020
 *  Projet      :   Moto Care
 *  Version     :   1.0
 *  Description :   Moto Care est une application développé dans le cadre
 *                  d'un travail pratique individuel (TPI) que les élèves d'informatique
 *                  doivent effectuer à la fin de leur CFC.
 *                  Cette application consiste à gérer des entretiens et trajets en fonction
 *                  de véhicules. On peut aussi voir et ajouter des points d'intérêts sur une carte.
 *  Fichier     :   PointInteret.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MotoCare
{
    /// <summary>
    /// PointInteret est une classe qui mémorise quelques informations sur des marqueur d'une map.
    /// -Latitude
    /// -Longitude
    /// -Nom du marqueur
    /// -Si déjà visité
    /// -Description
    /// </summary>
    public class PointInteret
    {
        //Champs
        private string _lat;
        private string _lng;
        private string _nom;
        private string _visite;
        private string _description;
        //Propriétés
        public string Lat { get => _lat; set => _lat = value; }
        public string Lng { get => _lng; set => _lng = value; }
        public string Nom { get => _nom; set => _nom = value; }
```

```
43          public string Visite { get => _visite; set => _visite = value; }
44          public string Description { get => _description; set => ↩
                _description = value; }
45
46          //Constructeur
47          public PointInteret(string lat, string lng, string nom, string ↩
                visite, string description)
48          {
49              Lat = lat;
50              Lng = lng;
51              Nom = nom;
52              Visite = visite;
53              Description = description;
54          }
55      }
56  }
```

Listing 1.3 – ./MotoCare/MotoCare/Modele/PointInteret.cs

1.1.4 InfoPage.cs

```
1     /*
2   *  Auteur       :    Luca Wohlers
3   *  Professeur   :    Mme. Anne Terrier
4   *  Experts      :    M. Borys Folomietow et M. Alain Fontanini
5   *  Date         :    08 Juin 2020
6   *  Projet       :    Moto Care
7   *  Version      :    1.0
8   *  Description  :    Moto Care est une application développé dans le cadre
9   *                    d'un travail pratique individuel (TPI) que les é↩
        lèves d'informatique
10  *                    doivent effectuer à la fin de leur CFC.
11  *                    Cette application consiste à gérer des entretiens ↩
        et trajets en fonction
12  *                    de véhicules. On peut aussi voir et ajouter des ↩
        points d'intérêts sur une carte.
13  *  Fichier      :    Trajet.cs
14  */
15  using System;
16  using System.Collections.Generic;
17  using System.Linq;
18  using System.Text;
19  using System.Threading.Tasks;
20
21  namespace MotoCare
22  {
23      /// <summary>
24      /// Trajet est une classe qui mémorise certaines informations sur ↩
            des trajets.
25      /// -Départ
26      /// -Arrivée
27      /// -Distance
28      /// -Date
29      /// -L'id du véhicule qui a effectué le trajet
30      /// </summary>
31      public class Trajet
32      {
33          //Champs
34          private string _depart;
35          private string _arrivee;
36          private string _distance;
37          private DateTime _date;
38          private string _idVehicule;
39          //Propriétés
40          public string Depart { get => _depart; set => _depart = value; }
41          public string Arrivee { get => _arrivee; set => _arrivee = ↩
                value; }
42          public string Distance { get => _distance; set => _distance = ↩
                value; }
```

```
43          public DateTime Date { get => _date; set => _date = value; }
44          public string IdVehicule { get => _idVehicule; set => ↩
                _idVehicule = value; }
45
46          //Constructeur
47          public Trajet(string depart, string arrivee, string distance, ↩
                string date, string idVehicule)
48          {
49              Depart = depart;
50              Arrivee = arrivee;
51              Distance = distance;
52              Date = new DateTime(Convert.ToInt32(date.Substring(6, 4)), ↩
                    Convert.ToInt32(date.Substring(3, 2)), ↩
                    Convert.ToInt32(date.Substring(0, 2)));
53              IdVehicule = idVehicule;
54          }
55      }
56  }
```

Listing 1.4 – ./MotoCare/MotoCare/Modele/Trajet.cs

1.1.5 DonkeyKong.cs

```
1     /*
2   *  Auteur       :    Luca Wohlers
3   *  Professeur   :    Mme. Anne Terrier
4   *  Experts      :    M. Borys Folomietow et M. Alain Fontanini
5   *  Date         :    08 Juin 2020
6   *  Projet       :    Moto Care
7   *  Version      :    1.0
8   *  Description  :    Moto Care est une application développé dans le cadre
9   *                    d'un travail pratique individuel (TPI) que les é↩
        lèves d'informatique
10  *                    doivent effectuer à la fin de leur CFC.
11  *                    Cette application consiste à gérer des entretiens ↩
        et trajets en fonction
12  *                    de véhicules. On peut aussi voir et ajouter des ↩
        points d'intérêts sur une carte.
13  *  Fichier      :    Vehicule.cs
14  */
15  using System;
16  using System.Collections.Generic;
17  using System.Drawing;
18  using System.Linq;
19  using System.Security.Cryptography;
20  using System.Text;
21  using System.Threading.Tasks;
22
23  namespace MotoCare
24  {
25      /// <summary>
26      /// Vehicule est une classe qui mémorise certaines information sur ↩
            le véhicule.
27      /// -L'id du véhicule
28      /// -Nom
29      /// -Description
30      /// -Kilométrage initial lors de l'achat du véhicule
31      /// -Kilométrage réel donc actuel du véhicule
32      /// -Image du véhicule
33      /// </summary>
34      public class Vehicule
35      {
36          //Champs
37          private string _idVehicule;
38          private string _nom;
39          private string _description;
40          private string _kmInitial;
41          private string _kmReel;
42          private Image _photo;
```

```
43
44          //Propriétés
45          public string IdVehicule { get => _idVehicule; set => ↩
                _idVehicule = value; }
46          public string Nom { get => _nom; set => _nom = value; }
47          public string Description { get => _description; set => ↩
                _description = value; }
48          public string KmInitial { get => _kmInitial; set => _kmInitial ↩
                = value; }
49          public string KmReel { get => _kmReel; set => _kmReel = value; }
50          public Image Photo { get => _photo; set => _photo = value; }
51
52          //Constructeur
53          public Vehicule(string idVehicule, string nom, string ↩
                description, string kmInitial, string kmReel, Image photo)
54          {
55              IdVehicule = idVehicule;
56              Nom = nom;
57              Description = description;
58              KmInitial = kmInitial;
59              KmReel = kmReel;
60              Photo = photo;
61          }
62      }
63  }
```

Listing 1.5 – ./MotoCare/MotoCare/Modele/Vehicule.cs

1.2 Vue

1.2.1 frmMain.cs

```
1   /*
2  *  Auteur      :   Luca Wohlers
3  *  Professeur  :   Mme. Anne Terrier
4  *  Experts     :   M. Borys Folomietow et M. Alain Fontanini
5  *  Date        :   08 Juin 2020
6  *  Projet      :   Moto Care
7  *  Version     :   1.0
8  *  Description :   Moto Care est une application développé dans le cadre
9  *                  d'un travail pratique individuel (TPI) que les é↩
       lèves d'informatique
10 *                  doivent effectuer à la fin de leur CFC.
11 *                  Cette application consiste à gérer des entretiens ↩
       et trajets en fonction
12 *                  de véhicules. On peut aussi voir et ajouter des ↩
       points d'intérêts sur une carte.
13 *  Fichier     :   frmMain.cs
14 */
15 using System;
16 using System.Collections.Generic;
17 using System.ComponentModel;
18 using System.Data;
19 using System.Drawing;
20 using System.Linq;
21 using System.Text;
22 using System.Threading.Tasks;
23 using System.Windows.Forms;
24 using GMap.NET.MapProviders;
25 using GMap.NET;
26 using GMap.NET.WindowsForms;
27 using GMap.NET.WindowsForms.Markers;
28 using GMap.NET.Internals;
29
30 namespace MotoCare
31 {
32     public partial class frmMain : Form
33     {
34         //Création de l'objet BD
35         BD bd = new BD();
36
37         List<PointInteret> pointsInterets = new List<PointInteret>();
38         List<Vehicule> vehicules;
39         Vehicule vehiculeSelectionne;
40         //Initialisation des valeur pour les marqueurs
41         string nomPointInteret = string.Empty;
42         string visitePointInteret = string.Empty;
43         string descriptionPointInteret = string.Empty;
44
45         //Initialisation de la couche de la carte
46         GMapOverlay overlayMarkers = new GMapOverlay("OverlayMarkers");
47
48         public frmMain()
49         {
50             InitializeComponent();
51             vehicules = new List<Vehicule>();
52
53             UpdateVehiculesAndCbxVehicules();
54             bd.maConnexion.Open();
55             pointsInterets = bd.LirePointsInterets();
56             bd.maConnexion.Close();
57             //Enregistre les composants des onglets pour que si on les ↩
                   supprime
58             //on puisse les afficher à nouveau
59             tpTrajets.SuspendLayout();
60             tpCarnet.SuspendLayout();
61             tpGestion.SuspendLayout();
62
63             AffichagePointsInterets();
```

```
64          }
65          /// <summary>
66          /// Affiche les points d'intérêts sur la carte
67          /// </summary>
68          public void AffichagePointsInterets()
69          {
70              foreach (PointInteret pointInteret in pointsInterets)
71              {
72                  PointLatLng location = new PointLatLng(Convert.ToDouble(pointInteret.Lat), Convert.ToDouble(pointInteret.Lng));
73                  AjouterMarqueurCarte(location, overlayMarkers, GMarkerGoogleType.green_dot, pointInteret);
74              }
75              //Supprimer et ajouter la couche pour rafraichir l'affichage
76              gmcCarte.Overlays.Clear();
77              gmcCarte.Overlays.Add(overlayMarkers);
78          }
79          /// <summary>
80          /// Ajoute le point d'intérêt fournit en paramètre sur la carte
81          /// </summary>
82          /// <param name="latLng">Position du marqueur</param>
83          /// <param name="overlayMarkers">La couche sur lequel ajouter le marqueur</param>
84          /// <param name="style">Style du marqueur</param>
85          /// <param name="pointInteret">PointInteret</param>
86          private void AjouterMarqueurCarte(PointLatLng latLng, GMapOverlay overlayMarkers, GMarkerGoogleType style, PointInteret pointInteret)
87          {
88              //Crée le marqueur sur une carte avec les bonnes coordonées et un avec un certain style
89              GMapMarker marqueur = new GMarkerGoogle(latLng, style);
90
91              string infos = string.Empty;
92
93              infos = "Nom:␣" + pointInteret.Nom + "\r\n";
94              infos += "Lieu␣visité:␣" + pointInteret.Visite + "\r\n";
95              infos += "Description:␣" + pointInteret.Description + "\r\n";
96
97              //Ajoute du texte à la bulle du marqueur lors du "hover" avec la souris sur le marqueur
98              marqueur.ToolTipText = infos;
99
100             //Ajouter le marker à la couche
101             overlayMarkers.Markers.Add(marqueur);
102         }
103         /// <summary>
104         /// Met à jour la liste des véhicules et tout l'affichage en fonction de s'il y a des véhicules ou non
105         /// </summary>
106         public void UpdateVehiculesAndCbxVehicules()
107         {
108             bd.maConnexion.Open();
109             vehicules = bd.LireVehicules();
110             bd.maConnexion.Close();
111
112             cbxVehicules.Items.Clear();
113
114             //S'il y a des véhicules dans ce que la bdd a retourné on les ajoute au comboBox et on active/ajoute des contrôles
115             if (vehicules.Count >= 1)
116             {
117                 foreach (Vehicule v in vehicules)
118                 {
119                     cbxVehicules.Items.Add(v.Nom);
120                 }
121                 cbxVehicules.Enabled = true;
122                 btnSupprimerVehicule.Enabled = true;
123                 btnModifierVehicule.Enabled = true;
124
125                 if (tcMenu.TabCount == 1)
```

```
126                 {
127                     tcMenu.TabPages.Insert(0, tpTrajets);
128                     tcMenu.TabPages.Insert(1, tpCarnet);
129                     tcMenu.TabPages.Insert(2, tpGestion);
130                 }
131             }
132             else
133             {
134                 //Sinon quand aucun véhicule n'est sélectionné on ↩
                       désactive presque tout les contrôles sauf le bouton ↩
                       ajouter et les points d'intérêts
135                 //On enlève aussi les onglets qui dépendent du véhicule ↩
                       sélectionné vu qu'il y en a pas
136                 cbxVehicules.Items.Add("Aucun véhicule, veuillez en↩
                       créer");
137                 cbxVehicules.Enabled = false;
138                 btnModifierVehicule.Enabled = false;
139                 btnSupprimerVehicule.Enabled = false;
140                 tcMenu.TabPages.Remove(tpTrajets);
141                 tcMenu.TabPages.Remove(tpCarnet);
142                 tcMenu.TabPages.Remove(tpGestion);
143                 tbxDescription.Text = "";
144                 tbxKmInitial.Text = "";
145                 tbxKmReel.Text = "";
146                 pcbPhoto.Image = null;
147             }
148             cbxVehicules.SelectedIndex = 0;
149         }
150         /// <summary>
151         /// Met à jour le DataGridView des trajets avec les valeurs ↩
                   actuelles des trajets et le kmRéel du véhicule sélectionné ↩
                   (Affichage et objet)
152         /// </summary>
153         public void UpdateTrajetContent()
154         {
155             int totalKmTrajets = 0;
156             bd.maConnexion.Open();
157             dtgvTrajets.Rows.Clear();
158             foreach (Trajet trajet in ↩
                       bd.LireTrajets(vehiculeSelectionne.IdVehicule))
159             {
160                 dtgvTrajets.Rows.Add(trajet.Depart, trajet.Arrivee, ↩
                       trajet.Distance, trajet.Date);
161                 totalKmTrajets += Convert.ToInt32(trajet.Distance);
162             }
163             bd.maConnexion.Close();
164 
165             //Tri les données par ordre de date
166             dtgvTrajets.Sort(colDate, ListSortDirection.Ascending);
167             //Calcule les km réel en fonction des trajets et du ↩
                       kilométrage lors de l'achat
168             int kmReel = totalKmTrajets + ↩
                       Convert.ToInt32(vehiculeSelectionne.KmInitial);
169             tbxKmReel.Text = kmReel.ToString();
170             vehiculeSelectionne.KmReel = kmReel.ToString();
171             //Mettre à jour dans la bdd le kmReel
172             bd.maConnexion.Open();
173             bd.MettreAJourVehicule(kmReel.ToString(), ↩
                       vehiculeSelectionne);
174             bd.maConnexion.Close();
175         }
176         /// <summary>
177         /// Met à jour le DataGridView du carnet d'entretien
178         /// </summary>
179         public void UpdateCarnetContent()
180         {
181             bd.maConnexion.Open();
182             dtgvCarnetEntretiens.Rows.Clear();
183 
184             foreach (Entretien entretien in ↩
                       bd.LireEntretiens(vehiculeSelectionne.IdVehicule))
185             {
```

```
186                 //Pour calculer dans combien de km aura lieux la ↩
                        maintenance (soit c'est la première et c'est du coup ↩
                        kmPremiereMaintenance soit il faut la calculer
187                 int prochaineMaitenance = 0;
188                 if (entretien.KmPremiereMaintenance == "0")
189                 {
190                     //C'est une répétition d'un entretien
191                     prochaineMaitenance = ↩
                            Convert.ToInt32(entretien.KmDerniereMaintenance) ↩
                            + Convert.ToInt32(entretien.FreqKm) - ↩
                            Convert.ToInt32(vehiculeSelectionne.KmReel);
192                     dtgvCarnetEntretiens.Rows.Add(entretien.IdMaintenance, ↩
                            entretien.Fait, entretien.Description, ↩
                            entretien.DateDerniereMaintenance, ↩
                            entretien.KmDerniereMaintenance, ↩
                            entretien.FreqKm, prochaineMaitenance);
193                 }
194                 else
195                 {
196                     //C'est le tout premier entretien
197                     prochaineMaitenance = ↩
                            Convert.ToInt32(entretien.KmPremiereMaintenance) ↩
                            - Convert.ToInt32(vehiculeSelectionne.KmReel);
198                     //Vu que c'est le premier entretien il n'a pas de ↩
                            km ni de date de dernier entretien
199                     dtgvCarnetEntretiens.Rows.Add(entretien.IdMaintenance, ↩
                            entretien.Fait, entretien.Description, "", "", ↩
                            entretien.FreqKm, prochaineMaitenance);
200                 }
201             }
202             bd.maConnexion.Close();
203             dtgvCarnetEntretiens.Sort(colProchaineMaintenance, ↩
                    ListSortDirection.Descending);
204             //Détails d'affichage pour les couleurs et les valeurs
205             foreach (DataGridViewRow row in dtgvCarnetEntretiens.Rows)
206             {
207                 //Si l'entretien a été effectué
208                 if (Convert.ToBoolean(row.Cells["colFait"].Value))
209                 {
210                     row.DefaultCellStyle.ForeColor = Color.Green;
211                     row.Cells["colProchaineMaintenance"].Value = "--";
212                 }
213                 else
214                 {
215                     //Si l'entretien doit être effectué maintenant on ↩
                            l'affiche en rouge sinon en orange s'il faut ↩
                            bientôt le faire
216                     if ↩
                            (Convert.ToInt32(row.Cells["colProchaineMaintenance"].Value)
                            <= 0)
217                         row.DefaultCellStyle.ForeColor = Color.Red;
218                     else if ↩
                            (Convert.ToInt32(row.Cells["colProchaineMaintenance"].Value)
                            <= 100)
219                         row.DefaultCellStyle.ForeColor = Color.Orange;
220                 }
221             }
222         }
223         /// <summary>
224         /// Met à jour le DataGridView de la gestion des entretiens
225         /// </summary>
226         public void UpdateGestionEntretiensContent()
227         {
228             bd.maConnexion.Open();
229             dtgvGestionEntretiens.Rows.Clear();
230 
231             foreach (Entretien entretien in ↩
                    bd.LireEntretiens(vehiculeSelectionne.IdVehicule))
232             {
233                 //S'il s'agit de la répétition d'un entretien on ↩
                        calcule le kilométrage du véhicule lorsque ↩
                        l'entretien a du ou devra être effectué
```

```
234                 if (entretien.KmPremiereMaintenance == "0")
235                     dtgvGestionEntretiens.Rows.Add(entretien.IdMaintenance, ↩
                            entretien.Fait, entretien.Description, ↩
                            Convert.ToInt32(entretien.KmDerniereMaintenance) ↩
                            + Convert.ToInt32(entretien.FreqKm), ↩
                            entretien.FreqKm);
236                 else//Sinon il s'agit juste du kilométrage du premier ↩
                        entretien
237                     dtgvGestionEntretiens.Rows.Add(entretien.IdMaintenance, ↩
                            entretien.Fait, entretien.Description, ↩
                            Convert.ToInt32(entretien.KmPremiereMaintenance), ↩
                            entretien.FreqKm);
238             }
239             bd.maConnexion.Close();
240             dtgvGestionEntretiens.Sort(colKmLorsEntretienGestion, ↩
                    ListSortDirection.Ascending);
241         }
242         private void cbxVehicules_SelectedIndexChanged(object sender, ↩
                EventArgs e)
243         {
244             //Indique quel vehicule est sélectionné et met à jour les ↩
                    champs en fonction du véhicule
245             foreach (Vehicule vehicule in vehicules)
246             {
247                 if (vehicule.Nom == cbxVehicules.SelectedItem.ToString())
248                 {
249                     vehiculeSelectionne = vehicule;
250                     tbxDescription.Text = vehicule.Description;
251                     pcbPhoto.Image = vehicule.Photo;
252                     tbxKmInitial.Text = vehicule.KmInitial;
253                     tbxKmReel.Text = vehicule.KmReel;
254 
255                     UpdateTrajetContent();
256                     UpdateCarnetContent();
257                     UpdateGestionEntretiensContent();
258                 }
259             }
260         }
261         private void gmcCarte_Load(object sender, EventArgs e)
262         {
263             //Indique quelle carte afficher
264             gmcCarte.MapProvider = GoogleMapProvider.Instance;
265             //Enlever la croix rouge
266             gmcCarte.ShowCenter = false;
267             //La ou débute la map (sur Genève)
268             gmcCarte.Position = new PointLatLng(46.2, 6.1667);
269         }
270         private void btnAjouterVehicule_Click(object sender, EventArgs e)
271         {
272             FrmAjoutVehicule frmAjoutVehicule = new FrmAjoutVehicule();
273             if (frmAjoutVehicule.ShowDialog() == DialogResult.OK)
274             {
275                 bd.maConnexion.Open();
276                 bd.CreerVehicule(frmAjoutVehicule.Nom, ↩
                        frmAjoutVehicule.Description, ↩
                        frmAjoutVehicule.KmInitial, frmAjoutVehicule.KmReel, ↩
                        frmAjoutVehicule.PhotoEnBytes);
277                 bd.maConnexion.Close();
278                 UpdateVehiculesAndCbxVehicules();
279             }
280         }
281         private void btnSupprimerVehicule_Click(object sender, ↩
                EventArgs e)
282         {
283             if (MessageBox.Show("Voulez-vous␣vraiment␣supprimer␣↩
                    définitivement␣ce␣véhicule␣?", "Supprimer␣véhicule", ↩
                    MessageBoxButtons.YesNo, MessageBoxIcon.Question) == ↩
                    DialogResult.Yes)
284             {
285                 bd.maConnexion.Open();
286 
```

```
287                 //Lors de la suppression d'un véhicule on supprime é↩
                        galement tous
288                 //les trajets et entretiens correspondants à ce véhicule
289                 bd.SupprimerTrajetsVehicule(vehiculeSelectionne.IdVehicule);
290                 bd.SupprimerEntretiensVehicule(vehiculeSelectionne.IdVehicule);
291 
292                 bd.SupprimerVehicule(vehiculeSelectionne);
293                 bd.maConnexion.Close();
294                 UpdateVehiculesAndCbxVehicules();
295             }
296         }
297         private void btnModifierVehicule_Click(object sender, EventArgs e)
298         {
299             FrmModifierVehicule frmModifierVehicule = new ↩
                    FrmModifierVehicule(vehiculeSelectionne);
300             if (frmModifierVehicule.ShowDialog() == DialogResult.OK)
301             {
302                 bd.maConnexion.Open();
303                 bd.MettreAJourVehicule(vehiculeSelectionne.IdVehicule, ↩
                        frmModifierVehicule.Nom, ↩
                        frmModifierVehicule.Description, ↩
                        frmModifierVehicule.KmInitial, ↩
                        frmModifierVehicule.KmReel, ↩
                        frmModifierVehicule.PhotoEnBytes);
304                 bd.maConnexion.Close();
305                 UpdateVehiculesAndCbxVehicules();
306             }
307         }
308         private void btnAjouterTrajet_Click(object sender, EventArgs e)
309         {
310             FrmAjoutTrajet frmAjoutTrajet = new FrmAjoutTrajet();
311             if (frmAjoutTrajet.ShowDialog() == DialogResult.OK)
312             {
313                 bd.maConnexion.Open();
314                 bd.CreerTrajet(vehiculeSelectionne.IdVehicule, ↩
                        frmAjoutTrajet.Depart, frmAjoutTrajet.Arrivee, ↩
                        frmAjoutTrajet.Distance, frmAjoutTrajet.Date);
315                 bd.maConnexion.Close();
316 
317                 UpdateTrajetContent();
318                 UpdateCarnetContent();
319                 UpdateGestionEntretiensContent();
320             }
321         }
322         private void dtgvTrajets_CellContentClick(object sender, ↩
                DataGridViewCellEventArgs e)
323         {
324             //Pour vérifier que la colonne sélectionnée est une colonne ↩
                    avec des boutons
325             //ET que l'index de la ligne n'est pas -1 (l'entête sinon ç↩
                    a relève une exeption plus tard dans le code)
326             if (dtgvTrajets.Columns[e.ColumnIndex] is ↩
                    DataGridViewButtonColumn && e.RowIndex != -1)
327             {
328                 int ligne = dtgvTrajets.Rows[e.RowIndex].Index;
329                 //on récupère les information sur le trajet pour savoir ↩
                        lequel modifier ou supprimer ensuite
330                 string depart = ↩
                        dtgvTrajets.Rows[ligne].Cells["colDepart"].Value.ToString();
331                 string arrivee = ↩
                        dtgvTrajets.Rows[ligne].Cells["colArrivee"].Value.ToString();
332                 string distance = ↩
                        dtgvTrajets.Rows[ligne].Cells["colDistance"].Value.ToString();
333                 string date = ↩
                        dtgvTrajets.Rows[ligne].Cells["colDate"].Value.ToString();
334 
335                 bd.maConnexion.Open();
336                 string idTrajet = ↩
                        bd.ObtenirIdTrajetAvecReste(depart.Replace("'", ↩
                        "''"), arrivee.Replace("'", "''"), distance, date, ↩
                        vehiculeSelectionne.IdVehicule);
337 
```

```
338                 if (dtgvTrajets.Columns[e.ColumnIndex].Name == ↩
                        "colModifier")
339                 {
340                     FrmModifierTrajet frmModifierTrajet = new ↩
                            FrmModifierTrajet(depart, arrivee, distance, date);
341                     if (frmModifierTrajet.ShowDialog() == DialogResult.OK)
342                     {
343                         bd.MettreAJourTrajet(frmModifierTrajet.Depart, ↩
                                frmModifierTrajet.Arrivee, ↩
                                frmModifierTrajet.Distance, ↩
                                frmModifierTrajet.Date, ↩
                                vehiculeSelectionne.IdVehicule, idTrajet);
344                     }
345                 }
346                 else if (dtgvTrajets.Columns[e.ColumnIndex].Name == ↩
                        "colSupprimer")
347                 {
348                     if (MessageBox.Show("Voulez-vous␣vraiment␣supprimer␣↩
                            ce␣trajet␣?", "Supprimer␣trajet", ↩
                            MessageBoxButtons.YesNo, ↩
                            MessageBoxIcon.Question) == DialogResult.Yes)
349                     {
350                         bd.SupprimerTrajet(idTrajet);
351                     }
352                 }
353                 bd.maConnexion.Close();
354                 UpdateTrajetContent();
355                 UpdateCarnetContent();
356                 UpdateGestionEntretiensContent();
357             }
358         }
359         private void dtgvCarnetEntretiens_CellContentClick(object ↩
                sender, DataGridViewCellEventArgs e)
360         {
361             //Vérifie que l'index de la ligne n'est pas -1 (l'entête ↩
                    sinon ça relève une exeption plus tard dans le code)
362             if (e.RowIndex != -1)
363             {
364                 //Vérifie si la cellule cliquée est bien dans une ↩
                        colonne de type DataGridViewCheckBoxColumn
365                 //ET que la checkbox n'est pas cochée
366                 if (dtgvCarnetEntretiens.Columns[e.ColumnIndex] is ↩
                        DataGridViewCheckBoxColumn && ↩
                        !Convert.ToBoolean(dtgvCarnetEntretiens.Rows[dtgvCarnetEntretiens
367                 {
368                     int ligne = ↩
                            dtgvCarnetEntretiens.Rows[e.RowIndex].Index;
369                     //On récupère quelques informations pour ↩
                            l'entretien qu'on va devoir crée automatiquement ↩
                            vu qu'un entretien a été coché
370                     string description = ↩
                            dtgvCarnetEntretiens.Rows[ligne].Cells["colDescription"].Valu
371                     string freqKm = ↩
                            dtgvCarnetEntretiens.Rows[ligne].Cells["colFrequence"].Value
372                     string faitApresClique = string.Empty;
373
374                     if ↩
                            (Convert.ToBoolean(dtgvCarnetEntretiens.Rows[ligne].Cells["co
375                         faitApresClique = "0";
376                     else
377                         faitApresClique = "1";
378
379                     //Date et kmDerniereMaintenance (donc ↩
                            kmNouvelleMaintenance) une fois avoir coché la ↩
                            checkbox (donc après avoir effectué la maintenance)
380                     string dateMtn = DateTime.Now.ToString();
381                     DateTime dateActuelle = new ↩
                            DateTime(Convert.ToInt32(dateMtn.Substring(6, ↩
                            4)), Convert.ToInt32(dateMtn.Substring(3, 2)), ↩
                            Convert.ToInt32(dateMtn.Substring(0, 2)));
382                     string kmNouvelleMaintenance = ↩
                            vehiculeSelectionne.KmReel;
```

```
383
384                     bd.maConnexion.Open();
385                     string idMaintenance = ↩
                            dtgvCarnetEntretiens.Rows[ligne].Cells["colIdEntretien"].Valu
386                     bd.MettreAJourEntretien(idMaintenance, ↩
                            faitApresClique);
387                     //Le nouvel entretien créé automatiquement n'a pas ↩
                            de km de première maintenance et il n'est pas ↩
                            encore effectué donc "-" et "0"
388                     bd.CreerEntretien(description, freqKm, "-", ↩
                            kmNouvelleMaintenance, dateActuelle.ToString(), ↩
                            "0", vehiculeSelectionne.IdVehicule);
389                     bd.maConnexion.Close();
390                     UpdateCarnetContent();
391                     UpdateGestionEntretiensContent();
392                 }
393             }
394
395         }
396         private void btnAjoutEntretien_Click(object sender, EventArgs e)
397         {
398             FrmAjoutEntretien frmAjoutEntretien = new FrmAjoutEntretien();
399             if (frmAjoutEntretien.ShowDialog() == DialogResult.OK)
400             {
401                 bd.maConnexion.Open();
402                 //Le nouvel entretien n'a pas de date ni de km de ↩
                        dernière maintenance et n'est logiquement pas encore ↩
                        effectué donc "" "" et "0"
403                 bd.CreerEntretien(frmAjoutEntretien.Description, ↩
                        frmAjoutEntretien.FreqKm, ↩
                        frmAjoutEntretien.KmPremierEntretien, "", "", "0", ↩
                        vehiculeSelectionne.IdVehicule);
404                 bd.maConnexion.Close();
405
406                 UpdateCarnetContent();
407                 UpdateGestionEntretiensContent();
408             }
409         }
410         private void dtgvGestionEntretiens_CellContentClick(object ↩
                sender, DataGridViewCellEventArgs e)
411         {
412             if (dtgvGestionEntretiens.Columns[e.ColumnIndex] is ↩
                    DataGridViewButtonColumn && e.RowIndex != -1)
413             {
414                 int ligne = dtgvGestionEntretiens.Rows[e.RowIndex].Index;
415                 //On récupère ces deux champs pour qu'on puisse ↩
                        préremplir les champs automatiquement lors de la ↩
                        modification
416                 string description = ↩
                        dtgvGestionEntretiens.Rows[ligne].Cells["colDescriptionGestion"]
417                 string freqKm = ↩
                        dtgvGestionEntretiens.Rows[ligne].Cells["colFreqKmGestion"].Value
418
419                 bd.maConnexion.Open();
420                 //On récupère l'id de l'entretien à partir de la ↩
                        colonne cachée
421                 string idEntretien = ↩
                        dtgvGestionEntretiens.Rows[ligne].Cells["colIdEntretienGestion"]
422                 if (dtgvGestionEntretiens.Columns[e.ColumnIndex].Name ↩
                        == "colModifierGestion")
423                 {
424                     if (!bd.EstDejaFait(idEntretien))
425                     {
426                         //On fournit en paramètre la description et la ↩
                                fréquence pour préremplir automatiquement
427                         //les champs dans le formulaire de modification ↩
                                de l'entretien
428                         FrmModifierEntretien frmModifierEntretien = new ↩
                                FrmModifierEntretien(description, freqKm);
429                         if (frmModifierEntretien.ShowDialog() == ↩
                                DialogResult.OK)
430                         {
```

```
431                             bd.MettreAJourEntretien(frmModifierEntretien.Description
                                    frmModifierEntretien.FreqKm, idEntretien);
432                         }
433                     }
434                     else
435                     {
436                         MessageBox.Show("Vous ne pouvez pas modifier un ↩
                            entretien déjà effectué.", "Impossible", ↩
                            MessageBoxButtons.OK, ↩
                            MessageBoxIcon.Information);
437                     }
438 
439                 }
440                 else if ↩
                    (dtgvGestionEntretiens.Columns[e.ColumnIndex].Name ↩
                    == "colSupprimerGestion")
441                 {
442                     if (MessageBox.Show("Voulez-vous vraiment supprimer ↩
                        cet entretien ?", "Supprimer entretien", ↩
                        MessageBoxButtons.YesNo, ↩
                        MessageBoxIcon.Question) == DialogResult.Yes)
443                     {
444                         bd.SupprimerEntretien(idEntretien);
445                     }
446                 }
447                 bd.maConnexion.Close();
448                 UpdateTrajetContent();
449                 UpdateCarnetContent();
450                 UpdateGestionEntretiensContent();
451             }
452         }
453         private void btnAjouterPointInteret_Click(object sender, ↩
                EventArgs e)
454         {
455             if (tbxNomPointInteret.Text == "" || ↩
                tbxDescriptionPointInteret.Text == "")
456             {
457                 lblErreurAjoutPointInteret.Text = "Veuillez renseigner ↩
                    tous les champs";
458             }
459             else
460             {
461                 //On stock les information du marqueur pour que ↩
                    lorsqu'on clique sur la carte on ajoute
462                 //le marqueur à l'emplacement cliqué avec les bonnes ↩
                    informations
463                 nomPointInteret = tbxNomPointInteret.Text;
464                 visitePointInteret = ↩
                    cbxVisitePointInteret.SelectedItem.ToString();
465                 descriptionPointInteret = tbxDescriptionPointInteret.Text;
466             }
467         }
468         private void gmcCarte_OnMapClick(PointLatLng pointClick, ↩
                MouseEventArgs e)
469         {
470             if (nomPointInteret != string.Empty && visitePointInteret ↩
                != string.Empty && descriptionPointInteret != string.Empty)
471             {
472                 PointInteret nouveauPointInteret = new ↩
                    PointInteret(pointClick.Lat.ToString().Replace(',', ↩
                    '.'), pointClick.Lng.ToString().Replace(',', '.'), ↩
                    nomPointInteret, visitePointInteret, ↩
                    descriptionPointInteret);
473                 pointsInterets.Add(nouveauPointInteret);
474                 AjouterMarqueurCarte(pointClick, overlayMarkers, ↩
                    GMarkerGoogleType.green_dot, nouveauPointInteret);
475 
476                 //Ajout à la BDD
477                 bd.maConnexion.Open();
478                 bd.CreerPointInteret(nouveauPointInteret);
479                 bd.maConnexion.Close();
480 
```

```
481                 //Reset des valeurs pour ne pas pouvoir ajouter ↩
                        intentionnellement plusieurs fois
482                 //de suite un marqueur contenant les mêmes informations ↩
                        mais à des endroits différents
483                 nomPointInteret = string.Empty;
484                 visitePointInteret = string.Empty;
485                 descriptionPointInteret = string.Empty;
486
487                 //Reset des champs de l'ajout du marqueur
488                 tbxNomPointInteret.Text = "";
489                 cbxVisitePointInteret.SelectedIndex = 0;
490                 tbxDescriptionPointInteret.Text = "";
491
492                 //Supprimer et ajouter la couche pour rafraichir ↩
                        l'affichage
493                 gmcCarte.Overlays.Clear();
494                 gmcCarte.Overlays.Add(overlayMarkers);
495             }
496         }
497     }
498 }
```

Listing 1.6 – ./MotoCare/MotoCare/Vue/frmMain.cs

1.2.2 frmAjoutEntretien.cs

```
1    /*
2  *  Auteur       :   Luca Wohlers
3  *  Professeur   :   Mme. Anne Terrier
4  *  Experts      :   M. Borys Folomietow et M. Alain Fontanini
5  *  Date         :   08 Juin 2020
6  *  Projet       :   Moto Care
7  *  Version      :   1.0
8  *  Description  :   Moto Care est une application développé dans le cadre
9  *                   d'un travail pratique individuel (TPI) que les é↩
       lèves d'informatique
10 *                   doivent effectuer à la fin de leur CFC.
11 *                   Cette application consiste à gérer des entretiens ↩
       et trajets en fonction
12 *                   de véhicules. On peut aussi voir et ajouter des ↩
       points d'intérêts sur une carte.
13 *  Fichier      :   frmAjoutEntretien.cs
14 */
15 using System;
16 using System.Collections.Generic;
17 using System.ComponentModel;
18 using System.Data;
19 using System.Drawing;
20 using System.Linq;
21 using System.Text;
22 using System.Threading.Tasks;
23 using System.Windows.Forms;
24
25 namespace MotoCare
26 {
27     public partial class FrmAjoutEntretien : Form
28     {
29         private string _description;
30         private string _kmPremierEntretien;
31         private string _freqKm;
32
33         public string Description { get => tbxDescription.Text; set => ↩
               _description = value; }
34         public string KmPremierEntretien { get => ↩
               nudKmPremierEntretien.Value.ToString(); set => ↩
               _kmPremierEntretien = value; }
35         public string FreqKm { get => nudFreqKm.Value.ToString(); set ↩
               => _freqKm = value; }
36
```

```
37          public FrmAjoutEntretien()
38          {
39              InitializeComponent();
40          }
41
42          private void btnValider_Click(object sender, EventArgs e)
43          {
44              if (Description == "" || KmPremierEntretien == "" || FreqKm ↩
                    == "")
45              {
46                  lblErreurAjoutTrajet.Text = "Veuillez␣renseigner␣tous␣↩
                        les␣champs";
47              }
48              else
49              {
50                  DialogResult = DialogResult.OK;
51              }
52          }
53      }
54  }
```

Listing 1.7 – ./MotoCare/MotoCare/Vue/frmAjoutEntretien.cs

1.2.3 frmModifierEntretien.cs

```
1     /*
2   *  Auteur       :    Luca Wohlers
3   *  Professeur   :    Mme. Anne Terrier
4   *  Experts      :    M. Borys Folomietow et M. Alain Fontanini
5   *  Date         :    08 Juin 2020
6   *  Projet       :    Moto Care
7   *  Version      :    1.0
8   *  Description  :    Moto Care est une application développé dans le cadre
9   *                    d'un travail pratique individuel (TPI) que les é↩
        lèves d'informatique
10  *                    doivent effectuer à la fin de leur CFC.
11  *                    Cette application consiste à gérer des entretiens ↩
        et trajets en fonction
12  *                    de véhicules. On peut aussi voir et ajouter des ↩
        points d'intérêts sur une carte.
13  *  Fichier      :    frmModifierEntretien.cs
14  */
15  using System;
16  using System.Collections.Generic;
17  using System.ComponentModel;
18  using System.Data;
19  using System.Drawing;
20  using System.Linq;
21  using System.Text;
22  using System.Threading.Tasks;
23  using System.Windows.Forms;
24
25  namespace MotoCare
26  {
27      public partial class FrmModifierEntretien : Form
28      {
29          private string _description;
30          private string _freqKm;
31
32          public string Description { get => tbxDescription.Text; set => ↩
                _description = value; }
33          public string FreqKm { get => nudFreqKm.Value.ToString(); set ↩
                => _freqKm = value; }
34
35          public FrmModifierEntretien(string description, string freqKm)
36          {
37              InitializeComponent();
38              tbxDescription.Text = description;
39              nudFreqKm.Value = Convert.ToInt32(freqKm);
```

```
40          }
41
42          private void btnValider_Click(object sender, EventArgs e)
43          {
44              if (Description == "" || FreqKm == "")
45              {
46                  lblErreurAjoutTrajet.Text = "Veuillez␣renseigner␣tous␣les␣champs";
47              }
48              else
49              {
50                  DialogResult = DialogResult.OK;
51              }
52          }
53      }
54  }
```

Listing 1.8 – ./MotoCare/MotoCare/Vue/frmModifierEntretien.cs

1.2.4 frmAjoutTrajet.cs

```
1     /*
2   *  Auteur       :    Luca Wohlers
3   *  Professeur   :    Mme. Anne Terrier
4   *  Experts      :    M. Borys Folomietow et M. Alain Fontanini
5   *  Date         :    08 Juin 2020
6   *  Projet       :    Moto Care
7   *  Version      :    1.0
8   *  Description  :    Moto Care est une application développé dans le cadre
9   *                    d'un travail pratique individuel (TPI) que les élèves d'informatique
10  *                    doivent effectuer à la fin de leur CFC.
11  *                    Cette application consiste à gérer des entretiens et trajets en fonction
12  *                    de véhicules. On peut aussi voir et ajouter des points d'intérêts sur une carte.
13  *  Fichier      :    frmAjoutTrajet.cs
14  */
15  using System;
16  using System.Collections.Generic;
17  using System.ComponentModel;
18  using System.Data;
19  using System.Drawing;
20  using System.Linq;
21  using System.Text;
22  using System.Threading.Tasks;
23  using System.Windows.Forms;
24
25  namespace MotoCare
26  {
27      public partial class FrmAjoutTrajet : Form
28      {
29          private string _depart;
30          private string _arrivee;
31          private string _distance;
32          private string _date;
33          public string Depart { get => tbxDepart.Text; set => _depart = value; }
34          public string Arrivee { get => tbxArrivee.Text; set => _arrivee = value; }
35          public string Distance { get => nudDistance.Value.ToString(); set => _distance = value; }
36          public string Date { get => dtpDate.Value.ToString(); set => _date = value; }
37
38          public FrmAjoutTrajet()
39          {
40              InitializeComponent();
41          }
```

```
42
43          private void btnValider_Click(object sender, EventArgs e)
44          {
45              if (Depart == "" || Arrivee == "" || Date == "")
46              {
47                  lblErreurAjoutTrajet.Text = "Veuillez␣renseigner␣tous␣les␣champs.";
48              }
49              else
50              {
51                  DialogResult = DialogResult.OK;
52              }
53          }
54      }
55  }
```

Listing 1.9 – ./MotoCare/MotoCare/Vue/frmAjoutTrajet.cs

1.2.5 frmModifierTrajet.cs

```
1     /*
2   *  Auteur       :    Luca Wohlers
3   *  Professeur   :    Mme. Anne Terrier
4   *  Experts      :    M. Borys Folomietow et M. Alain Fontanini
5   *  Date         :    08 Juin 2020
6   *  Projet       :    Moto Care
7   *  Version      :    1.0
8   *  Description  :    Moto Care est une application développé dans le cadre
9   *                    d'un travail pratique individuel (TPI) que les élèves d'informatique
10  *                    doivent effectuer à la fin de leur CFC.
11  *                    Cette application consiste à gérer des entretiens et trajets en fonction
12  *                    de véhicules. On peut aussi voir et ajouter des points d'intérêts sur une carte.
13  *  Fichier      :    frmModifierTrajet.cs
14  */
15  using System;
16  using System.Collections.Generic;
17  using System.ComponentModel;
18  using System.Data;
19  using System.Drawing;
20  using System.Linq;
21  using System.Text;
22  using System.Threading.Tasks;
23  using System.Windows.Forms;
24
25  namespace MotoCare
26  {
27      public partial class FrmModifierTrajet : Form
28      {
29          private string _depart;
30          private string _arrivee;
31          private string _distance;
32          private string _date;
33          public string Depart { get => tbxDepart.Text; set => _depart = value; }
34          public string Arrivee { get => tbxArrivee.Text; set => _arrivee = value; }
35          public string Distance { get => nudDistance.Value.ToString(); set => _distance = value; }
36          public string Date { get => dtpDate.Value.ToString(); set => _date = value; }
37
38          public FrmModifierTrajet(string depart, string arrivee, string distance, string date)
39          {
40              InitializeComponent();
41              tbxDepart.Text = depart;
```

```
42              tbxArrivee.Text = arrivee;
43              nudDistance.Value = Convert.ToDecimal(distance);
44              dtpDate.Value = new ↩
                    DateTime(Convert.ToInt32(date.Substring(6, 4)), ↩
                    Convert.ToInt32(date.Substring(3, 2)), ↩
                    Convert.ToInt32(date.Substring(0, 2))); ;
45          }
46
47          private void btnValider_Click(object sender, EventArgs e)
48          {
49              if (Depart == "" || Arrivee == "" || Date == "")
50              {
51                  lblErreurAjoutTrajet.Text = "Veuillez␣renseigner␣tous␣↩
                        les␣champs.";
52              }
53              else
54              {
55                  DialogResult = DialogResult.OK;
56              }
57          }
58      }
59  }
```

Listing 1.10 – ./MotoCare/MotoCare/Vue/frmModifierTrajet.cs

1.2.6 frmAjoutVehicule.cs

```
1     /*
2   *  Auteur       :    Luca Wohlers
3   *  Professeur   :    Mme. Anne Terrier
4   *  Experts      :    M. Borys Folomietow et M. Alain Fontanini
5   *  Date         :    08 Juin 2020
6   *  Projet       :    Moto Care
7   *  Version      :    1.0
8   *  Description  :    Moto Care est une application développé dans le cadre
9   *                    d'un travail pratique individuel (TPI) que les é↩
      lèves d'informatique
10  *                    doivent effectuer à la fin de leur CFC.
11  *                    Cette application consiste à gérer des entretiens ↩
      et trajets en fonction
12  *                    de véhicules. On peut aussi voir et ajouter des ↩
      points d'intérêts sur une carte.
13  *  Fichier      :    frmAjoutVehicule.cs
14  */
15  using System;
16  using System.Collections.Generic;
17  using System.ComponentModel;
18  using System.Data;
19  using System.Drawing;
20  using System.Linq;
21  using System.Text;
22  using System.Threading.Tasks;
23  using System.Windows.Forms;
24
25  namespace MotoCare
26  {
27      public partial class FrmAjoutVehicule : Form
28      {
29          //Déclaration de bd pour pouvoir faire un appel à la classe BD ↩
              et pouvoir convertir l'image en tableau d'octet
30          BD bd = new BD();
31          Image image;
32
33          private string _nom;
34          private string _description;
35          private string _kmInitial;
36          private string _kmReel;
37          private byte[] _photoEnBytes;
38
```

```
39          public string Nom { get => tbxNom.Text; set => _nom = value; }
40          public string Description { get => tbxDescription.Text; set => ↩
                _description = value; }
41          public string KmInitial { get => nudInitial.Value.ToString(); ↩
                set => _kmInitial = value; }
42
43          //Même valeur que KmInitial car KmReel est calculé avec les ↩
                trajets
44          public string KmReel { get => nudInitial.Value.ToString(); set ↩
                => _kmReel = value; }
45          public byte[] PhotoEnBytes { get => _photoEnBytes; set => ↩
                _photoEnBytes = value; }
46
47          public FrmAjoutVehicule()
48          {
49              InitializeComponent();
50          }
51
52          private void btnPhoto_Click(object sender, EventArgs e)
53          {
54              OpenFileDialog ofdImage = new OpenFileDialog();
55
56              //Définit les critères de l'openFileDialog pour éviter de ↩
                    sélectionner autre chose qu'une image
57              ofdImage.InitialDirectory = "C:\\Desktop\"";
58              ofdImage.Filter = "png␣files␣(*.png)|*.png|All␣files␣↩
                    (*.*)|*.*";
59              ofdImage.FilterIndex = 2;
60              ofdImage.RestoreDirectory = true;
61              ofdImage.Multiselect = false;
62
63              if (ofdImage.ShowDialog() == DialogResult.OK)
64              {
65                  //Permet la sélection d'une image png ou jpg seulement
66                  if ↩
                        (ofdImage.SafeFileName.Substring(ofdImage.SafeFileName.Length ↩
                        - 3, 3) == "png" ||
67                       ofdImage.SafeFileName.Substring(ofdImage.SafeFileName.Length ↩
                            - 3, 3) == "jpg")
68                  {
69                      image = Image.FromFile(ofdImage.FileName);
70                      PhotoEnBytes = bd.ImageToBytesArray(image);
71
72                      pcbPhoto.Image = image;
73                  }
74                  else
75                  {
76                      MessageBox.Show("Veuillez␣sélectionner␣une␣image␣↩
                            .png␣ou␣.jpg", "Impossible", ↩
                            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
77                  }
78
79              }
80          }
81
82          private void btnValider_Click(object sender, EventArgs e)
83          {
84              if (Nom == "" || Description == "" || PhotoEnBytes == null)
85              {
86                  lblErreurAjoutVehicule.Text = "Veuillez␣renseigner␣tous␣↩
                        les␣champs";
87              }
88              else
89              {
90                  DialogResult = DialogResult.OK;
91              }
92          }
93      }
94  }
```

Listing 1.11 – ./MotoCare/MotoCare/Vue/frmAjoutVehicule.cs

1.2.7 frmModifierVehicule.cs

```
1   /*
2   *  Auteur      :   Luca Wohlers
3   *  Professeur  :   Mme. Anne Terrier
4   *  Experts     :   M. Borys Folomietow et M. Alain Fontanini
5   *  Date        :   08 Juin 2020
6   *  Projet      :   Moto Care
7   *  Version     :   1.0
8   *  Description :   Moto Care est une application développé dans le cadre
9   *                  d'un travail pratique individuel (TPI) que les élèves d'informatique
10  *                  doivent effectuer à la fin de leur CFC.
11  *                  Cette application consiste à gérer des entretiens et trajets en fonction
12  *                  de véhicules. On peut aussi voir et ajouter des points d'intérêts sur une carte.
13  *  Fichier     :   frmModifierVehicule.cs
14  */
15 using System;
16 using System.Collections.Generic;
17 using System.ComponentModel;
18 using System.Data;
19 using System.Drawing;
20 using System.Linq;
21 using System.Text;
22 using System.Threading.Tasks;
23 using System.Windows.Forms;
24
25 namespace MotoCare
26 {
27     public partial class FrmModifierVehicule : Form
28     {
29         //Déclaration de bd pour pouvoir faire un appel à la classe BD et pouvoir convertir l'image en tableau d'octet
30         BD bd = new BD();
31         Image image;
32
33         private string _nom;
34         private string _description;
35         private string _kmInitial;
36         private string _kmReel;
37         private byte[] _photoEnBytes;
38
39         public string Nom { get => tbxNom.Text; set => _nom = value; }
40         public string Description { get => tbxDescription.Text; set => _description = value; }
41         public string KmInitial { get => nudInitial.Value.ToString(); set => _kmInitial = value; }
42         public string KmReel { get => _kmReel; set => _kmReel = value; }//Ne change pas
43         public byte[] PhotoEnBytes { get => _photoEnBytes; set => _photoEnBytes = value; }
44
45         public FrmModifierVehicule(Vehicule vehicule)
46         {
47             InitializeComponent();
48             tbxNom.Text = vehicule.Nom;
49             tbxDescription.Text = vehicule.Description;
50             nudInitial.Value = Convert.ToDecimal(vehicule.KmInitial);
51             KmReel = vehicule.KmReel;
52             pcbPhoto.Image = vehicule.Photo;
53
54             image = vehicule.Photo;
55             PhotoEnBytes = bd.ImageToBytesArray(image);
56         }
57
58         private void btnPhoto_Click(object sender, EventArgs e)
59         {
60             OpenFileDialog ofdImage = new OpenFileDialog();
61
```

```
62              ofdImage.InitialDirectory = "C:\\Desktop\"";
63              ofdImage.Filter = "png files (*.png)|*.png|All files (*.*)|*.*";
64              ofdImage.FilterIndex = 2;
65              ofdImage.RestoreDirectory = true;
66              ofdImage.Multiselect = true;
67
68              if (ofdImage.ShowDialog() == DialogResult.OK)
69              {
70                  //Permet la sélection d'une image png ou jpg seulement
71                  if (ofdImage.SafeFileName.Substring(ofdImage.SafeFileName.Length - 3, 3) == "png" ||
72                      ofdImage.SafeFileName.Substring(ofdImage.SafeFileName.Length - 3, 3) == "jpg")
73                  {
74                      image = Image.FromFile(ofdImage.FileName);
75                      PhotoEnBytes = bd.ImageToBytesArray(image);
76
77                      pcbPhoto.Image = image;
78                  }
79                  else
80                  {
81                      MessageBox.Show("Veuillez sélectionner une image .png ou .jpg", "Impossible", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
82                  }
83              }
84          }
85
86          private void btnValider_Click(object sender, EventArgs e)
87          {
88              if (Nom == "" || Description == "" || PhotoEnBytes == null)
89              {
90                  lblErreurAjoutVehicule.Text = "Veuillez renseigner tous les champs";
91              }
92              else
93              {
94                  DialogResult = DialogResult.OK;
95              }
96          }
97      }
98  }
```

Listing 1.12 – ./MotoCare/MotoCare/Vue/frmModifierVehicule.cs