Manuel Technique

Moto Care – Wohlers Luca

# 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.

# Rappel de l’énoncé

## 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> |

## Livrables

* Planning prévisionnel
* Rapport du projet
* Manuel utilisateur
* Journal de bord

## 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 utilisé à l’école.

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

## 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 fait.

* **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.

# Méthodologie

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

Une image contenant texte, carte, signe

Description générée automatiquement

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.

1. **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.

1. **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.

1. **Réaliser**

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

1. **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.

1. **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.