ES

[](https://www.google.ch/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=2ahUKEwiClujH1ITjAhXHIlAKHS9eC-IQjRx6BAgBEAU&url=https%3A%2F%2Ffr.wikipedia.org%2Fwiki%2FFichier%3AMySQL.svg&psig=AOvVaw0KJKunrLYmjNtjCmpCxy33&ust=1561553046142880)

Etudiants : Jonny HOFMANN

& Bilal JOHRY

Filières : informatique & systèmes ind.

Année : 1ère

Année du diplôme : 2021

Responsable : Dominique MONTAVON

Suivi technique : Arnaud MONTFERME

Gestion de projet : Willy GRÜTER

Date : mardi 12.11.2019

**Mandant**

Division Technique

Cités des Microtechniques 3

2900 Porrentruy

**Module 2 – Gestion flexible du chariot**

*Rapport technique*

Table des matières

[1. Introduction 3](#_Toc24448553)

[1.1. But du document 3](#_Toc24448554)

[1.2. But de l’installation 3](#_Toc24448555)

[1.3. Description du travail à réaliser 3](#_Toc24448556)

[1.4. Organisation du projet 3](#_Toc24448557)

[2. Déroulement du projet 4](#_Toc24448558)

[2.1. Cahier des charges 4](#_Toc24448559)

[2.2. Organigramme 4](#_Toc24448560)

[2.3. Planification 4](#_Toc24448561)

[2.4. Base de données 4](#_Toc24448562)

[2.5. Partie C# 5](#_Toc24448563)

[2.6. Partie automate 5](#_Toc24448564)

[2.7. Documentations 5](#_Toc24448565)

[2.8. Journal de travail 5](#_Toc24448566)

[3. Base de données 5](#_Toc24448567)

[3.1. Conception et modélisation 6](#_Toc24448568)

[3.2. Base de données pendant le développement 6](#_Toc24448569)

[4. Partie C# 6](#_Toc24448570)

[4.1. Déroulement 6](#_Toc24448571)

[4.2. Problèmes rencontrés 6](#_Toc24448572)

[4.2.1. Incompatibilités connecteur MySQL 6](#_Toc24448573)

[4.3. Améliorations possibles 7](#_Toc24448574)

[4.3.1. Gestion d’erreur 7](#_Toc24448575)

[4.3.2. Design 7](#_Toc24448576)

[4.3.3. Menu de paramétrage 7](#_Toc24448577)

[4.3.4. Paramètres dans un fichier config 7](#_Toc24448578)

[5. Partie automate 7](#_Toc24448579)

[5.1. Déroulement 7](#_Toc24448580)

[5.2. Problèmes rencontrés 7](#_Toc24448581)

[5.2.1. Stabilité de la bibliothèque MySQL 7](#_Toc24448582)

[5.2.2. Environnement CodeSys 8](#_Toc24448583)

[5.3. Améliorations possibles 8](#_Toc24448584)

[5.3.1. Bibliothèque MySQL 8](#_Toc24448585)

[5.3.2. Mémoire tampon pour l’envoi d’événements 8](#_Toc24448586)

[6. Planification et suivi 9](#_Toc24448587)

[6.1. Bilal JOHRY 9](#_Toc24448588)

[6.2. Jonny HOFMANN 9](#_Toc24448589)

[6.3. Parties communes 10](#_Toc24448590)

[7. Bilans personnels 10](#_Toc24448591)

[7.1. Bilal JOHRY 10](#_Toc24448592)

[7.2. Jonny HOFMANN 11](#_Toc24448593)

[8. Conclusion 11](#_Toc24448594)

[9. Remerciements 12](#_Toc24448595)

[10. Annexes 12](#_Toc24448596)

# Introduction

## But du document

Ce document est destiné à expliquer les parties importantes du projet ainsi que les méthodes de travail adoptées entre les exécuteurs du travail. Les choix que nous avons dû prendre seront également abordés dans ce présent document. Il expliquera également l’état actuel de l’installation, les problèmes rencontrés lors de la réalisation et des futurs améliorations possibles. Il devra être compréhensible pour tous les types de lecteurs, que ce soit un programmeur aguerri ou une personne sans compétence particulière.

## But de l’installation

L’entreprise « NewIndustry 4.0 » mandate un groupe de techniciens pour faire évoluer un système de transport comportant un certain nombre de postes de travail. L’objectif principal est de développer un gestionnaire de lots avec un système de traçabilité de ces lots.

Il s’agit donc de réaliser un logiciel sur PC proposant à l’utilisateur :

1. Un éditeur de recettes

2. Un gestionnaire de programme journalier

3. Un système de traçabilité des lots.

## Description du travail à réaliser

Dans le cadre de la réussite du module 2 (informatique), il nous a été demandé de réaliser la mise à jour d’une installation « chariot » réalisée dans le cadre du module 1 (automatisation). Pour ce faire, il a fallu adapter le programme automate du module 1 pour qu’il puisse communiquer avec une base de données et exécuter des lots depuis celle-ci. De plus, il doit envoyer des événements (« alarmes » qui se produisent sur la ligne de de production). Il a fallu en outre réaliser un logiciel en C# pour créer des lots et des recettes

## Organisation du projet

Comme nous sommes deux à réaliser ce projet, il a fallu diviser le travail pour maximiser la productivité. Jonny est titulaire d’un CFC Informaticien c’est pourquoi il a apparu naturel qu’il s’occupe de la partie programmation en C#. Bilal est titulaire lui d’un CFC Automaticien. Il est donc logique qu’il prenne en charge la programmation automate. Pour le reste, comme la base de données ou la documentation, nous avons coopérer dans le but d’unifier notre travail et ainsi prévenir les mauvaises surprises. Nous avons décidé de séparer le projet en quatre parties distinctes :

* Base de données (coopération)
* Partie automate (Bilal)
* Partie C# (Jonny)
* Documentation (coopération)

# Déroulement du projet

## Cahier des charges

La première chose que nous avons dû réaliser était le cahier des charges. Pour ce faire, il a fallu analyser le mandat fourni par le responsable du module. De cette analyse, nous avons établi le fonctionnement spécifique de chaque partie du projet. Nous avons clairement délimité les fonctionnalités de l’installation et nous nous baserons sur ce document pour toute la réalisation du projet. Le cahier des charges contient les fonctionnalités du programme C# ainsi que celles du programme automate. Une première version de celui-ci a été vérifiée par un superviseur et a émis des modifications à effectuer.

## Organigramme

Ensuite, nous avons établi un organigramme du travail à réaliser. Le travail a été divisé en 4 activités distinctes qui elles-mêmes ont été divisées en tâches spécifiques. Chacune de ces tâches a ensuite été décrite minutieusement dans le document contenant la planification du projet. L’organigramme ainsi que la description des tâches sont consultables dans le document « planification du module 2 » au point 2. Etablir l’organigramme a été un point très important car tout le travail qui va suivre se basera sur cette division du travail.

## Planification

Après avoir séparé le travail d’une manière distinctes et attribuer les différentes tâches aux différents collaborateurs du projet. Il a fallu estimer la durée de chaque tâche, c’est pourquoi nous avons réalisé un diagramme de GANTT pour organiser le travail et assurer un suivi d’évolution du travail. Cette planification est consultable dans les annexes (annexe C).

## Base de données

Nous avons pris le temps de bien d’analyser tout ce qu’il fallait stocker sur la base de données. C’est une des parties les plus importantes du projet car c’est la base de données qui va déterminer ce que va enregistrer l’automate ou encore ce que devra faire le programme C# pour interagir avec celle-ci. C’est la base de données qui régi les interactions entre elle et les divers participants (automate et ordinateur). Nous avons alors développé le concept de la base de données sur le logiciel libre « JMerise ». Ensuite nous avons implanté dans un système de gestion de base de données nommé « phpMyAdmin ».

## Partie C#

Le programme C# a besoin d’une interface utilisateur. Le mandat ne spécifiait pas d’interface spécifique. Dès lors, nous avions le choix entre une interface console ou une interface graphique. Nous avons opté pour le deuxième choix car Jonny maitrisait bien cette technique et Bilal souhaitait découvrir ce type d’interface. Une fois que la base de données a été validée par le responsable du module, Jonny a pu commencer à développer le programme en C# à l’aide du logiciel « Visual Studio Community 2017 ». Ce développement a bien été encadré par le cahier des charges et tout le reste de la documentation. Tout d’abord, il a créé l’interface graphique pour avoir une idée de l’aspect graphique. Ensuite, il s’est attaqué à la programmation de chaque fonctionnalité. Enfin, un test général de fonctionnalités a été effectué pour s’assurer que le programme répondait bien au cahier des charges.

## Partie automate

En ce qui concerne la partie programmation automate, c’est différent de la partie C# car le programme faisant fonctionner l’installation « Chariot » a déjà réalisé dans le cadre du module 1 (automatisation). Cependant, ce projet demande des modifications importantes du code car il faut l’adapter à la base de données et cela demande des changements radicaux. Une fois la base de données validée, Bilal a commencé à modifier le code à l’aide du logiciel « CodeSys V2.3 ». Tout comme la partie C#, le développement a également été encadré par le cahier des charges. Il a débuté par créer les nouvelles visualisations et ensuite, il a adapté le programme pour que celui-ci corresponde à ce qui était demandé dans le cahier des charges.

## Documentations

Nous avons réalisé une documentation technique contenant toutes les informations nécessaires à la compréhension des programmes ainsi que de la base de données.

Un rapport technique a également été rédigé relatant tous les faits qui se sont produits lors de la réalisation de ce projet tels que des erreurs ou encore les améliorations à apporter dans le futur.

## Journal de travail

Un journal de travail propre à chaque collaborateur a été tenu de telle sorte à retracer toutes les actions réalisées par les différents acteurs du projet lors des séances scolaires dédiées au projet ainsi que le travail effectué à la maison. Les journaux de travail sont consultables dans les annexes de ce document (annexes A et B).

# Base de données

## Conception et modélisation

La conception de la base de données se faisait en coopération pour gérer correctement les besoins des deux programmes qui devront y accéder. La modélisation a été faite avec le logiciel JMerise dans la version étudiante gratuite.

Le modèle conceptuel a été corrigé par Monsieur Montavon et par la suite corrigé selon nos besoins. Pour chaque modification effectuée, le fichier JMerise, le script de la création de la base de données ainsi que l’image du MCD ont été actualisés en même temps sur GitHub pour éviter d’avoir des fichiers pas à jour.

## Base de données pendant le développement

Durant le développement du programme automate, la base de données mise à disposition par l’école a été utilisée. Pour le programme C#, une base de données locale a été préférée en raison de la mobilité. Ceci a créé peu de problèmes. Cependant, les statuts n’étaient pas dans le bon ordre sur la base de données locale une seule fois.

Dès qu’une modification sur la base de données était nécessaire ou effectué, la personne qui l’a faite informe l’autre pour que toutes les base de données, locales ou non, soient mises à jour.

# Partie C#

## Déroulement

Bilal avait créé un petit test d’interface graphique pour l’accueil que j’ai (Jonny pour le reste de la partie C#) utilisé pour la suite du programme. J’ai passé beaucoup de temps dans l’apprentissage de l’interface graphique et surtout dans le gestionnaire de base de données *DBManager*. Dès que les fondements dans *DBManager* étaient programmés, j’ai appris les bases de WinForms et tout s’est déroulé plus rapidement.

C’est à cause de cela que la planification est un peu décalée, mais finalement, le travail a été effectué.

Une structure simple possédant la possibilité de facilement être agrandie et modifiée. Ceci est expliqué plus en détail au point *4.1 Structure du programme* de la documentation technique.

## Problèmes rencontrés

### Incompatibilités connecteur MySQL

Le connecteur MySQL qu’on utilise en cours avec Monsieur Conus ne fonctionne pas sur Visual Studio 2019 que j’utilisais sur mes ordinateurs personnels. Des changements de versions, réinstallations et tests ont dû être effectués pour connaitre la meilleure combinaison correspondant au projet.

Il y aurait eu la possibilité d’utiliser le connecteur ODBC qui est compatible avec Visual Studio 2019. Cependant, Bilal n’est pas informaticien, et nous n’avons pas vu ce type de connecteur en cours.

## Améliorations possibles

### Gestion d’erreur

La gestion d’erreurs et d’exceptions n’est pas encore totalement fonctionnelle, certaines entrées peuvent encore créer des crashs de programme. Chaque erreur ou bug trouvé est ajouté dans GitHub sous *issues*.

### Design

Le design est en ce moment minimaliste mais fonctionnel. L’interface en générale manque de capacité à s’adapter à des grandeurs de fenêtres différentes.

Ajouter des règles à l’interface pour la rendre visuellement plus parlante serait envisageable également.

### Menu de paramétrage

Un menu pour changer les paramètres comme par exemple, les paramètres de connexion à la base de données, serait de grande utilité mais le temps ne nous l’a pas permis car ce n’était pas une priorité.

### Paramètres dans un fichier config

Sauvegarder les paramètres dans un fichier config serait très utile et serait fait en même temps que le menu de paramétrage. Il faudra faire attention à éviter de sauvegarder le mot de passe en claire pour des questions de sécurité.

# Partie automate

## Déroulement

Avant tout, Bilal a repris l’ancien programme qui avait été réalisé dans le cadre de la réussite du module 1 (automatisation). Ensuite, les visualisations ont toutes été créées ou modifiées pour qu’elles correspondent aux spécificités du module 2 (informatique). Dès lors, tous les éléments étaient réunis pour commencer à programmer les fonctionnalités. Bilal a en premier lieu créer un programme principal qui s’occupe de toutes les interactions avec la base de données. En effet, c’est lui qui connecte, déconnecte, lit ou écrit sur la base de données. Après cela, j’ai créé plusieurs autres programmes dédiés à des tâches spécifiques comme la récupération des lots, des recettes ou encore l’envoi d’événements vers la base de données. Tous ces sous-programmes interagissent avec ce programme principal pour communiquer avec la base de données. Tous les programmes dédiés au module 2 (informatique) sont expliqués dans la documentation technique au point 3.5.

## Problèmes rencontrés

### Stabilité de la bibliothèque MySQL

J’ai rencontré beaucoup de problèmes de stabilité avec l’utilisation de cette bibliothèque Wago. En effet, au début, plusieurs erreurs concernant des problèmes de syntaxe MySQL apparaissaient ce qui impliquait que l’automate ne pouvait pas récupérer les informations depuis la base de données. Dès lors, il était impossible de laisser ce problème de côté car si la base ne fonctionne pas, il est alors impossible de continuer la programmation. Le bloc fonction de lecture mentionnait qu’il y avait une erreur de syntaxe. Cependant, après plusieurs lectures et relectures par moi-même et les enseignants ainsi que nombreuses heures d’essais et de tentatives pour comprendre ce qui posait problème, j’ai identifié que c’était le « ; » en fin de requête qui faisait bugger le bloc fonction. En effet, j’avais mis un « ; » à la fin de chaque requête et cela engendrait une erreur dans 3 cas sur 10 ce qui en faisait un problème très difficile à comprendre. Pourtant, rien n’était mentionné dans la documentation de la bibliothèque. Ce problème a nécessité 6 heures pour le résoudre. C’est un temps relativement important pour un projet de cette ampleur.

### Environnement CodeSys

Un autre problème majeur est survenu lors de la programmation de l’automate. Celui-ci concernait directement l’automate car celui-ci s’éteignait de manière intempestive et il fallait alors recharger tout le code dans l’automate. Ce problème est apparu en milieu de projet et aucune raison ne peut expliquer son apparition. En effet, la veille, le programme fonctionnait de manière correcte et c’est le lendemain en matinée que le problème est survenu. Il a été très compliqué d’identifier la cause de cette anomalie car comme l’automate s’éteignait. L’environnement de programmation « CodeSys » ne mentionnait pas le code d’erreur étant donné que la liaison était rompue. Dès lors, j’ai immédiatement demandé de l’aide au responsable de l’automatisation au sein de l’Ecole Technique. Nous avons alors passé plusieurs heures à essayer d’identifier la cause de l’erreur mais en vain. Après beaucoup de changement et de tests, l’erreur s’est corrigée d’elle-même. Personne n’a été en mesure de me dire la raison de cette erreur. Cependant, je pense qu’il y a eu une erreur de compilation au niveau d’une bibliothèque interne à l’automate. Ce problème m’a demandé plus de 8 heures pour essayer de la résoudre.

## Améliorations possibles

### Bibliothèque MySQL

Comme expliqué au point 5.2.1, la bibliothèque que nous avons utilisée a été conçue d’une manière totalement amateure. Je m’explique, la documentation est très basique et n’explique pas comment interagir de manière correcte avec les blocs fonctions. En effet, il est décrit de manière très sommaire le fonctionnement du bloc fonction ainsi que ses erreurs possibles. Il n’y a pas de manuel utilisateur digne de ce nom. Je pense que c’est entre autres à cause de cette fragilité que le problème de syntaxe est apparu. La facilité d’utilisation est remise également en cause car on pourrait rassembler deux fonctions en une seule.

Si un jour il devenait nécessaire de communiquer avec une base de données à l’aide d’un automate Wago dans un milieu professionnel, il faudrait, à mon sens, utiliser une autre bibliothèque MySQL réalisée par une autre société ou un autre organisme. Pourquoi ne pas la développer soi-même ou mandaté un informaticien pour s’assurer de la stabilité de la bibliothèque ainsi que de la facilité d’utilisation.

### Mémoire tampon pour l’envoi d’événements

Actuellement, l’envoi d’événements se fait au cas par cas. En effet, dès qu’il y a un événement à envoyer ou une valeur à mettre à jour, l’automate se connecte à la base de données et envoie ce qu’il y a à envoyer. Le problème avec cette technique, c’est que dès que l’installation est en production et que l’on déconnecte la base de données, les informations ne sont plus envoyées. Une fois la base de données reconnectée, l’automate enverra chaque type d’événements qui se sont produits durant la coupure mais la date réelle à laquelle l’événement s’est produit sera fausse puisqu’elle correspondra à la date ou l’événement a été ré-envoyé après la reconnexion. Un autre problème majeur est qu’un type d’événement ne peut être envoyé qu’une seule fois. Par exemple, si la barrière lumineuse droite a été coupée cinq fois durant la coupure, il n’y aura qu’un seul événement « barrière droite coupée » et en plus la date sera fausse.

Pour résoudre ce problème, il suffirait de remplacer ce système par une mémoire tampon prénommée « buffer ». Rien de très compliqué car cette mémoire tampon consiste à remplir un tableau au fur et à mesure que les événements apparaissent. Grâce à cela, fini les pertes d’informations quand l’installation est déconnectée de la base de données car les événements sont immédiatement stockés dans le tableau dans l’attente d’être transmis à la base de données à la prochaine reconnexion.

# Planification et suivi

## Bilal JOHRY

C’était à moi de m’occuper de tout ce qui concernait la partie automate programmable industriel. C’est donc moi qui ai approximativement estimé le temps nécessaire aux tâches concernant l’automate. Assez compliqué au départ car les tâches consistaient principalement à communiquer avec la base de données et je n’avais encore jamais expérimenté de telles fonctions. J’ai donc utilisé l’expérience que je possède en automatisation pour estimer au mieux.

Quand on observe maintenant la planification avec son suivi, on aperçoit que la tâche « Communication avec la base de données » a nécessité environ trois heures de moins que ce que j’avais prévu. En ce qui concerne la tâche « Lecture depuis la base de données », elle m’a pris environ deux heures de plus notamment à cause du problème de bibliothèque MySQL (voir le point 5.2.1 de ce document pour plus de précision concernant ce problème). Contrairement à la lecture, la tâche « Ecriture vers la base de données » a été terminée avec six heures d’avance par rapport au temps estimé initialement. Enfin, la tâche « Tests et dépannages » m’a pris beaucoup plus de temps car il y a eu un problème qui a nécessité de nombreuses heures de tests (voir le point 5.2.2 de ce document pour plus de précision concernant le problème).

Malgré les quelques problèmes survenus lors de ce projet, la partie automate avait été estimée à 60 heures, j’en ai eu besoin de 57. C’est 3 heures de moins. Le travail de planification a bien été réalisé.

## Jonny HOFMANN

C’était à moi de m’occuper de la majorité du programme C#, j’ai donc fait les approximations des temps nécessaires pour les différentes tâches. Ce ne fut pas facile et finalement pas assez précis parce que la partie “connexion à la base de données” contenait la création de toute la classe *DBManager* qui fut la classe la plus complète et avec les fonctions les plus complexes.

La première interface a également pris plus de temps que prévu pour cause le manque de connaissances et de méthodes de travaille avec des objets d’interface C#. Dès que la première page a été finie, les autres ont suivi très rapidement.

Le programme C# a demandé beaucoup de travail à effectuer pendant les vacances d’automne parce que coder sur sa propre machine avec ses outils et ses écrans est beaucoup plus agréable. Durant ces vacances j’ai rattrapé suffisamment de travail pour ne pas être trop stressé à la fin.

## Parties communes

Au début de ce projet, quand on nous a demandé de réaliser une planification avec une estimation des temps pour chaque tâche, nous savions qu’elle allait être très compliquée à établir et surtout à suivre. En effet, ce travail comporte beaucoup de fonctionnalités peu connues pour Bilal au niveau de la programmation et pour Jonny ces méconnaissances se situent au niveau de la partie automate. De plus, ce travail devait se faire par deux ce qui complique la tâche de la planification. La clé de la réussite se trouvait notamment dans la communication avec une répartition des tâches claire et précise.

Malgré les difficultés liées à l’estimation du temps que prendra chaque tâche, nous les avons relativement bien estimés. Dans l’idéal, nous avions prévu de réaliser toute la documentation au fur et à mesure que le projet avançait, mais ça n’a été qu’une utopie. En effet, nous ne disposions de pas beaucoup de temps pour réaliser ce projet et il y avait beaucoup de matière à produire. Nous avons donc mis un accent sur la programmation (automate et C#) dans le but de faire fonctionner l’installation pour la présentation. Une fois la défense orale passée, nous nous sommes attaqués au rapport technique ainsi qu’à la documentation technique.

Dans l’ensemble, notre planification a relativement bien été suivie malgré les quelques problèmes qui nécessités plus de temps qu’estimé. C’est toujours compliqué d’évalué le temps que prendra telle ou telle tâche.

# Bilans personnels

## Bilal JOHRY

Quand j’ai pris connaissance de tout le contenu de ce projet et le temps que nous avions à disposition, j’ai instinctivement pensé qu’il ne faudrait pas traîner ou s’attarder sur des petits détails. Mais au fur et à mesure que la programmation avançait, je prenais confiance en moi et je donnais tout pour essayer de réaliser ce qui était demandé dans le cahier des charges.

C’est lors de l’écriture de ce bilan personnel que je me suis rendu compte que ça fait presque trois mois que ce travail m’occupe. Ce qui est amusant dans la réalisation d’un tel travail, c’est de partir de pas grand-chose et de voir le résultat final.

Je pense que le fait d’être face à un projet d’une ampleur assez conséquente sur un temps dédié aussi court est très important pour la suite de ma vie professionnelle.

Je suis finalement divisé entre un sentiment de satisfaction et de frustration. En effet, je suis très ravi du travail accompli car le rendu est plus que correcte à mes yeux. En même temps j’aurais tellement voulu finir ce projet avec une qualité industrielle mais malheureusement le temps ne me le permettait pas.

Enfin, j’ai beaucoup apprécié travailler avec Jonny. En effet, il m’a apporté des connaissances d’un autre métier qui auront permis d’avoir une perception différente lors de mes futurs travaux.

## Jonny HOFMANN

Quand j’ai pris connaissance du projet et surtout que nous pouvions faire le programme C# en interface graphique, j’ai été très motivé de pouvoir réaliser un logiciel graphique complet qui pourrait théoriquement, avec un peu plus de temps, être utilisé dans une entreprise.

J’ai remarqué mon manque d’expérience avec des projets de programmation de cette ampleur. J’aurais dû utiliser beaucoup plus d’outils comme le met à disposition GitHub, par exemple la gestion des erreurs (issues) . La puissance et la simplicité des outils GitHub sont souvent sous-estimées. Une petite introduction durant un cours de programmation pourrait être utile pour beaucoup de personne selon moi.

J’ai également profité de ce projet et du programme C# pour me forcer à prendre de meilleures habitudes de programmation, comme par exemple bien commenter au fur et à mesure, et de ne pas « push » (envoyer) sur GitHub du code non documenté.

J’ai apprécié travailler avec Bilal parce que nous avons des compétences qui se complétaient bien pour ce projet et des connaissances professionnelles qui varient, ce qui donne toujours des autres points de vue sur certains sujets et c’est cela qui nous a permis de prendre les bonnes décisions.

Il est également une personne plus organisée que moi ce qui m’a permis d’améliorer mon organisation et ma gestion de temps. Ce sont mes plus grands points faibles dans ce genre de projets.

# Conclusion

Après plus de 221 heures de travail, nous pouvons enfin tirer les conclusions de ce module 2 (informatique). Le travail concernant ce module était clairement axé sur l’informatique avec notamment le développement d’un logiciel de gestion des lots et également sur l’automatisation avec la modification du programme dans le but que ces deux programmes puissent « communiquer » au travers de la base de données. En ce qui concerne ces points, nous n’avons rien à redire. En effet, toutes les fonctionnalités mentionnées dans le cahier des charges ont été respectées.

Le programme C# est fonctionnel, simple à utiliser, dispose d’une panoplie de filtres permettant de sélectionner les lots à afficher et de plus, il a été réalisé avec une interface graphique. Ce dernier point n’était pas spécifiquement demandé mais nous avons trouvé qu’il était plus clair de procéder de cette manière et cette méthode transférerait des connaissances entre les corps de métiers (un automaticien n’a en principe jamais réalisé d’interface graphique).

En ce qui concerne le programme automate, il est lui aussi très simple à utiliser et dispose d’une ribambelle de visualisations réalisées avec beaucoup de soin permettant à l’utilisateur d’accéder à toutes les informations nécessaires d’une manière intuitive.

Si le projet était à refaire, nous ne changerions probablement pas beaucoup de chose car nous sommes très satisfaits du résultat. En effet, toutes les fonctionnalités ont été réalisées de manière professionnelle et de plus, nous avons ajouté plusieurs autres fonctions qui n’étaient pas demandées comme par exemple l’interface graphique du programme C# ou encore la gestion des erreurs de communications dans le programme automate. La seule chose que nous aurions pu faire autrement est la documentation. Nous aurions dû la réaliser durant tout le projet mais cela est beaucoup plus facile à dire qu’à faire.

# Remerciements

Nous tenons à remercier Monsieur Caillet pour le temps qu’il nous a accordé à chercher des erreurs propres à l’automate. Sans son aide, nous n’aurions probablement pas réussi à trouver la cause des problèmes et aurions sans doute perdu énormément de temps.

Nous remercions Monsieur Montavon pour le temps qu’il a accordé à répondre à nos questions et tenter de résoudre nos problèmes. C’est également lui qui a corrigé notre cahier des charges et validé notre base de données.

Enfin, nous remercions Monsieur Grüter pour nous avoir appris à réaliser toute la documentation.

De plus, toutes ces personnes ont été très présentes pour nous donner de précieux conseils.

# Annexes

**Annexe A** : Journal de travail – Bilal JOHRY

**Annexe B** : Journal de travail – Jonny HOFMANN

**Annexe C** : Planification et suivi