SANCHEZ Hugo
ROUSSEL Emma
TDM

LIVRABLE PROBLÈME

# Informations importantes :

**Veuillez prendre connaissance de ces informations avant de faire tourner notre solution.**

## Version du logiciel

Le projet a été réalisé sous Visual Studio Community 2022, avec l'option “Développement.NET Desktop” pour pouvoir créer des Windows Forms WPF.

## Références et Packages requis

Afin de réaliser ce problème nous avons stocké nos données sur une base de données MySql.

- La dll Mysql.Data.dll doit être ajoutée en tant que référence projet (Projet >> ajouter une référence >> sélectionner la dll)
- Il faut cliquer sur le nom de la solution dans l'explorateur de solutions >> Gérer les packages NuGet >> Parcourir, puis il faut installer les packages suivants pour que Mysql fonctionne : Renci.SshNet.Async et SSH.NET

## Diagramme UML

Nous avons ajouté un diagramme UML à l’aide du module Diagramme de classes pour mieux visualiser la conception et les liens entre les classes.
Il faut installer le composant Concepteur de classes :

- ouvrir le programme d’installation de Visual Studio via le menu Démarrer de Windows
- sélectionner l’onglet Composants individuels puis le catégorie Outils de code
- sélectionner Concepteur de classes, puis Modifier

Le diagramme en format image est dans le dossier rendu.

## Publication de la solution

La solution a été générée dans le dossier ‘Build’ de notre rendu pour pouvoir le tester sans encombre.

## I. Evolution du programme

La première chose que nous avons faite a été de synthétiser et de structurer toutes les attentes et contraintes de l'énoncé. Nous avons regroupé chacunes des instructions sur un schéma, puis les avons traduites par leur interprétation en C# avec les commandes et programmes dont nous allions avoir besoin. Cela nous a permis d'avoir une vue d'ensemble sur le problème assez rapidement, et de commencer à le structurer par classes. Pour certaines consignes, nous avons dû réfléchir plus longtemps afin de trouver la meilleure solution, pour l'affichage de l'organigramme par exemple, qui nous a posé soucis, ou encore pour le choix ou non de classes héréditaires pour les véhicules. Cette réflexion nous a permis d'ajouter des fonctionnalités non exigées qui nous paraissaient judicieuses.

Ensuite, nous avons commencé à coder. Nous avons pris certains TD en support, comme le TD11 et 12 pour créer notre algorithme de Dijkstra afin de connaître le chemin le plus court entre deux villes.

Enfin, nous avons vérifié, testé et amélioré notre programme. Nous avons à nouveau passé en revue l'énoncé pour vérifier que nous n'avions rien omis.

## II. Problème(s) rencontré(s)

Afin de réaliser l'organigramme, nous nous sommes inspirés du TD 10 pour l'affichage en arborescence et du TD 11 pour les arbres n-aire. Or, nous ne pouvons instancier les nœuds et leurs liens directement dans le code car le nombre d'employés et leur poste peuvent changer à tout instant. Il nous a donc fallu passer par une liste de 'NoeudOrganigramme' que nous remplissons en parcourant la base de données. Afin d'afficher l'organigramme, nous avons modifié la fonction 'AffichageOffset' et la fonction 'AffichageArborescence' afin de récupérer le résultat dans une seule chaîne de caractère afin de l'afficher dans un MessageBox.Nous n'avons hélas pas réussi à afficher un organigramme correct car l'affichage ne se transmet pas correctement dans le MessageBox.

## III. Ajouts et spécificités de notre programme

**Affichage**
Nous avons utilisé le module Windows Form WPF pour créer la fenêtre d'affichage du programme avec une véritable interface.

**Statistiques et données**
Nous avons choisi de ne pas faire un menu 'Statistiques' mais de les inclure là où elles sont pertinentes dans l'un des trois menus principaux (Gestion des clients/commandes/salariés). Nous avons inclus un système de réduction clients en fonction de leur fidélité : on leur attribue un taux de remise permanente pour toutes leurs commandes en fonction de leur classement par Montant des Achats Cumulés. Le salaire des chauffeurs évolue dans le temps en fonction du salaire d'embauche et de l'ancienneté.

**Modification / Suppression des clients / salariés**
Il suffit de faire un clic droit sur le profil de la personne et de choisir l'option voulue, nous avons trouvé que c'était la solution la plus simple et la plus intuitive.

**Ajout de ville**
Nous avons la possibilité d'ajouter autant de villes que possible, en y indiquant ville A et une ville B, la distance entre les deux, ainsi que le temps de trajet nécessaire pour aller de la ville A à la ville B (fichier NewVille.cs dans le programme). Nous avons ajouté quelques distances et durées que nous estimions pertinentes pour raccourcir le chemin à parcourir en plus de celles données dans le fichier Distances.csv .

**Véhicule**
Nous avons choisi d'utiliser des classes héréditaires pour les véhicules. Lors de la création d'une commande, il faut sélectionner la marchandise à transporter par le véhicule, ce qui permet de déterminer lequel sera le plus adapté à la livraison. Il est possible d'ajouter des informations qui serviront à la description de la commande.

**SQL**
Nous voulions stocker nos données et pouvoir y accéder depuis n'importe quelle plateforme une fois la solution générée et publiée. Avec un fichier CSV, il y avait trop de contraintes (une seule personne de l'entreprise peut utiliser le programme, donc les chefs d'équipes doivent passer par la même personne pour organiser l'emploi du temps de leurs salariés, ect), c'est pour cela que nous avons décidé de stocker nos données dans une base de données. Avec l'utilisation d'une base de données, l'utilisation d'interface n'était pas nécessaire grâce aux requêtes SQL nous permettant de facilement trier nos éléments. Une interface 'IComparable.cs' est cependant présente, que nous avions commencé à coder avant de changer d'avis, mais non utilisée dans notre solution.

La base de données (BDD) a été créée sur un serveur linux avec MariaDb tournant sur un Raspberry Pi. Un accès distant a été créé ainsi que l'ouverture des ports nécessaire pour autoriser l'accès à la BDD depuis n'importe quel endroit.

# IV. Conclusion

Ce problème a été pour nous un réel défi, et nous a permis de travailler de nouveau presque toutes les notions vues depuis le début de l'année. Nous avons également réalisé qu'avec un peu plus de temps, nous aurions été capables de réaliser une vraie application de gestion pour une société de transport (sans prendre en compte l'aspect sécurité de notre programme), ce qui nous a nous même impressionné.