# Revue de projet : TransConnect®

Charles DUVAL – Martin RAMPONT

## Explications des classes de notre code :

- **Les contrôleurs :**
  - ***AffichageGraphique :*** la classe manipulant le terminal dans le but d'afficher les menus et les différentes interactions avec l'utilisateur. Elle est l'User Interface de notre code.
  - ***Manager :*** à la manière d'un jeu développé sur Unity, manager est ici la classe manipulant tous les objets de notre code. Il dispose d'une liste d'employée, une liste de client, une liste de véhicule et une liste de commande en cours. Le but de cette classe est donc de simuler la gestion de l'entreprise dans notre code.
  - ***LireFichier :*** la classe I/O de notre code. Elle lit les fichiers correspondant à client.csv, employee.csv et distance.csv afin de sauvegarder les données entre chaque session.
  - ***Program :*** Classe principale contenant un Main dont le principal appel est à la méthode MainAffichage() de AffichageGraphique.
- **Les objets abstraits :**
  - ***Personne :*** Classe abstraite définissant une personne dans le sens de l'énoncé
- **Les objets :**
  - ***Vehicule :*** classe mère à tous les véhicules. Définit l'immatriculation (=idVehicule) du véhicule. Choix *à postériori* de ne pas mettre véhicule en abstract car besoin de créer des instances temporaires de Vehicule lors de la génération de commande.
  - ***Voiture :*** une simple voiture transportant des passagers
  - ***Camion :*** classe mère des différents types de camions. Définit une capacité et des matières transportées
  - ***Camion-benne :*** un camion avec divers équipements
  - ***Camion-citerne :*** un camion avec un type de cuve selon ce qu'il transporte
  - ***Camion-frigorifique :*** dispose d'un nombre de groupe électrogène permettant une certaine « autonomie frigorifique ». La livraison est impossible si la distance (Dijkstra) dépasse l'autonomie
  - ***Client :*** la classe correspondant au client. Elle hérite de Personne et permet de récupérer et de traiter des informations basiques sur le client en lui-même (comme ses commandes archivées)
  - ***Commande :*** représente une commande à venir ou archivée et ses informations
  - ***Salarie :*** classe représentant les salariés de l'entreprise. Elle ne dispose pas de hiérarchie incorporée directement
- **Gestion des graphes :**
  - ***Arete :*** Représente une étape de l'itinéraire et les infos associées (distance, durée)
  - ***Graphe :*** Choix d'un graphe orienté (si travaux dans un seul sens par exemple). Représenté par une liste de sommets (villes) et une liste d'arêtes.
- **Gestion des arbres :**
  - ***SalariesArbre :*** Classe représentant la hiérarchie sous forme d'arbre n-aire de l'entreprise. Dispose de toutes les fonctions demandées dans le sujet
- **Les interfaces :**
  - ***IId :*** interface obligeant l'intégration d'une id aux classes dont elle est héritée.
  - ***IPrix :*** interface obligeant l'intégration d'un quelconque prix aux classes dont elle est héritée.

## Les Bases de Données (BDD) :

Afin d'optimiser le temps pour proposer des fonctionnalités avancées, nous avons décidé de n'utiliser la sauvegarde externe de BDD uniquement pour certains objets. En voici un récapitulatif.

**Par exemple :** les commandes fonctionnent en deux temps : les commandes saisies via l'interface à venir ne sont pas sauvegardées en externes tant qu'elles n'ont pas été archivées (= on migre la commande des commandes à venir vers l'historique des commandes propre à chaque utilisateur). Cela évite notamment des problèmes d'enregistrement d'itinéraires des commandes en cours dans le CSV et le rechargement de ceux-ci au démarrage de la console (car un itinéraire peut ne plus exister au nouveau lancement si on modifie distances.csv)

| BDD internes = List C# reset à chaque lancement | BDD externes = CSV réutilisés à chaque lancement |
|---|---|
| Client | Client |
| Salariés / Chauffeurs | Salariés / Chauffeurs |
| Véhicules | |
| Commandes (à venir) | |
| Commandes (archivées) | Commandes (archivées) |
| Distances | Distances |

## Les spécificités de nos modules :

- 👫 **Module Client :**

  Ce module permet de traiter des clients. On peut notamment y ajouter un client, le modifier ou le supprimer. En sus, il permet aussi d'afficher les clients dans un ordre trié selon les données qui lui sont fournis. Enfin, toutes les modifications faites sont en temps réel apporté sur la BDD client, afin de garder en mémoire les changements apportés.

- 💼 **Module Salarié :**

  Ce module utilise l'architecture d'arbre n-aire pour traiter des salariés de l'entreprise. En dehors des méthodes ajouter, modifier et enlever demandées, écrivant toutes dans le BDD en temps réel, il est aussi possible d'afficher sous différentes formes notre organigramme : via un affichage d'arbre inspiré de l'arbre de recherche de Windows (commande tree dans le PowerShell), sous la forme d'une liste d'adjacence, sous la forme d'une relation employée/ employeur et finalement en énumération classique sans ordre prédéfini en dehors de leur ordre d'ajout. Tout cela est aussi sauvegardé en live dans notre BDD.

- 🎁 **Module Commande :**

Ce module repose essentiellement sur un algorithme de Dijkstra avancé qui gère le cas des chemins impossibles à relier ainsi que les livraisons au sein d'une même ville.

Notez l'affichage dynamique du plan de route 😊 : le nombre de | | entre chaque ville est proportionnel à la distance réelle entre les villes. Les distances et durée du point de départ jusqu'à chaque étape sont également indiquées.

Un module (beta) permet également de modifier une commande déjà crée et ajuste automatiquement l'itinéraire en fonction du nouveau départ, nouvel arrivée, nouvelle date de livraison. Attention à l'orthographe exacte des villes. Exemple : Paris, La Rochelle, Lyon…

Une commande ne peut être généré si aucun chauffeur ou véhicule n'a de disponibilité pour la date de livraison. La commande est également annulée si l'autonomie frigorifique dans le cas du camion frigorifique (dépendant de son nb de groupe électrogène) ne couvre pas la distance totale du trajet générée par Dijkstra.

Enfin une gestion des identifiants de commandes à été mise en place afin d'éviter les doublons.

- 📈 **Module Statistiques :**

Affiche les statistiques demandées (essentiellement sur les commandes archivées) à l'exception des moyennes de prix qui sont affichées pour les commandes archivées et à venir.

- 🚚 **Module Véhicule (cf. Module Autre) :**

Ce module permet de gérer la flotte de véhicules. Le modèle de chaque véhicule est différent et fonctionnel. Par exemple dans le cas du calcul du prix d'une commande, non seulement la distance totale est prise en compte mais aussi les spécificités du type de véhicules choisi. Ainsi le prix va également dépendre du nombre d'équipements (camion benne), du nombre de groupe électrogène (camion frigorifique) et de la capacité du véhicule suivant les cas.

De plus, chaque véhicule possède son propre planning de dates où il a été / sera utilisé pour livrer une commande. Un véhicule ne peut pas prendre 2 commandes à la même date.

## Commentaires sur les idées implémentés dans le module autre :

- **Cryptographie :** possibilité de chiffrer et déchiffrer les BDD. Pour plus de sécurité, la clef est stockée dans un fichier texte crypté par Windows directement via l'usage de File.Encrypt(). Cependant, comme cela nécessite Windows pro pour être utilisé, les codes affiliés au cryptage de la clef sont par défaut en commentaire. A noter aussi que, une fois les BDD chiffrées, il est nécessaire de quitter le programme via le choix «6. Quitter » de la page principale , sans quoi les modifications ne seront pas sauvegardées entre chaque exécution du programme.
- **Promouvoir un salarié :** Il est déjà possible de rajouter un salarié dans la base de données en précisant son supérieur. Cependant, cela ne permet pas de le promouvoir ensuite. Via la fonction « promouvoir un salarié », il est possible de changer la position d'un salarié dans l'arbre en indiquant qui est son nouveau supérieur, et qui est son nouvel employé direct. L'employé direct doit être un ancien employé direct de son nouveau supérieur, sans quoi le programme ne fonctionnera pas.
- **Module véhicule :** permet une gestion plus simple de la flotte de véhicules directement depuis la console avec une implémentation personnalisée et dynamique pour chaque véhicule. Les spécificités de chaque véhicule interagissent donc avec le code à plusieurs endroits (cf. ci-dessus).
- **Une animation de bienvenue :** « Bienvenue chez Transconnect ». Le Défilement de ce texte repose sur un système de Threading

## Les autres bonus :

- Une interface détaillée et intuitive avec un système de navigation entre les menus
- Un guide lors de la création des différentes instances
- De nombreuses fonctions et vérifications empêchant les erreurs d'exécutions dues aux entrées utilisateurs erronés ou bien des créations d'objets non conformes
- De nombreuses délégations et collections génériques
- Et bien d'autres à découvrir…