Projet C# S6: TransConnect

Maxime LANGELIER

31 mai 2024

1 Introduction

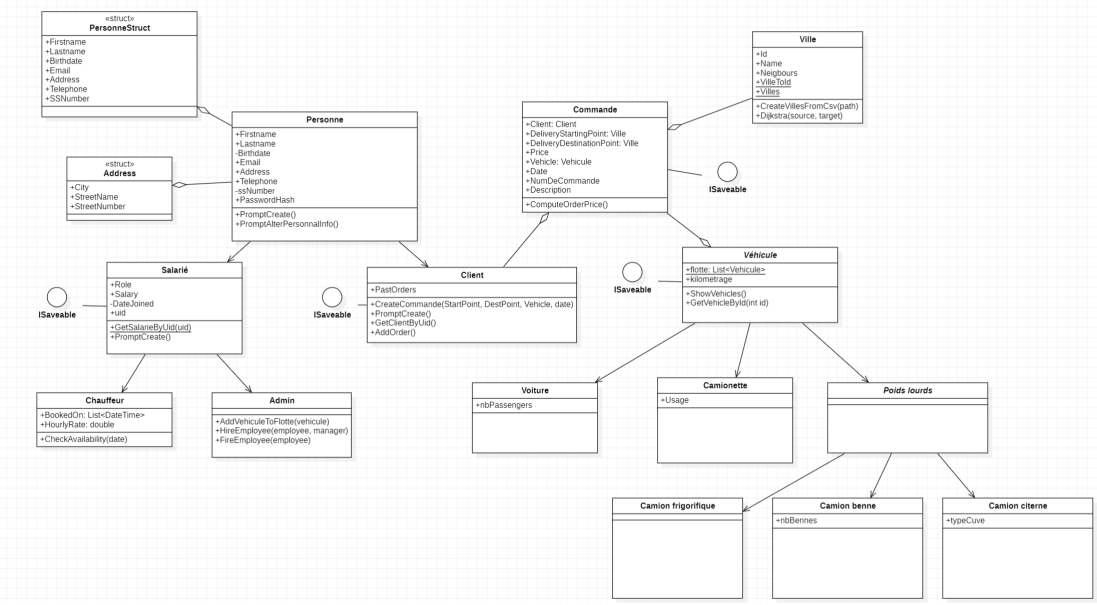
Le programme consiste en une succession de menu déroulants qui peuvent être controlés à l'aide des flèches du clavier. La touche flèche du bas permettant d'aller vers le bas et réciproquement. Pour sélectionner une option, on utilisera la touche entrée.

Le programme nécessite à un client de s'identifier ou de s'enregister en utilisant son addresse email en tant qu'identifiant ainsi que son mot de passe défini à la création du compte.

Afin de pouvoir tester le programe dans de bonnes conditions, des couples identifiants/ mot de passe sont renseignés ci-dessous :

— Client : `c.grolet@gmail.com : CG123`
— Salarié (Classique) : `ccouleur@transconnect.com : defaultPassword`
— Salarié (Chauffeur) : `tromu@transconnect.com : defaultPassword`
— Salarié (Admin + PDG) : `jdupond@transconnect.com : defaultPassword`

Ci-dessous le diagramme UML de la partie orientée objet de notre programme. Il est possible que les signatures des méthodes ne soient pas entièrement à jour, cependant la structure du programme et les relations entre les classes sont conservées :



2 Le module Client

Le module client, comme le module salarié, fonctionne sous forme de compte. Au lancement du module, l'utilisateur a l'option de se connecter ou de s'enregistrer. L'enregistrement est la seule manière de créer un compte client. Si le client choisit de s'enregister, il fera alors appel à la méthode `PromptCreate()`, et des messages console le feront renseigner ses informations personnelles. S'il possède déjà un compte Client et décide de se connecter, il lui faudra alors renseigner son email et son mot

de passe. Dans les 2 cas, l'appel de ces méthodes récupère une instance du client et s'en sert pour effectuer les actions dans les menus suivants.

Une fois arrivé au menu principal, le client a le choix entre trois actions principales :

2.1 Passer commande

Le client sera alors redirigé vers le "Module commande" (même s'il n'existe pas à proprement parler dans cette implémentation) et il lui faudra alors rentrer les informations relatives à sa commande :

1. Ville de départ
2. Ville d'arrivée
3. Description de la commande
4. Date de la commande
5. Véhicule
6. Chauffeur

La ville de départ et la ville d'arrivée doivent nécessairement se trouver dans le fichier "Distances.csv" et être des composantes connexes entre elles. Le cas échéant, il sera demandé à l'utilisateur de choisir une nouvelle ville d'arrivée et un nouveau départ.

L'algorithme de Dijkstra permet donc de déterminer la feuille de route, le nombre de kilomètres à parcourir ainsi que le temps nécéssaire pour réaliser la commande.

Une fois que le client a saisi la date de la commande, il doit ensuite choisir un Véhicule et un Chauffeur parmi ceux disponibles le jour de la commande. Si toutes les informations que le client a saisi sont correctes, la commande est donc ajoutée.

2.2 Voir mes commandes & Voir mes informations

Pour que le client puisse utiliser le programme de la meilleure manière possible, il est primordial que celui-ci puisse visualiser certaines informations de manière compréhensive. Ainsi, ce dernier à la possibilité d'afficher toutes ses commandes et les informations relatives à son compte. Il possède également la possibilité de changer certaines de ses informations personnelles (nom, addresse, tél) au cas où ces dernières devraient être mises à jour.

3 Le module Salarié

Le module salarié fonctionne également sous forme de comptes. Cependant, il est important de noter qu'un salarié ne possède pas l'option de s'enregister, il ne peut être ajouté que par un autre salarié.

3.1 Les salariés classiques

Les salariés "classiques" ne disposent que de peu d'options sur le module salarié TransConnect. Ils ont la possibilité de :

— Afficher la liste des clients
— Afficher l'organigramme de la société
— Afficher la flotte de véhicules
— Accéder au module statistiques
— Mettre à jour leurs informations personnelles, comme les clients

Le **module statistique** permet d'afficher différentes statistiques, telles qu'afficher le nombre de livraisons effecuées par chauffeur, la moyenne des prix des commandes, et bien plus.

3.2 Les salariés administrateurs

En plus des fonctionnalités listées ci-dessus, les employées considérés comme administrateurs ont la possibilité de modifier durablement la société. En effet, ils peuvent licensier, embaucher, modifier les salaires, et ajouter et retirer des véhicules de la flotte.

3.3 Les salariés chauffeur

Les chauffeurs ont besoin, pour mener à bien leur mission, de davantage d'informations sur les commandes que les autres employés. Un salarié chauffeur aura par conséquent la possibilité de visualiser, pour chacune de ses commandes, une description détaillée de cette dernière, ainsi que la feuille de route lui permettant d'obtenir le trajet pour cette commande.

3.4 Le PDG

Le PDG de la société possède la permission de promouvoir un salarié en administrateur. Il est le seul employé à posséder cette permission. Le PDG n'est pas un paramètre à renseigner lors de la création d'un salarié. Il s'agit en fait du **noeud racine de l'arbre n-aire des employés**. Ainsi il est également possible qu'un PDG ne soit pas considéré comme administrateur. Il pourra cependant se promouvoir lui-même en administrateur.

4 La liste chaînée

L'implémentation de la classe `ListeChainee<T>` est une implémentation assez avancée qui permet d'utiliser cette classe à peu près comme la classe `List<T>` native au Framework `.NET`. Ainsi, cette classe remplace la classe List par défaut dans la plupart du programme.

La classe implémente l'interface `IEnumerable` et contient une sous-classe `LCEnumerator` qui implémente `Ienumerator`. Cela permet notamment d'utiliser des boucles `foreach`, ce qui rend l'utilisation de la classe bien plus fluide.

Les indexeurs sont également implémentés sur cette classe. Cela permet de pouvoir accéder au i-ème élément de la liste comme : `list[i]`. Tous ces ajustements permettent à la classe `ListeChainee` de remplacer sans problème la classe `List`.

De plus, la classe implémente des méthodes comme `Sort`, `FindAll`, et `ForEach`.

5 Les interfaces

En plus des interfaces implémentées dans la classe `ListeChainee.cs`, mon implémentation de TransConnect implémente deux interfaces : `ISaveable` et `IDataStruct`.

L'interface ISaveable implémente les méthodes statiques :
- `GetFromFile()`
- `SaveToFile()`

L'interface IDataStruct implémente les méthodes :
- `FindAll()`
- `ForEach()`

6 Innovations

Le projet comporte également 4 innovations comme demandé dans le sujet. Ces innovations sont :

1. Le paramètre salarié `isAdmin`, qui donne ou non des droits d'administrateur aux salarié.
2. Ajouter / Retirer un véhicule dans la flotte de véhicules, qui permet aux administrateurs d'ajouter ou de retirer un type de véhicule spécifique
3. Un système de Login / Register à la fois pour les clients et les salariés, avec un email et un mot de passe.
4. Une méthode permettant à une personne (client ou salarié) de modifier ses informations personnelles dans la "base de données" (représentées ici par des fichiers csv).
5. Un menu déroulant pouvant être controlé à l'aide des flèches du clavier pour une meilleure expérience utilisateur.