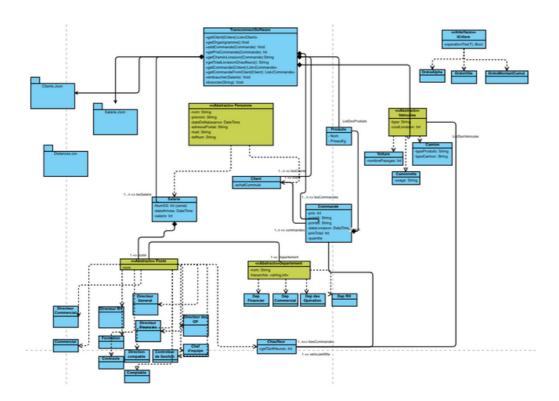
TRANSCONNECT PROJECT

Structure du projets

Contexte:

Transconnect est une société fictive de transport routier. Dans le contexte du projet, il nous était demandé de créer une application permettant de gérer les différentes structures (salariés, commandes, etc...) de cette dernière.

UML:



Modules:

Ci-dessous, la listes des différents modules et leurs fonctionnalités, implémentés dans mon projet:

- Module Client
 - o Affichage liste de clients
 - o Ajout de salarie
 - o Supprimation d'un salarie
 - o Modification d'un client
- Module Salariés
 - o Affichage de l'organigramme
 - o Ajout de salarie
 - o Supprimation d'un salarie
- Module Commandes
 - o Ajout de commandes
 - Supprimation de commande
 - Modification d'une commande
- Module Produits
 - Affichage liste de produits
 - Ajout d'un nouveau produit
 - supprimation d'un produit
- Module Véhicules
 - o Affichage liste de Véhicules
 - o Ajout d'un nouveau véhicule
 - supprimation d'un produit
- Module statistique
 - o Les fonctionnalités de ce module sont contenues dans les 4 premiers.
 - Moyenne achats clients/ Moyenne prix total commandes/Liste de commande par client/Nombre commandes par chauffeurs.

Architecture:

Le projet à été conçu via un "pseudo" MVC (Modèle-Vue-Controleur), qui est un motif d'architecture logicielle destiné aux interfaces graphiques lancé. "Pseudo" car je n'ai pas utilisé de réel interface graphique par contrainte de temps, mais la console.

Le modèle Permet de construire les données afficher, dans mon projets il y a 6 différents de modèles:

- CommandeModel: Dédié à la création de l'objet commande.
- PersonneModel: Dédié à la création d'objet Salarié et d'objet Client.
- PosteModel: Dédié à la création d'objet Poste.
- <u>DépartementModel:</u> Dédié à la création d'objet Département.
- ProductModel: Dédié à la création d'objet Produit.
- <u>VéhiculeMode</u>: Dédié à la création d'objet Véhicule.

Le controleur est le composant qui gère les interactions entre l'utilisateur et l'application. J'ai dans mon projet créé une classe "TransconnectControleur", qui est un objet regroupant toutes les fonctionnalité des modules (ajout, modification, etc...).

Le vue est responsable de l'affichage de l'interface utilisateur (UI) de l'application. Ma vue est gérer directement depuis le main.

Utilitaires

Certains algorithmes, type de fichier ou bien librairies C Sharp ont été nécessaires pour l'implémentation de fonctionnalités, de tests unitaires, ou bien l'initialisation d'objets.

Json:

Il nous était demandé de trouver une solution pour pouvoir charger salariés/clients à l'aide de fichiers. J'ai donc opté pour le sérialisation via les fichiers Json (format de données léger qui est utilisé pour stocker et échanger des données structurées). Pour ce faire, j'ai utilisé une librairie C Sharp appelée *Newtonsoft*, fournissant de nombreuse méthode permettant le sérialisation et le désérialisation d'objets sous json. Ce fut assez simple de gérer les fichiers Json, mais j'ai pu rencontré des problèmes de désérialisation pour les classes héritées. Tout ceci a été géré via ma classe utilitaire *JsonUtil.cs*.

CSV:

Je suis resté sur le type de fichier fourni par notre professeur, c'est-à-dire CVS (fichier au format tableau), pour la gestion des chemins entre villes. Pour ce faire, j'ai utilisé une librairie nommée *CsvHelper*, qui m'a permit de récupérer les données, et de créer des objets *PathCity*. Un *PathCity* contient: une ville de départ, une ville d'arrivée et une distance (la distance entre les deux villes). La récupération de ces données a été gérée par ma classe/objet *PathCityWritter*. Dans cette classe, plusieurs attribues bénéfiques à mon algorithme de Djkstra sont créés, tel que la matrice des distances entre villes ou bien tout simplement la liste des villes.

<u> Dijkstra:</u>

L'algorithme de Dijkstra est un algorithme de recherche de plus court chemin. Il est utilisé pour trouver le chemin le plus court entre deux sommets d'un graphe pondéré. Il était très important de l'implémenter pour trouver les chemins et les distances des commandes à livrées. Pour Dijkstra, j'ai tout simplement implémenté une méthode static dans ma classe nommée *DijkstraFeaturesBI*. Cette méthode retourne la distance et instancie le chemin (string) dans un *PathCityWritter*, donné en paramètre.

N-ary three:

Les arbre n-aires ont été bénéfique pour la création de l'organigramme. Mon fichier utilitaire *Node.cs* contient toutes les classes nécessaires à l'implémentation de l'organigramme. Je suis partie d'une classe abstraite générique de création de neud d'arbre nommée *Node*, puis j'ai implémenté sa classes fille *SalarieNode*. L'arbre complet est créé via la classe *SalarieTree*.

Nunit:

Pour mes testes unitaires, j'ai utilisé la libraire C Sharp NUnit3.

LIENS

- Git: https://github.com/shainaBk/TransconnectProject
- Uml: https://online.visual-paradigm.com/w/glcbxqwf/app/diagrams/#diagram:workspace=glcbxqwf&proj=0&id=1