Compte Rendu 2 du 30 Mars

Pierre MONGRAND Léo GRAVELINE

1. Les stations
2. L’application et la base de données

➞ Objectif : créer une application de commande de nourriture livrée par le cuisinier à Paris  
**Note : Nous avons appelé l’application BTcook avec BT en référence au Bluetooth, l’idée étant d’avoir des livraisons rapides comme la vitesse de transmission du Bluetooth, tout en restant proche et connecté avec le client comme la portée du Bluetooth. Et le cook s’ajoute car c’est une application de livraison de nourriture avant tout !**

1. Les stations

Pour la station, nous avons d’abord pensé créer une base de données contenant les stations, les lignes et leurs informations, mais nous avons finalement gardé le fichier Excel converti en CSV pour lire les stations.  
Pour l’affichage des stations dans la console Windows Form, on est d’abord parti du principe de créer une station par identifiant en modifiant l’identifiant (il contient le numéro de la station et sa ligne). Mais cela posait problème plus tard pour les algorithmes de parcours de chemin, car on avait plusieurs fois la même station (avec le même nom). On n’a donc plus tenu compte de son identifiant mais uniquement de son nom, et on a fait la liste des lignes qui passent par la station.  
L’algorithme fait une liste des stations en lisant les cases de chaque ligne du fichier et, si on a plusieurs fois le même nom de station, on complète la liste des lignes qui la traversent.  
Pour la liste des arcs, on a corrigé l’Excel et mis le nom des stations suivantes et précédentes dans le fichier CSV sous la forme d’une liste (car il y a une station qui a plusieurs suivants). L’algorithme, lors de la création de la carte, lit le nom de la station dans le CSV et sa suivante(s) et précédente(s), puis crée un arc entre les deux et retrouve les stations dans la liste des stations.

Pour l’affichage, nous avons utilisé l’IA et le code de conversion de la latitude et longitude pour avoir des flèches lisibles (méthode ReculerFlèche) et permettre de zoomer. Nous avons ajusté les couleurs propres aux lignes (méthode ObtenirCouleurLigne) et raccourci le nom des stations (méthode FormaterNomStation).

Une fois la carte créée, nous avons créé une méthode qui retrouve deux stations à partir de leur nom dans la liste. Cela sert pour les algorithmes de recherche de chemin.  
L’algorithme de Dijkstra fonctionne en attribuant un poids initial infini à toutes les stations sauf la station de départ, qui reçoit un poids de 0. Ensuite, il sélectionne à chaque itération la station non visitée ayant le poids le plus faible et met à jour les distances de ses voisines si un chemin plus court est trouvé. Ce processus se poursuit jusqu'à ce que toutes les stations aient été traitées ou que la destination soit atteinte. L'algorithme prend également en compte les temps de changement de ligne, ajoutant un coût supplémentaire lorsqu'un voyageur doit changer de ligne de métro. Une fois la station d’arrivée atteinte, le chemin optimal est reconstruit en remontant depuis cette station jusqu'au point de départ.  
L'algorithme de Bellman-Ford initialise également tous les poids à l'infini sauf pour la station de départ. Ensuite, il parcourt tous les arcs du graphe et met à jour les poids en effectuant (nombre de stations - 1) itérations. À chaque passage, il vérifie si une distance plus courte peut être trouvée pour chaque station en relaxant les arcs, c'est-à-dire en mettant à jour les poids si un meilleur chemin est découvert. L’algorithme prend également en compte les changements de ligne en ajoutant un temps supplémentaire si nécessaire. Une optimisation est intégrée pour arrêter l’algorithme plus tôt si aucune mise à jour n’est effectuée durant une itération. Après avoir terminé les itérations, le chemin est reconstruit de manière similaire à Dijkstra en remontant de l’arrivée vers le départ. Bellman-Ford est plus lent que Dijkstra.  
Les deux algorithmes renvoient la liste des stations dans l’ordre grâce à l’attribue récursive Précédent, qui renvoie la station précédente de la station parcourue.

L'algorithme AlgoFloydWarshall calcule les chemins les plus courts entre toutes les paires de stations dans un graphe. Il initialise une matrice distance pour stocker les temps de trajet entre les stations et une matrice predecesseur pour suivre les stations précédentes sur les chemins optimaux. Ensuite, il parcourt toutes les stations intermédiaires possibles et met à jour les distances si passer par une station intermédiaire réduit le temps de trajet. Une fois les distances minimales calculées, il reconstruit le chemin optimal entre deux stations spécifiques en remontant la matrice predecesseur. Si aucun chemin n'existe, il retourne null. Cet algorithme est efficace pour analyser tous les chemins possibles dans un graphe dense.

La méthode AfficherCheminOptimal affiche le trajet optimal entre deux stations en indiquant le temps total, le nombre de stations et les éventuels changements de ligne. Après avoir vérifié que le chemin est valide, elle affiche la station de départ avec sa ligne, puis parcourt les stations intermédiaires en détectant les correspondances. Si un changement de ligne est nécessaire, elle l'affiche avec le temps de correspondance, sinon elle liste simplement la station sur la même ligne. Enfin, elle affiche la station d’arrivée et structure les informations pour une meilleure lisibilité.

Une fois cette partie terminée nous avons lié les codes à la partie base de donnée et interface client.

2) La base de données

La base de données a été modifiée notamment les attribues password qui ont été rajouter aux tables, puis nous les avons complétés en insérant des valeurs par défaut pour les afficher dans l’interface client. Et nous avons retiré le système des menus pour facilité les requêtes.

Enfin nous avons mis les privilèges en créant 3 utilisateurs : l’administrateur qui a tout les droit, le client qui peut voir et modifier supprimer ses messages, ses bon de promo, ses commandes et son compte et il peut voir tous les plats, pareil pour le cuisinier mais on rajoute aussi la possibilité de modifier ses plats et ingrédients.

Pour que chaque client ou cuisinier ne puisse modifier ou voir les informations de quelqu’un d’autre, nous avons mis des View qui permette d’accéder seulement à l’information des tables du client/cuisinier connecté (CURRENT\_USER(); )

Au lancement de l'application, l'utilisateur arrive sur un premier formulaire où il peut choisir de se **connecter** ou de **s'inscrire**, selon qu’il possède déjà un compte ou non.  
S’il choisit de s’inscrire, il devra renseigner son **prénom**, **nom**, **adresse e-mail**, **numéro de téléphone** et **mot de passe**. Une fois l’inscription validée, un nouveau compte utilisateur est créé et les informations sont enregistrées dans la base de données.

L'utilisateur peut ensuite se connecter avec son **e-mail** et son **mot de passe**. Une fois connecté, il choisit son **rôle** dans l'application : **cuisinier** ou **client**. Cette sélection permet d’associer son compte utilisateur au profil correspondant.

Si l'utilisateur choisit le rôle de **cuisinier**, il a alors la possibilité de **créer un plat** en renseignant : le **nom du plat**, les **ingrédients**, le **prix**, la **quantité (pour combien de personnes)**, l’**origine du plat**, le **type de plat**, ainsi que la **station de métro la plus proche** du lieu de préparation.  
Les plats créés sont ensuite enregistrés dans la base de données et visibles pour les clients.

Si l'utilisateur choisit d’être **client**, il a accès à tous les plats déjà créés par les cuisiniers. Il peut alors **sélectionner un plat**, **l’ajouter à son panier**, et **valider sa commande**. Une fois cette commande validée, un **récapitulatif** s’affiche avec le **prix total**, et le client doit indiquer : la **station de métro la plus proche de chez lui**, et éventuellement laisser un **message pour le livreur** (champ facultatif).  
La commande est ensuite enregistrée dans la base de données avec le statut **"en attente"**.

Après cela, une **carte du métro** s’affiche. Le client peut consulter le **chemin le plus court** entre le cuisinier et lui-même à l’aide de **trois algorithmes différents**, permettant d’afficher : le **chemin optimal**, et la **durée estimée de livraison**.

Enfin, le client peut accéder au **formulaire d’avis** afin de **laisser un retour** sur sa commande et **confirmer sa réception**. Une fois cet avis validé : il est enregistré dans la base de données, et la commande passe de l’état **"en attente"** à **"livrée"**.