Cyril BOUHTOU

Emin BOGHOS AZIZIAN

Adli BOUZGUENDA

**Rapport 2ème rendu PSI :**

1. **Consignes de la console**

La console est constituée de 3 menus :

Le 1er menu permet de s’inscrire en tant que nouveau utilisateur. Puis l’utilisateur a le choix de s’inscrire en tant que client, cuisinier ou bien les deux.

Le 2e menu permet à l’utilisateur de se connecter à son compte. Après s’être connecté avec ses identifiants d’utilisateur et son mot de passe utilisateur, il a la possibilité d’accéder aux informations de son compte. Il peut aussi choisir de modifier ses informations et aussi de supprimer son compte cuisinier, cuisinier ou carrément son compte utilisateur. En particulier, s’il est un cuisinier, il peut aussi consulter la liste des clients qu’il a livré depuis son inscription et il peut regarder les commandes qu’il doit livrer (c’est-à-dire les commandes qui n’ont pas encore été livrés).

Le 3e menu est un espace admin. Le code secret est : azerty.

L’admin a un accès global sur la base de données : il peut consulter l’ensemble des utilisateurs, clients et cuisiniers de la plateforme. Ces informations peuvent être affichés par ordre alphabétique des noms. Pour repérer les clients les plus fidèles, l’admin peut aussi afficher la liste des clients triés par achats cumulés. D’autres requêtes sont également possibles comme connaitre l'historique des commandes et plat d'un client triés par nationalité du plat ou encore connaitre la moyenne des prix des commandes.

Enfin, l’admin peut mettre à jour la base de données, s’il y a eu modification sur les fichiers CSV Utilisateurs, Clients et Cuisiniers. INFORMATION : Si lorsque vous lancer la mise à jour des données vous voyez : « Vous voulez insérez l'utilisateur ... dans la BDD », (cela marche également pour les clients et cuisiniers) : cela signifie que l'individu n'existe effectivement pas dans la BDD et qu'il s'apprête à être ajouter. Si cette instruction ne s'affiche pas, c'est que l'individu est déjà présent dans la base : dans ce cas seul son id s'affichera.

Voici une liste d’instructions qui permet de suivre pour manipuler plus simplement la console :

(Pour vous connectez à la base de données, il faut rentrer le numéro de serveur, le numéro de port, le numéro du id et le mot de passe)

1-Ouvrer les deux fichiers SQL. Vérifier que la protection des données et bien activer (SET SQL\_SAFE\_UPDATES = 1). Vérifiez que la data base plateforme ainsi que toutes les tables existent. (Si ce n’est pas le cas, faites-le manuellement). Ne peuplez pas encore les tables.

2- Par défaut, la data base est vide. Il faut donc peupler les tables utilisateurs, clients et cuisiniers. Pour cela, ouvrez la console, aller sur le menu 3 puis faites une mise à jour des données : vous allez ainsi peuplez les tables utilisateurs, clients et cuisiniers. Vérifiez que les instructions « Vous voulez insérez l'utilisateur ... dans la BDD » s’affichent lorsque vous lancez la mise à jour pour confirmer que les données vont bien être ajoutés dans la data base.

3-Ensuite, retourner sur SQL et peuplez manuellement les tables Commande, Plat, livrer, constituer\_de\_ et évaluer.

4-Après cela vous pouvez faire tout ce que vous voulez sur la console. Si vous voulez vous connecter à un compte, je vous donne des comptes d’utilisateurs (un client, un cuisinier et un client/cuisinier) qui possèdent des informations intéressantes sur les commandes, les livraisons, leurs achats …

idU : Cyril23, mdp : fdazdo3 client : CL4

idU : SophieL, mdp : s6t7u8v9 cuisinier : CU7

idU : ClaireL c9d0e1f2 cuisinier CU3 (ClaireL est à la fois un client et un cuisiner, mais elle n’a pas d’informations sur les commandes passées, ou clients livrés…)

**Pour Github :**

[EminBgA/PSI-Rendu1](https://github.com/EminBgA/PSI-Rendu1)

[EminBgA/PSI-Rendu2](https://github.com/EminBgA/PSI-Rendu2)

Nous avons constaté que le premier dépôt GitHub avait accumulé un grand nombre de bugs au fil des manipulations. Par conséquent, nous avons pris la décision de repartir sur un second dépôt GitHub.

Pseudo Emin BOGHOS AZIZIAN : EminBgA

Pseudo Cyril BOUHTOU : Cyril447

Pseudo Adli BOUZGUENDA : adli1805

1. **Schéma entité/association :**

Une image contenant texte, diagramme, capture d’écran, Plan

Le contenu généré par l’IA peut être incorrect.

Description du schéma E/A :

* Table Utilisateur : Décrit les utilisateurs du système avec Id\_Utilisateur et Mot\_de\_Passe.

Un utilisateur est un Client ou un Cuisinier. Il peut être les deux à la fois.

* Table Client : Représente les clients avec des attributs tels que IdClient, NomC, PrénomC, AdresseC, TéléphoneC, EmailC, régimeAlC., LatitudeC et LongitudeC.

Un client passe une ou plusieurs Commande(s).

Un client est un utilisateur.

Un client évalue un Cuisinier.

* Table Cuisinier : Représente les cuisiniers avec des attributs comme spécialités, IDCuisinier, nomP, prénomP, adresseP, téléphoneP, emailP, LatitudeC et LongitudeC.

Un cuisinier prépare un ou plusieurs Plat(s).

Un cuisinier livre une ou plusieurs livraisons.

Un cuisinier est évalué par un Client.

* Table Évaluation : Contient les évaluations des cuisiniers par les clients, avec IDévaluation, Note, et commentaire.

Une évaluation est liée à un Client et un Cuisinier.

* Table Livraison : Décrit les livraisons avec AdresseDep, AdresseArr, Distance, CheminOpt, et statut.

Une livraison est effectuée par un Cuisinier.

* Table Commande : Représente les commandes passées par les clients, avec IdCommande et dateComm.

Une commande est passée par un Client.

Une commande constitue un ou plusieurs Plat(s).

* Table Plat : Décrit les plats avec des attributs tels que Id\_Plat, NomP, DateFabP, typeP, DateP, Prix, Nationalité, Rég\_alI, Ingr, Nb\_portionsP, et photoP.

Un plat est préparé par un Cuisinier.

Un plat constitue une Commande

Le schéma a été amélioré par rapport à la version précédente pour mieux répondre aux exigences fonctionnelles et techniques du projet :

1. **Ajout des coordonnées géographiques (Latitude et Longitude)**

Les attributs LatitudeC et LongitudeC ont été ajoutés à l'entité Client, et LatitudeP, LongitudeP à Cuisinier.

1. **Ajout de la date de commande (dateComm)**

L’attribut dateComm a été introduit dans l’entité Commande pour tracer la date de création de chaque commande.

Cela permet une analyse temporelle des commandes.

1. **Suppression de l'identifiant de livraison (idLivraison)**

L'attribut idLivraison a été supprimé de l'association livrer car il n'était pas nécessaire : chaque livraison est définie par la commande qu’elle concerne, le cuisinier qui la réalise, et les adresses de départ/arrivée.

Cela permet d’alléger la modélisation tout en conservant les informations essentielles.

1. **Analyse des algorithmes de PCC**

**Algorithme de Floyd-Warshall :**

**Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Il a été programmé à l’aide d’une matrice de distances distances[i, j] et d’une matrice de prédécesseurs pred[i, j]. Dans notre version, chaque distance est initialisée à une très grande valeur maxInt, sauf les diagonales qui sont mises à 0. Le traitement des arêtes est effectué grâce à une boucle foreach sur listeLiens, et l’algorithme met à jour les distances via trois boucles imbriquées k, i, j.

Cette implémentation nous permet donc de calculer les plus courts chemins entre toutes les paires de sommets. Elle a une complexité en temps de O(n³).

**Algorithme de BellmanFord :**

**Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.**

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Notre version de Bellman-Ford utilise deux dictionnaires, distances et pred, pour suivre les distances et les prédécesseurs. Chaque sommet est initialisé à int.MaxValue, sauf le sommet de départ. L’algorithme répète la mise à jour des distances sur toutes les arêtes pendant n - 1 itérations.

Elle a une complexité en temps de O(numSommets × numLiens)

**Algorithme de Dijkstra :**

**Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Le contenu généré par l’IA peut être incorrect.**

L’algorithme de Dijkstra que nous avons programmé fonctionne avec une liste de sommets non visités, un dictionnaire des distances, et un dictionnaire des prédécesseurs. Il sélectionne à chaque étape le sommet non visité avec la plus petite distance avec une boucle manuelle. Le chemin est ensuite reconstruit depuis le sommet d’arrivée.

Cette version est très bien adaptée à notre projet, notamment pour trouver rapidement le chemin optimal entre un cuisinier et un client dans un réseau où tous les poids sont positifs. Ce choix est donc privilégié pour les livraisons simples et directes. Elle a une complexité en temps de O(n²).

**Parmi les trois algorithmes implémentés :**

Floyd-Warshall, de complexité O(n³), est le plus complet, car il calcule tous les chemins possibles entre tous les sommets, mais aussi le plus lent, à réserver pour de petites analyses globales.

Même si Bellman-Ford a une complexité théorique de O(n × numLiens) contre O(n²) pour Dijkstra, ce dernier est plus rapide en pratique, notamment sur des graphes comme le nôtre.

En effet, Dijkstra évite les répétitions inutiles et traite chaque sommet de manière plus efficace. Étant donné que notre projet ne prévoit aucun poids négatif, Dijkstra est l’algorithme le plus adapté et performant pour calculer les trajets de livraison.

1. **Niveau Graphe**

Graphe :

Une image contenant motif

Le contenu généré par l’IA peut être incorrect.

Le fichier Excel a été construit avec chacune des stations de métro parisiennes identifiée par un identifiant unique. Pour chaque station, les stations adjacentes ont été listées, ainsi que le degré sortant et le degré entrant, permettant de savoir à quelles stations elle est directement connectée. À partir de ces données, nous avons pu générer automatiquement le graphe du réseau de métro parisien, où chaque nœud correspond à une station et chaque arête représente une liaison directe entre deux stations. Le graphe est orienté, car certaines stations ne vont pas forcément dans les deux sens. Cette construction a servi de base pour appliquer les algorithmes de plus court chemin de Dijkstra, Bellman-Ford et Floyd-Warshall. Elle nous a permis de confronter nos implémentations à un cas concret, en mesurant la performance de chaque algorithme sur un graphe qui simule un réseau de transport.

Programme pour dessiner le graphe :

Une image contenant texte, capture d’écran, logiciel, affichage

Le contenu généré par l’IA peut être incorrect.

On utilise SkiaSharp pour dessiner le graphe à partir des données de nos stations. On commence par placer chaque station sur l’image, puis on trace les connexions entre elles. Chaque station est représentée par un cercle bleu, et son identifiant est écrit à côté. L’image finale est enregistrée en PNG. Cela nous a permis de bien visualiser notre graphe et de vérifier qu’il correspond bien au réseau du métro parisien.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Cette fonction sert à placer correctement chaque station sur l’image en fonction de ses vraies coordonnées GPS. On calcule la position X et Y en tenant compte de l’échelle du canevas, puis on les utilise dans le dessin du graphe.