# RENDU 2 PSI (4 avril)

[RENDU 2 PSI 1](#_Toc1054377275)

[Schéma Entité-association 1](#_Toc66512431)

[Base de données 2](#_Toc1548325091)

[Efficacité des Algorithmes de parcours de graphe 4](#_Toc1321358457)

Lien github: <https://github.com/LoXaAa/PSI_DENNERY_DEGIOANNI_DEPARCEVAUX/tree/b46d6315380535f4f6670095601f82ea7ca09063/PSI_RENDU1>

Pseudo :

* Clementden --> Clément DENNERY
* LoXaAa --> Hugo DEGIOANNI
* Mimicou --> Alexandra DE PARCEVAUX

## Schéma Entité-association

Schéma Entité/ Association

Une image contenant texte, capture d’écran, diagramme, Post-it

Le contenu généré par l’IA peut être incorrect.

Ci-dessus nous avons notre Schéma Entité/Association. Il comprend les entités suivantes :

- Compte : Centralise les informations communes des clients et/ou cuisiniers. Il consigne toutes les informations d’un utilisateur (nom, prénom, adresse postale, rue, numéro de rue, ville, numéro de téléphone, Email, la station de métro la plus proche de chez lui et son statut sur l’application (actif/inactif),mot de passe) à chaque compte est lié un numéro de compte unique.

- Client : Passe 0 à n commande(s) et peut donner 0 à n des avis. Il est constitué d’un numéro de client unique et d’un nom d’entreprise s’il y en a un. Il est lié par un héritage total à l’entité Compte, ce qui signifie que chaque client a obligatoirement un unique compte qui lui est lié.

- Cuisinier : Prépare 1 à n plat(s) et reçoit 0 à n avis. Il est constitué d’un unique numéro de compte cuisinier qui lui est propre et d’une zone de livraison. Il est lui aussi lié en héritage totale à Compte.

- Commande : Contient 1 à n plat(s) commandé(s) par un et un seul client. La commande est liée au client ce qui signifie que si le client est supprimé alors la commande aussi. La commande a un numéro unique de commande et des données décrivant la commande c’est-à-dire en donnant le statut de la commande, si elle a été validée ou pas, le statut de la transaction si elle a été payée ou non, la date où la commande a été faite, son prix, la date où le paiement a été fait et par quel moyen (PayPal, Lydia, liquide, ...)

- Plat : Préparé par un et un seul cuisinier et défini par une et une seule recette. Il est constitué lui aussi d’un numéro unique. Les données le décrivant y sont aussi consignées avec le type de plat (dessert, entrée, ...), sa date de péremption, sa date de fabrication qui sont assez explicites, une photo du plat, le type de régime (végan, Allal, sans gluten,), une description plus ou moins détaillé du plat a nationalité du plat s'il en a une (Italien, japonais, ...), le prix du plat, ses ingrédients qui le composent en majorité et enfin le nombre de portion qu’il contient.

- Recette : Liste les étapes et les 1 à n ingrédient(s) nécessaire(s). Là encore il a un numéro unique de recette, puis il a un nom, des instructions pour le faire, une estimation du temps de cuisson et de préparation et enfin la difficulté estimée pour le faire.

- Ingrédient : Utilisé dans 0 à n recettes. Il est composé d’un numéro unique ingrédients et du volume ainsi que de l’ingrédient le composant.

- Avis : Donné par un client à un cuisinier. Il est comme les autres composés d’un numéro unique, d’une note allant de 1 à 5, de la date à laquelle l’avis a été fait et de l’avis en question.

## Base de données

Les comptes administrateur et leur code sont tous les deux “root”.

-- Créer la base de données et les tables DROP DATABASE IF EXISTS livparis; CREATE DATABASE livparis; USE livparis;

-- Création des tables

CREATE TABLE IF NOT EXISTS Compte ( Id\_Compte INT AUTO\_INCREMENT, Prenom VARCHAR(50), Nom VARCHAR(50), Rue VARCHAR(50), Numero INT, Code\_postal INT, Ville VARCHAR(50), No\_tel VARCHAR(20), Email VARCHAR(50), Station\_de\_Métro\_la\_plus\_Proche VARCHAR(50), Statut VARCHAR(50),

Mot\_Passe VARCHAR(50), PRIMARY KEY(Id\_Compte) );

CREATE TABLE IF NOT EXISTS Client( Id\_Client INT AUTO\_INCREMENT, Nom\_Entreprise VARCHAR(50), Id\_Compte INT NOT NULL, PRIMARY KEY(Id\_Client), UNIQUE(Id\_Compte), FOREIGN KEY(Id\_Compte) REFERENCES Compte(Id\_Compte) ON DELETE CASCADE );

CREATE TABLE IF NOT EXISTS Recette( Id\_Recette INT AUTO\_INCREMENT, Nom\_Recette VARCHAR(50), Instructions VARCHAR(50), Temps\_Préparation INT, Temps\_Cuisson INT, Difficulte VARCHAR(50), PRIMARY KEY(Id\_Recette) );

CREATE TABLE IF NOT EXISTS Ingredient( Id\_Ingredient INT AUTO\_INCREMENT, Volume INT, Ingredient VARCHAR(50), PRIMARY KEY(Id\_Ingredient) );

CREATE TABLE IF NOT EXISTS Cuisinier( Id\_Cuisinier INT AUTO\_INCREMENT, Zone\_Livraison VARCHAR(50), Id\_Compte INT NOT NULL, PRIMARY KEY(Id\_Cuisinier), UNIQUE(Id\_Compte), FOREIGN KEY(Id\_Compte) REFERENCES Compte(Id\_Compte) ON DELETE CASCADE );

CREATE TABLE IF NOT EXISTS Plat( Id\_Plat INT AUTO\_INCREMENT, Type\_Plat VARCHAR(50), Date\_Peremption DATE, Date\_Fabrication DATE, Type\_Regime VARCHAR(50), Photo VARCHAR(50), Description VARCHAR(50), Nationalité VARCHAR(50), Prix DECIMAL(15,2), Nombre\_Portion INT, Ingrédients\_Principaux VARCHAR(50), Id\_Recette INT, Id\_Cuisinier INT NOT NULL, PRIMARY KEY(Id\_Plat), FOREIGN KEY(Id\_Recette) REFERENCES Recette(Id\_Recette) ON DELETE SET NULL, FOREIGN KEY(Id\_Cuisinier) REFERENCES Cuisinier(Id\_Cuisinier) ON DELETE CASCADE );

CREATE TABLE IF NOT EXISTS Avis( Id\_Avis INT AUTO\_INCREMENT, Note INT, Date\_Avis DATE, Commentaire VARCHAR(50), Id\_Client INT NOT NULL, Id\_Cuisinier INT NOT NULL, PRIMARY KEY(Id\_Avis), FOREIGN KEY(Id\_Client) REFERENCES Client(Id\_Client) ON DELETE CASCADE, FOREIGN KEY(Id\_Cuisinier) REFERENCES Cuisinier(Id\_Cuisinier) ON DELETE CASCADE );

CREATE TABLE IF NOT EXISTS Commande( Id\_Commande INT AUTO\_INCREMENT, Date\_Commande DATE, Statut\_Commande VARCHAR(50), Prix\_Total DECIMAL(15,2), Statut\_Transaction VARCHAR(50), Date\_Paiement DATE, Mode\_Paiement VARCHAR(50), Id\_Client INT NOT NULL, Id\_Cuisinier INT NOT NULL, PRIMARY KEY(Id\_Commande), FOREIGN KEY(Id\_Client) REFERENCES Client(Id\_Client) ON DELETE CASCADE, FOREIGN KEY(Id\_Cuisinier) REFERENCES Cuisinier(Id\_Cuisinier) ON DELETE CASCADE );

CREATE TABLE IF NOT EXISTS contient( Id\_Recette INT, Id\_Ingredient INT, PRIMARY KEY(Id\_Recette, Id\_Ingredient), FOREIGN KEY(Id\_Recette) REFERENCES Recette(Id\_Recette) ON DELETE CASCADE, FOREIGN KEY(Id\_Ingredient) REFERENCES Ingredient(Id\_Ingredient) ON DELETE CASCADE );

CREATE TABLE IF NOT EXISTS comprend( Id\_Commande INT, Id\_Plat INT, PRIMARY KEY(Id\_Commande, Id\_Plat), FOREIGN KEY(Id\_Commande) REFERENCES Commande(Id\_Commande) ON DELETE CASCADE, FOREIGN KEY(Id\_Plat) REFERENCES Plat(Id\_Plat) ON DELETE CASCADE );

--Peuplement de la BDD

USE livparis; -- rajouter après la date de fin du le rendu -- Peuplons la table Compte INSERT INTO livparis.Compte (Id\_Compte, Prenom, Nom, Rue, Numero, Code\_postal, Ville, No\_tel, Email, Station\_de\_Métro\_la\_plus\_Proche, Statut,Mot\_Passe)

VALUES (1, NULL, NULL, 'Rue de la Paix', 10, 75001, 'Paris', 0123456789, '[jean.dupont@example.com](mailto:jean.dupont@example.com)', 'Opéra', 'Actif',1);-- cas où il manque des informations INSERT INTO livparis.Compte (Id\_Compte, Prenom, Nom, Rue, Numero, Code\_postal, Ville, No\_tel, Email, Station\_de\_Métro\_la\_plus\_Proche, Statut,Mot\_Passe)

VALUES (2, 'Sophie', 'Martin', 'Avenue des Champs-Élysées', 50, 75008, 'Paris', 0123456790, '[sophie.martin@example.com](mailto:sophie.martin@example.com)', 'Franklin D. Roosevelt', 'Inactif',2);

INSERT INTO livparis.Compte (Id\_Compte, Prenom, Nom, Rue, Numero, Code\_postal, Ville, No\_tel, Email, Station\_de\_Métro\_la\_plus\_Proche, Statut,Mot\_Passe)

VALUES (3, 'Claude', 'Bernard', 'Rue de la République', 15, 69002, 'Lyon', 0123456791, NULL, 'Bellecour', 'Actif',3);-- cas où est à la fois cuisinier et client

INSERT INTO livparis.Compte (Id\_Compte, Prenom, Nom, Rue, Numero, Code\_postal, Ville, No\_tel, Email, Station\_de\_Métro\_la\_plus\_Proche, Statut,Mot\_Passe)

VALUES (4, 'Alice', 'Durand', 'Cours Mirabeau', 5, 13100, 'Aix-en-Provence', 0123456792, '[alice.durand@example.com](mailto:alice.durand@example.com)', 'Rotonde', 'Actif',4);

INSERT INTO livparis.Compte (Id\_Compte, Prenom, Nom, Rue, Numero, Code\_postal, Ville, No\_tel, Email, Station\_de\_Métro\_la\_plus\_Proche, Statut,Mot\_Passe) VALUES (5, 'Paul', 'Petit', 'Rue de Strasbourg', 8, NULL, 'Strasbourg', 0123456793, '[paul.petit@example.com](mailto:paul.petit@example.com)', 'Homme de Fer', 'Inactif',5);

-- Clients

INSERT INTO livparis.Client (Id\_Client, Nom\_Entreprise, Id\_Compte)

VALUES (1, 'Entreprise A', 1);

INSERT INTO livparis.Client (Id\_Client, Nom\_Entreprise, Id\_Compte)

VALUES (2, 'Entreprise B', 2);

INSERT INTO livparis.Client (Id\_Client, Nom\_Entreprise, Id\_Compte)

VALUES (3, 'Entreprise C', 3);

-- Cuisiniers

INSERT INTO livparis.Cuisinier (Id\_Cuisinier, Zone\_Livraison, Id\_Compte)

VALUES (3, 'Ouest', 3);

INSERT INTO livparis.Cuisinier (Id\_Cuisinier, Zone\_Livraison, Id\_Compte) VALUES (4, 'Sud', 4);

INSERT INTO livparis.Cuisinier (Id\_Cuisinier, Zone\_Livraison, Id\_Compte) VALUES (5, 'Est', 5);

-- Peuplons la table Commande

INSERT INTO livparis.Commande (Id\_Commande, Date\_Commande, Statut\_Commande, Prix\_Total, Statut\_Transaction, Date\_Paiement, Mode\_Paiement, Id\_Client, Id\_Cuisinier)

VALUES (1, '2025-03-01', 'En cours', 50.75, 'Payé', '2025-03-02', 'Carte de crédit', 1, NULL);

INSERT INTO livparis.Commande (Id\_Commande, Date\_Commande, Statut\_Commande, Prix\_Total, Statut\_Transaction, Date\_Paiement, Mode\_Paiement, Id\_Client, Id\_Cuisinier)

VALUES (2, NULL, 'Livrée', 30.50, 'Payé', '2025-03-03', 'PayPal', 2,4);

INSERT INTO livparis.Commande (Id\_Commande, Date\_Commande, Statut\_Commande, Prix\_Total, Statut\_Transaction, Date\_Paiement, Mode\_Paiement, Id\_Client, Id\_Cuisinier)

VALUES (3, '2025-03-03', 'En préparation', 45.00, 'Non payé', NULL, 'Espèces', 3,5);

INSERT INTO livparis.Commande (Id\_Commande, Date\_Commande, Statut\_Commande, Prix\_Total, Statut\_Transaction, Date\_Paiement, Mode\_Paiement, Id\_Client, Id\_Cuisinier)

VALUES (4, '2025-03-04', 'Annulée', 25.00, 'Non payé', NULL, 'Carte de crédit', 2,4);

INSERT INTO Recette (Nom\_Recette, Instructions, Temps\_Préparation, Temps\_Cuisson, Difficulte)

VALUES ('Quiche Lorraine', 'Préparer la pâte puis la garniture', 15, 30, 'Facile'), ('Tarte Tatin', 'Cuire les pommes puis monter la tarte', 20, 40, 'Moyen');

INSERT INTO Ingredient (Volume, Ingredient) VALUES (200, 'Farine'), (100, 'Beurre');

INSERT INTO Plat (Type\_Plat, Date\_Peremption, Date\_Fabrication, Type\_Regime, Photo, Description, Nationalité, Prix, Nombre\_Portion, Ingrédients\_Principaux, Id\_Recette, Id\_Cuisinier)

VALUES ('Entrée', '2025-04-10', '2025-04-01', 'Végétarien', 'photo1.jpg', 'Délicieuse entrée végétarienne', 'Française', 12.50, 2, 'Tomate, Basilic', 1, 4), ('Dessert', '2025-04-15', '2025-04-05', 'Sans gluten', 'photo2.jpg', 'Dessert gourmand sans gluten', 'Française', 8.75, 1, 'Pommes, Caramel', 2, 5);

INSERT INTO Avis (Note, Date\_Avis, Commentaire, Id\_Client, Id\_Cuisinier)

VALUES (5, '2025-04-03', 'Excellent service', 1, 4), (4, '2025-04-02', 'Très bon plat', 2, 5);

## Efficacité des Algorithmes de parcours de graphe

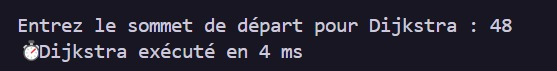
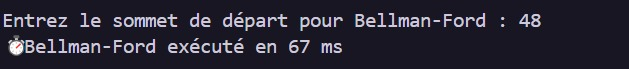
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithme | Complexité | Interprétation | Temps d’exécution | Conclusion |
| Dijkstra | O((V\*E)\*log(V)) | **Dijkstra avec tas** est le plus rapide sur les **grands graphes clairsemés**. | 4 ms | **Dijkstra** est **très rapides** pour un graphe de cette taille. |
| Bellman-Ford | O(V\*E)  On relâche **toutes les arêtes** sur **V - 1 itérations**.  Vérification finale pour détecter un cycle négatif : O(E) | **Bellman-Ford** est plus lent mais gère les **poids négatifs**. | 67 ms | **Bellman-Ford** est **très rapides** pour un graphe de cette taille. |
| Floyd-Warshall | O(V3) | **Floyd-Warshall** est inefficace pour les **grands graphes**, mais pratique pour les **petits graphes denses** | 2167ms | **Floyd-Warshall** est inefficace pour les **grands graphes**, mais pratique pour les **petits graphes denses** |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

En conclusion : Pour des graphes avec poids positifs il vaut mieux utiliser Dijkstra. Pour des graphes avec poids négatifs (ce qui n’est pas le cas ici), il faut utiliser Bellman-Ford. Et pour des graphes complets Floyd-Warshall est très efficace

Voici l’algorithme utilisé pour calculer le temps d’exécution :



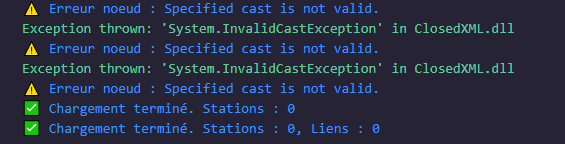


## Promptes issues de l’IA générative

* Comme pour le dernier rendu, la modélisation des graphes a été faite notamment via l’IA

**Difficultés liées aux conversions de types (cast) lors de l'importation Excel**

Dans un premier temps, l’intégration des données issues du fichier Excel s’est révélée plus complexe que prévu. Bien que l’IA générative ait été mobilisée pour générer les premières fonctions de lecture, un obstacle bloquant est rapidement survenu : une exception InvalidCastException empêchait la conversion correcte des cellules Excel vers les types attendus



Ce problème provenait d’une tentative de conversion directe de certaines cellules vers des types int ou double, sans prendre en compte les cellules vides ou les formats non conformes. Malgré plusieurs prompts et ajustements, le problème persistait.

Solution : pour isoler cette problématique, une nouvelle classe dédiée a été introduite. Celle-ci centralise la lecture des feuilles Excel (LecteurExcel) et y applique des vérifications plus robustes (tests de cellule vide, parsing sécurisé, etc.). Cela a permis de stabiliser l’importation des nœuds et des arcs.

Dans la continuité du travail précédent, nous avons de nouveau fait appel à l’IA générative pour automatiser certaines tâches de modélisation, notamment la lecture des fichiers Excel, l’interprétation des nœuds et des liens, et la construction du graphe orienté.

Le mapping entre colonnes Excel et indices utilisés dans le code (via Cell(x)) ne correspondait pas à la réalité du fichier.

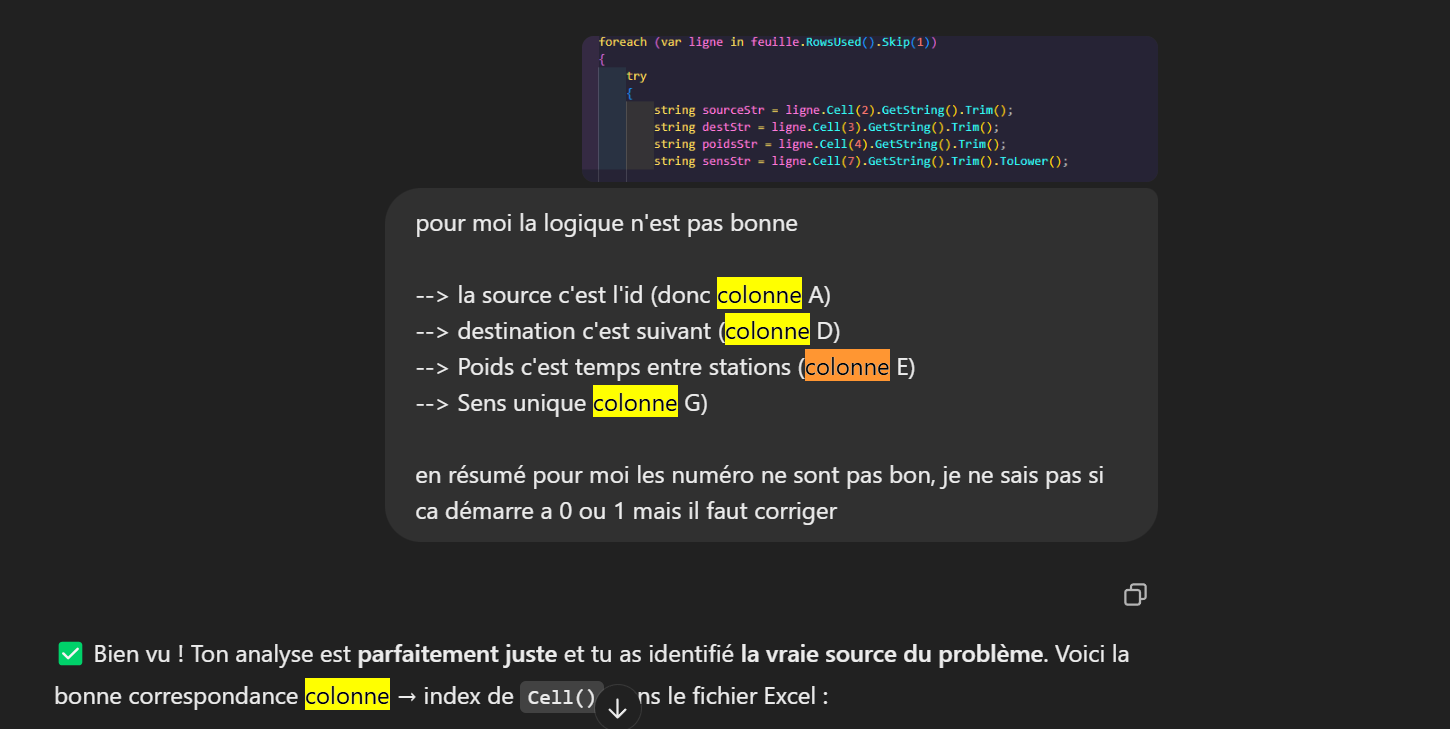
Solution : relecture attentive des colonnes et réaffectation manuelle des index corrects dans la méthode LireArcs() :

Source → colonne A

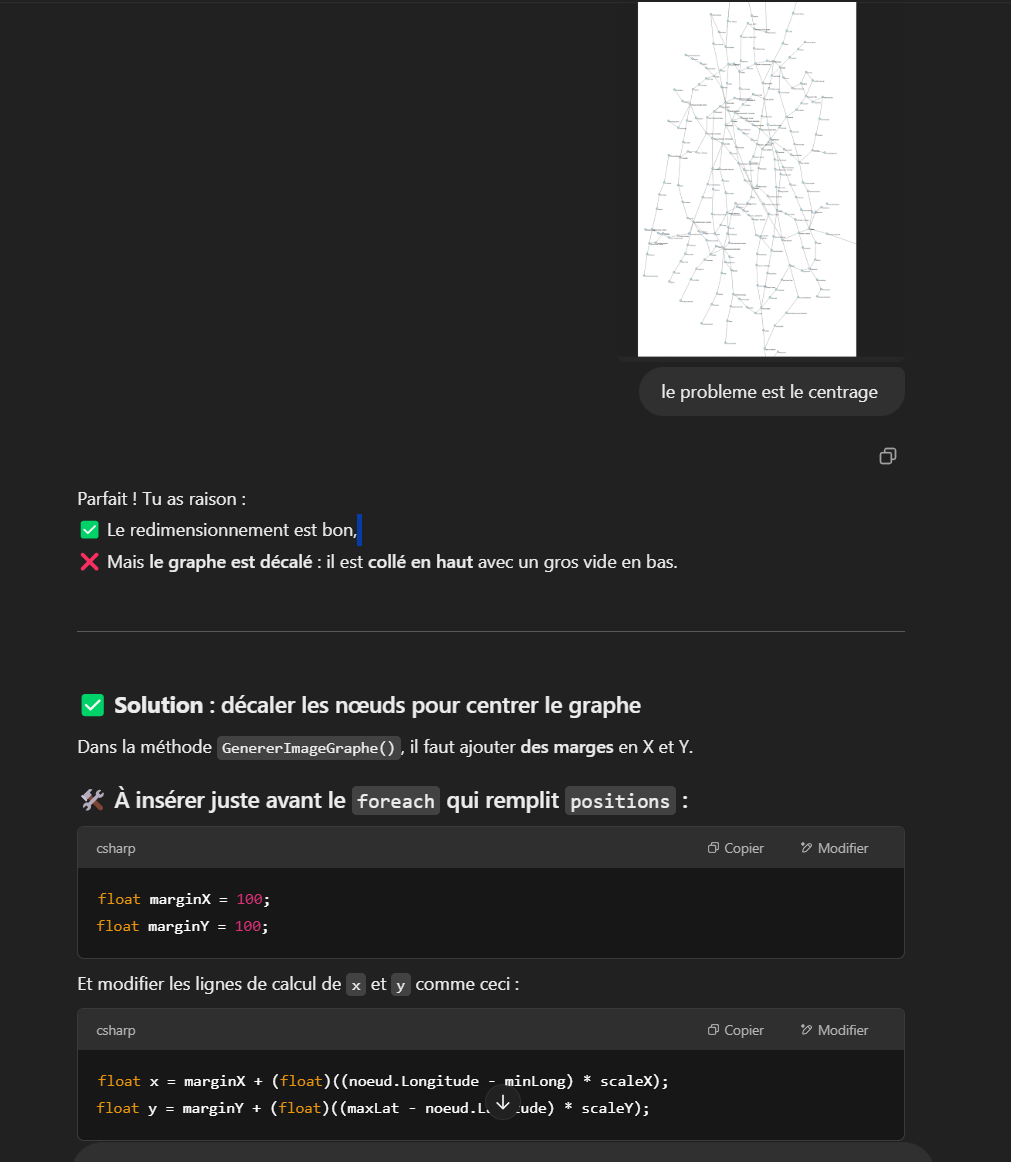
Destination → colonne D

Poids → colonne E

Sens unique → colonne G



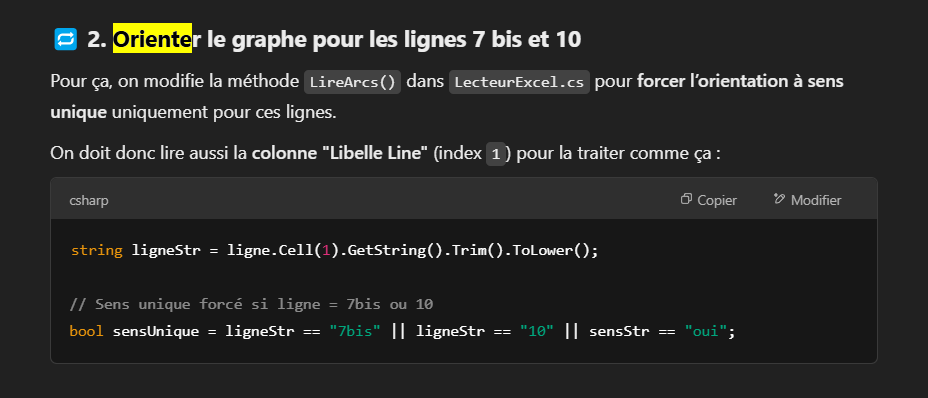
Problème rencontré : Le problème est que le redimensionnement de l’image était bon, mais le graphe apparaissait décalé vers le haut, laissant un grand vide en bas.



Solution proposée : ajout de marges X et Y dans la méthode GenererImageGraphe() pour centrer les nœuds dans le bitmap généré.

Orientation du graphe :

Après avoir validé la structure générale du graphe, nous avons constaté que certaines lignes spéciales du métro parisien, notamment les lignes 7 bis et 10, posaient un problème. Ces lignes sont partiellement ou totalement à sens unique, ce qui faussait la représentation orientée du graphe.



Solution : dans la méthode LireArcs() de la classe LecteurExcel, nous avons ajouté une condition spécifique pour détecter les lignes "7bis" et "10", et ainsi forcer le graphe à créer les arcs dans un seul sens. Après cela le graphe correspondait à nos attentes.