GLO-4035

Bases de données avancées

Automne 2023

TRAVAIL LONGITUDINAL

Parcours de vélo épicurien

Travail présenté à

David Beauchemin

Boubacar Hama Bague, boubacar.hama-bague.1@ulaval.ca,

Shelly CAPRICE, shelly.caprice.1@ulaval.ca,

Baptiste LEFEBVRE, baptiste-benoit.lefebvre.1@ulaval.ca

Abdelaziz ABDELKEFI, abdelaziz.abdelkefi.1@ulaval.ca

Table des matières

Table des figures

1. Introduction

1.1. Explication de la problématique

Nous avons le plaisir de vous présenter notre application de planification de promenade épicurienne à vélo à Québec. Elle a pour but de proposer des parcours de pistes cyclables sur lesquels sont situés les meilleurs restaurants. Notre solution comprend donc les informations de centaines de restaurants de la ville ainsi que les pistes cyclables sécurisées officielles fournies par le gouvernement du Québec. La solution finale permettra ainsi de découvrir différents parcours en fonction des critères choisis, comme le type de restaurants ou la distance à parcourir.

Ce rapport technique s'adresse aux investisseurs potentiels qui souhaitent nous aider à faire de cette idée une réalité. Il vous permettra de découvrir les détails techniques de la preuve de concept que nous avons réalisé et notre plan d'expansion.

1.2. Présentation du rapport

A la suite de cette introduction, ce rapport présente la stratégie d'acquisition des données en expliquant d'où viennent les données que nous utilisons, comment nous les récupérons ainsi que des exemples. Nous détaillons ensuite les technologies que nous avons choisies pour le développement de cette application, que ce soit le langage de programmation ou les différentes bases de données. Par la suite, le rapport expose les détails du processus d'extraction, de transformation et de conversion des données. Cette partie inclue des explications pour chaque partie du processus, ainsi que des diagrammes UML et des exemples de nos données. Ultérieurement, le rapport met en lumière les détails de la pipeline de données en expliquant l'algorithme de production des parcours et l'algorithme de calcul du meilleur parcours pour l'usager. Enfin, nous présentons notre plan d'expansion en clarifiant nos stratégie de réplication, de partitionnement et de sécurité de nos bases de données. Vous trouverez également en annexe une estimation des coûts d'acquisition de nouvelles données et du développement des fonctionnalités.

2. Stratégie d'acquisition des données

2.1. Source et méthode d'extraction

Nous avons décidé de choisir la *ville de Québec au Canada* comme lieu de réalisation du projet. Pour développer notre application, nous avons besoin de deux sources de données : les restaurants présents dans Québec ainsi que les pistes cyclables.

Ci-dessous les méthodes d'acquisition ainsi qu'une analyse d'un extrait des données :

A) Restaurants : API Yelp

Yelp est l'une des plus grandes plateformes de revues de commerces dans le monde et offre une riche base de données sur les restaurants, y compris leurs évaluations, emplacements, types de cuisine, et plus encore.

Pour recueillir les données des restaurants, nous avons décidé d'utiliser l'API de cette plateforme, ce qui nous garantit d'obtenir des informations fiables.

Pour extraire les données, nous allons utiliser les requêtes GET de *l'API Business Search Yelp* en spécifiant la localisation souhaitée et le type d'établissement *'restaurant'*.

B) Pistes cyclables : API CKAN de Données Québec

Données Québec est une initiative du gouvernement du Québec qui donne accès ouvertement à des informations sur certaines villes, dont Québec City. Ce portail fourni un outil nommé API CKAN qui permet de récupérer les données des pistes cyclables comme les coordonnées, les noms, le sens de circulation, et plus encore. Étant donné que ce site est maintenu par le gouvernement, sa fiabilité est garantie.

Pour extraire les données, nous utiliserons les requêtes GET qui permettront d'extraire les données des pistes cyclables basées sur la localité souhaitée.

2.2. Exemple de données

A) Exemple de données restaurants

Dans l'Annexe A : Exemple de données de l'API Yelp, nous pouvons voir l'extrait d'un restaurant possédant plusieurs attributs vitaux comme son nom, son adresse, ses coordonnées géographiques.

Sur ces données, nous pouvons constater qu'il existe des champs vides. Cela est principalement dû à un manque d'informations renseignées comme pour un numéro de téléphone ou le type de transaction. Aussi, d'autres attributs peuvent être vides par manque de données pour l'attribut *rating*.

B) Exemple de données pistes cyclables

L'Annexe B : Exemples de données de l'API CKAN présente un exemple de données de l'API CKAN qui nous permet de récupérer les pistes cyclables de la ville de Québec. L'exemple est celui d'une section du Boulevard des Alliés. Chaque section est représentée notamment par le nom et le type de la rue, la longueur de la section ainsi que 2 ou 3 coordonnées géographiques.

Il est pertinent de noter que les coordonnées des pistes cyclables indiquent d'abord le point de départ et ensuite les coordonnées latitude/longitude de la fin de la piste. Cependant, ces points ne sont pas forcément continus, ce qui signifie qu'une piste ne se prolonge pas directement par une autre route au même endroit.

3. Technologies utilisées

3.1. Langage de programmation

A) Langage de programmation : Python

Pour ce projet, nous avons choisi le langage de programmation Python en raison de sa fiabilité, maintenabilité et extensibilité. Python assure la fiabilité grâce à une gestion robuste des exceptions et de la mémoire. En ce qui concerne la maintenabilité de ce langage, il est facilité par sa lisibilité, sa documentation abondante et ses tests unitaires. Quant à son extensibilité, elle est garantie par des frameworks, des modules tiers et une communauté active. En somme, Python offre un équilibre idéal entre ces trois aspects cruciaux, assurant la stabilité, la gestion efficace du code et la flexibilité pour évoluer avec les besoins du projet de base de données.

Enfin, le langage de programmation Python semble évident pour ce projet car il dispose de bibliothèques robustes qui simplifient grandement l'intégration avec nos API *(API Yelp et l'API CKAN de Données Québec)*.

B) Framework : Flask

Concernant le framework, nous avons adopté Flask non seulement pour sa simplicité, mais surtout parce qu'il se prête parfaitement à la gestion des requêtes spécifiques demandées pour ce projet. Flask permet de créer des points d’accès à notre application web de façon efficace et transparente. Pour rappel, cette étape est essentielle pour fournir les données des pistes cyclables et des restaurants de manière optimale à nos utilisateurs.

3.2. Bases de données

A) Base de données pour les pistes cyclables : Neo4j

Neo4j est une base de données orientée graphe utilisée pour stocker et gérer des données interconnectées. L'utilisation de Neo4j pour gérer les données des pistes cyclables de la ville de Québec est justifiée par sa capacité à modéliser efficacement les relations complexes entre les éléments du réseau cyclable. Neo4j, en tant que base de données orientée graphe, assure la fiabilité des données en permettant l'établissement de contraintes, tout en facilitant la maintenabilité grâce à son modèle souple et à son langage de requête. Ainsi, l'utilisation de Neo4j garantit à notre projet une gestion précise et évolutive des données des pistes cyclables.

B) Base de données pour les restaurants : MongoDB

MongoDB est une base de données NoSQL qui trouve une application pertinente dans la gestion des données des restaurants de la ville de Québec. Son utilisation est justifiée en se fondant sur le principe de l'extensibilité, car elle permet de stocker efficacement des informations relatives aux restaurants et des données semi-structurées telles que les avis des clients, les informations sur les menus, les heures d'ouverture et la localisation des restaurants.

4. Les détails du processus d'extraction, de transformation et de conversion

4.1. Processus d'acquisition initiale des données

A) Restaurants

Pour récupérer les données des restaurants, nous utilisons les requêtes GET de *l'API Business Search Yelp* en spécifiant la localisation souhaitée et le type d'établissement *'restaurant'*. Nous récupérons alors les informations au format *.json*. C'est à partir de ce fichier que nous pourrons ensuite transformer les données pour la base de données MongoDB.

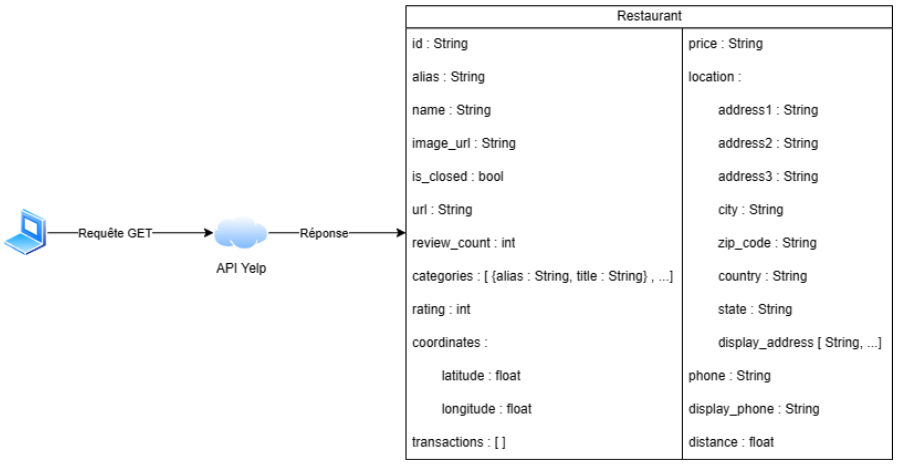


Figure 1 : Diagramme d'acquisition des restaurants

B) Pistes cyclables

Pour récupérer les données des pistes cyclables, nous avons utilisé l'API du CKAN pour télécharger un fichier *.geojson* avec toutes les données. C'est à partir de ce fichier que nous pourrons ensuite transformer les données pour la base de données Neo4j.

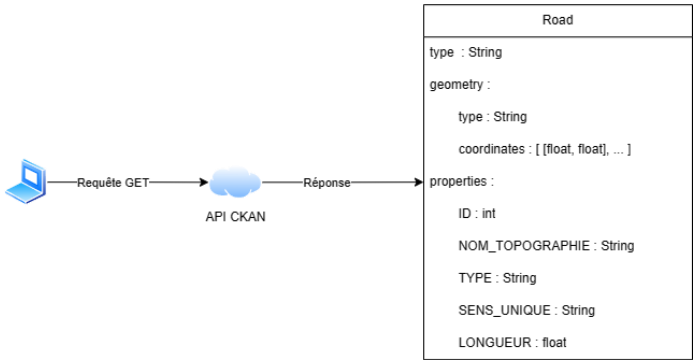


Figure 2 : Diagramme d'acquisition des pistes cyclables

4.2. Processus d'acquisition incrémental des données

Le processus d'acquisition incrémentale des données en informatique désigne une approche progressive et itérative visant à collecter des informations de manière incrémentale au fil du temps, permettant ainsi une mise à jour régulière et une amélioration continue des données.

Dans le cadre de ce projet, l'ajout incrémental de données nous permet de mettre à jour nos bases de données *(s'il y a de nouveaux restaurants ou pistes cyclables)* sans avoir à recommencer tout le processus d'acquisition. En effet, lorsque nous réalisons cette mise à jour, nous pouvons alors conserver nos données et seulement ajouter les éléments qui n'existent pas encore dans nos bases de données.

Pour développer ceci, nous avons créé une fonction python *compare_and_update*. Nous utilisons cette fonction de façon occasionnelle *(une fois par mois)* pour mettre à jour les fichiers d'origines, pour ensuite appliquer les modifications sur les bases de données en réinitialisant ces bases de données si nécessaire. Nous utilisons cette fonction de façon périodique car que ce soit pour les données fourni par YELP *(les restaurants)* ou les données fournies par le gouvernement du Québec *(pistes cyclables)*, ces deux sources d'informations sont rarement mises à jour.

Cette fonction pour l'ajout incrémental fonctionne de la façon suivante : Tout d'abord, les fichiers JSON sont chargés dans des structures de données en mémoire à l'aide de la bibliothèque JSON de Python. Ces structures sont ensuite comparées de manière itérative, en parcourant récursivement les dictionnaires pour détecter les différences au niveau des clés et des valeurs.

4.3. Processus de transformation des données

Une fois les données récupérées, il nous faut les transformer afin de les stocker efficacement dans les différentes bases de données. Le traitement est différent pour les restaurants et les pistes cyclables. Mais dans les deux cas, le but est de filtrer les données en choisissant les informations à stocker ou à retirer. Enfin, nous transformerons les données dans le format de la base de données choisie.

A) Restaurants

Le format *.json* étant accepté par MongoDB, il nous suffit d'appliquer un filtre sur notre fichier pour appliquer le processus de transformation des données.

Ci-dessous un diagramme UML des étapes de la transformation *(Figure 4)* et un exemple de cette transformation *(Figure 3)*.

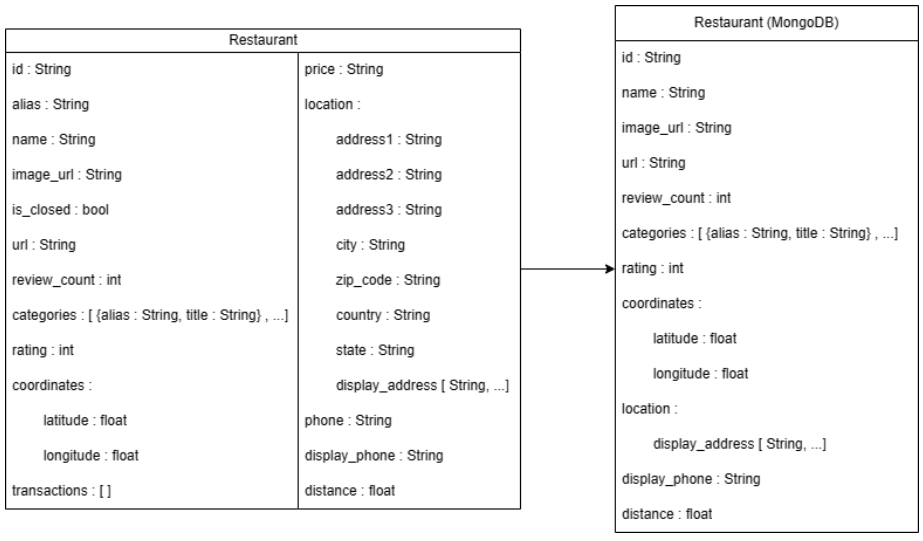


Figure 4 : Diagramme de transformation des restaurants

Figure 3 : Exemple de transformation des restaurants

Ainsi, comme on peut le voir sur la *Figure 3*, nous avons supprimé plusieurs objets/tableaux :

- *Alias :* cette information est similaire à l'information *nom*
- *Is_closed :* Lors de notre filtre, nous avons pris soin de sélectionner uniquement les restaurants ouverts.
- *transactions :* Ne contient pas d'informations pertinentes pour notre projet
- La plupart des paramètres de *location* : Afin de ne conserver que *display_address*
- *Phone* : nous avons déjà *display_phone* qui est plus lisible

B) Pistes cyclables

En ce qui concerne les pistes cyclables, la transformation est un peu plus complexe car les données seront stockées sous forme de graphe sur Neo4j. Pour cela, nous avons choisit de représenter chaque route par un nœud « Road ». Chacun de ces nœuds est alors lié à des nœuds « Point » qui représentent les coordonnées de départ et d'arrivée de la route.

Toutes les informations récupérées de l'API ne sont pas utilisées. Nous avons choisi de ne conserver que : les coordonnées *(latitude, longitude)*, le nom de la rue, le sens de circulation, ainsi que la longueur de la rue. En ce qui concerne les coordonnées, nous avons décidé de conserver uniquement le point de départ et d'arrivée de chaque section de route. En effet, il n'est pas utile de conserver les informations de chaque légère déviation d'une section de route.

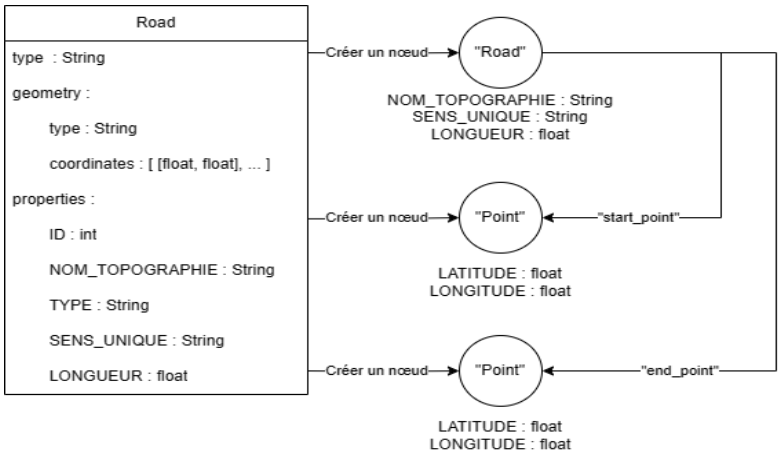


Figure 6 : Diagramme de transformation des pistes cyclables

Figure 5 : Exemple de transformation des pistes cyclables

4.4. Schéma de la pipeline d'ETL

Pour les deux bases de données, nous avons commencé par les initialiser et les peupler avec leurs données transformées respectives.

Suite à cela, que ce soit pour MongoDB ou Neo4j, nous avons sauvegardé les volumes, ce qui nous permet de charger les bases de données bien plus rapidement à partir de leurs volumes respectifs. Ainsi, cela évite à l'utilisateur de devoir réinitialiser chaque base de données sur sa machine.

L'Annexe C présente le diagramme de la pipeline d'ETL de notre application.

5. Les détails de la pipeline de données

5.1. L'algorithme de production des parcours

L'obtention du parcours se fait par l'exécution d'une requête avec les paramètres suivants :

- le point de départ du parcours
- la longueur totale du parcours
- le type de restaurant
- le nombre d'arrêts souhaités

La distance du parcours doit être comprise dans plus ou moins 10% de la distance demandée par l'utilisateur.

Le parcours commence à un premier restaurant, le plus proche du point de départ. Le programme va ensuite chercher le chemin pour aller au prochain restaurant le plus proche en s'assurant de ne pas repasser sur un point déjà visité. Cette étape est exécutée jusqu'à avoir le nombre d'arrêts souhaités ou jusqu'à ne plus pouvoir ajouter de restaurant sans dépasser la distance maximale acceptée.

Si la distance souhaitée n'est pas atteinte, on va alors compléter le parcours en rajoutant des routes jusqu'à atteindre la distance souhaitée.

L'algorithme sélectionne les restaurants au fur et à mesure du trajet en prenant le plus proche de la position courante et qui est du type souhaité par l'utilisateur.

L'algorigramme de ce processus est présenté ci-contre.

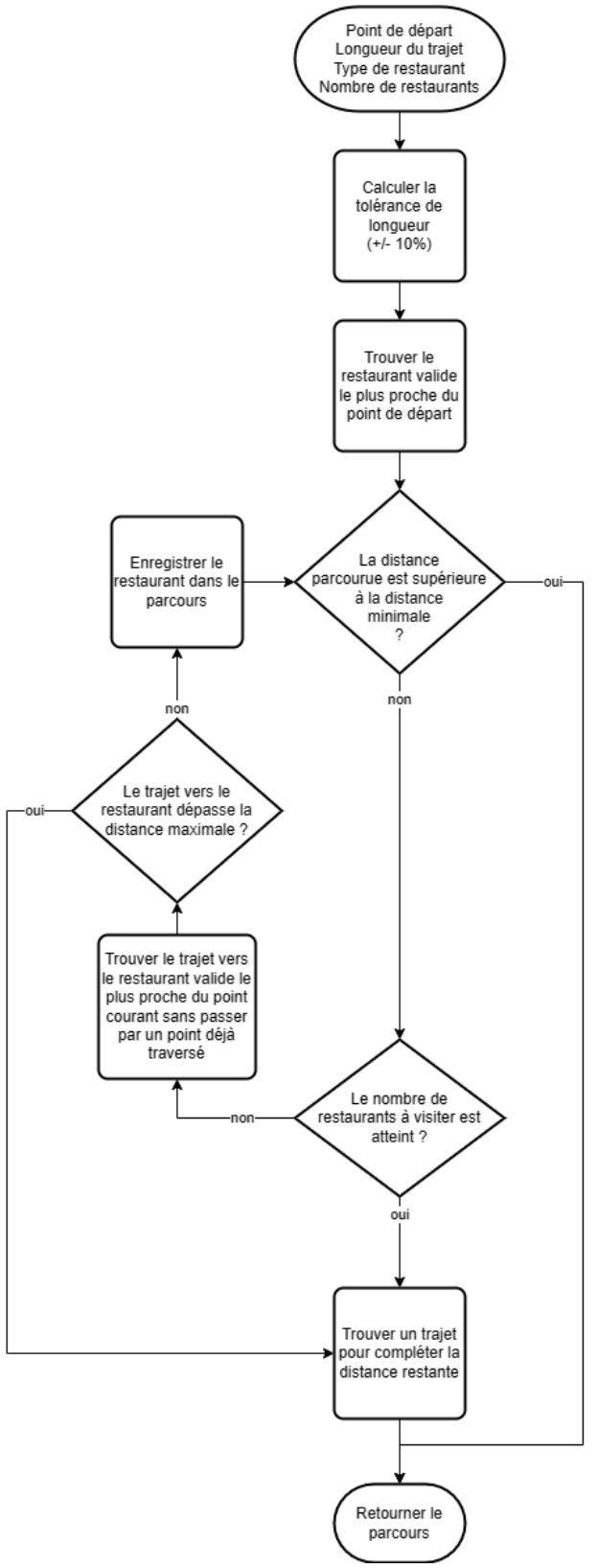


Figure 7 : Algorigramme de production d'un parcours

5.2. L'algorithme de calcul du parcours le plus intéressant pour l'usager

Dans un premier temps, nous effectuons une requête "GET" sur le chemin "/type" qui nous retourne les différents types de restaurants. Cela permettra ainsi à l'utilisateur de sélectionner le type de restaurant à découvrir. Une fois la sélection effectuée, on la donne en paramètre d'un "POST" sur "/starting_point" accompagnée du point de départ, de la longueur de parcours souhaitée, et du nombre d'arrêts.

Ensuite, il ne nous reste qu'à exécuter le "POST" sur "/parcours" qui nous retourne le parcours à effectuer.

Le code nous permet également de vérifier les informations fournies par l'utilisateur. Si certaines informations sont incorrectes, la requête POST sur "/parcours" renvoi un code d'erreur.

L'algorigramme de ce processus est présenté ci-contre.

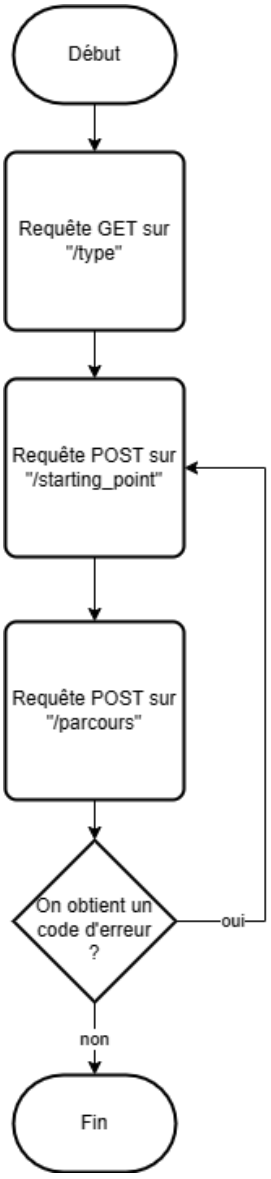


Figure 8 : Algorigramme d'obtention d'un parcours

6. Une explication du plan d'expansion

Nous estimons que l'application va avoir une croissance progressive de l'ordre de 20 à 30% de nouveaux utilisateurs réguliers de l'application par mois.

Pour suivre cette expansion, nous avons identifié 2 métriques importantes :

- **Le nombre d'utilisateurs actifs de l'application** : Cette métrique vise à quantifier le nombre d'utilisateurs actifs de l'application. Un taux d'utilisation croissant indique que l'application est appréciée et que les utilisateurs en sont satisfaits. En revanche, si le nombre d'utilisateurs stagne ou diminue cela signifiera que l'application n'arrive pas à garder actifs les utilisateurs indiquant un besoin d'améliorations.

- **Le nombre de partenariats avec les restaurateurs** : Cette métrique évaluera la quantité de restaurateurs partenaires qui utilisent l'application comme plateforme de vente. L'augmentation du nombre de partenaires est tout aussi importante car elle permet de quantifier l'offre disponible mais aussi d'évaluer notre crédibilité et attractivité. De plus, plus le nombre de partenaires augmente, plus cela renforce notre position sur le marché.

Ces métriques fourniront des indications importantes sur la perception de notre application, et nous permettrons de pouvoir cibler nos efforts sur des améliorations à prévoir.

6.1. Stratégie de réplication

Notre stratégie de réplication repose sur un objectif de proximité géographique et de diminution de la latence relative au traitement des requêtes. En effet, lors de la prise en compte de nouvelles villes dans l'application, il serait pertinent de conserver les données des restaurants et des pistes cyclables dans des centres de données au plus près des villes concernées. Nous utiliserons alors une réplication de type multi leader qui nous permettra de segmenter les données et les requêtes par région géographique.

Cette stratégie nous permettra des gains en performance et de devenir plus résilient aux pannes des centres de données puisque qu'une panne dans l'un des centres n'impactera pas les autres. Cela améliora donc l'expérience utilisateur et aura à terme un impact positif sur le nombre d'utilisateurs actifs de l'application.

6.2. Stratégie de partitionnement

Dans le cadre de la gestion de nos données, nous prévoyons de partitionner la base de données en fonction des informations relatives aux restaurants. C'est pourquoi nous prévoyons de partitionner selon 2 grandes catégories :

- Les informations essentielles : Cette catégorie sera composée des adresses des restaurants, leur type et le statut pour savoir s'il est ouvert. Ces informations essentielles doivent rester accessible le plus rapidement possible pour offrir un calcul rapide et précis lors de la recherche d'itinéraire.

- Les informations non essentielles : Cette catégorie sera composée par exemple des avis et des commentaires de restaurants qui ne sont pas indispensables au bon fonctionnement de l'application. Néanmoins même si elles ne sont pas vitales, elles restent pour la plupart des fonctionnalités de l'application.

Cette approche de partitionnement permettra une gestion plus efficace des données en concentrant les ressources sur les informations vitales de l'application.

6.3. Sécurité des bases de données

Dans l'application, les restaurateurs pourront modifier les informations de leurs restaurants. Pour cela, il faut que les restaurateurs aient un accès spécifique à l'application et à la base de données. En effet, pour une question de sécurité des données, les utilisateurs ne peuvent pas avoir ces accès. Nous devons donc adopter une stratégie de contrôle des privilèges. Ainsi, au moins trois rôles seront créés, dans l'ordre du plus ou moins de privilèges :

- Administrateur : accès le plus permissif
- Restaurateur : accès en lecture et en écriture aux informations uniquement de son restaurant
- Utilisateur : accès en lecture aux résultats de ses requêtes et aux informations des restaurants

De plus, les échanges entre les différents types d'utilisateurs et les serveurs nécessiteront d'encrypter les communications, par exemple pour gérer les connexions (et donc les mots de passe) des utilisateurs.

Annexes

Annexe A : Exemple de données de l'API Yelp

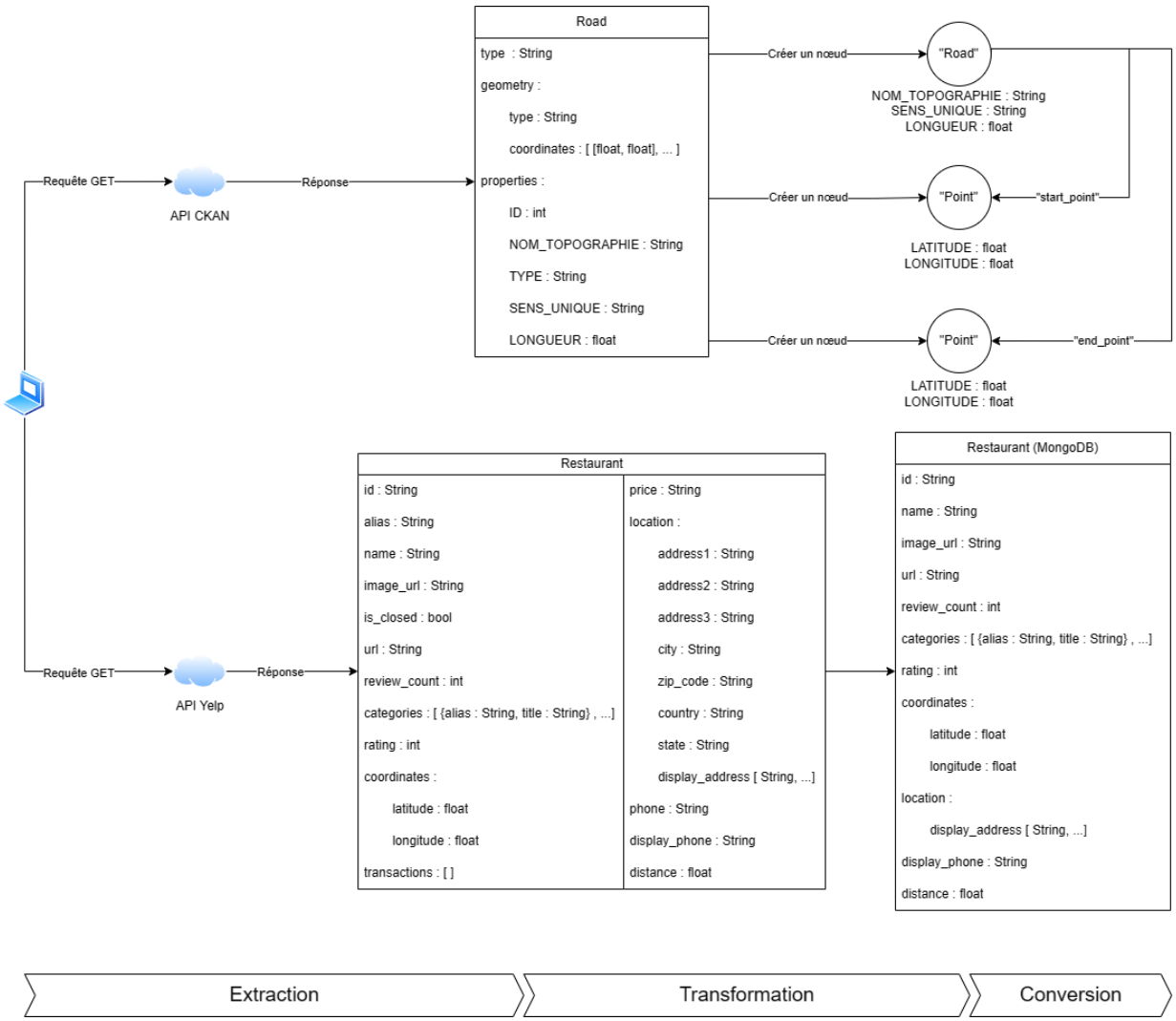

```
{'id': 'NSbVtd0lzORHV7NaubG9Pg',
 'alias': 'la-bûche-québec-2',
 'name': 'La Bûche',
 'image_url': 'https://s3-media3.fl.yelpcdn.com/bphoto/gXN6UVdsHCG33TXr5gmYKQ/o.jpg',
 'is_closed': False,
 'url': 'https://www.yelp.com/biz/la-b%C3%BBche-qu%C3%A9bec-2?adjust_creative=bd-gMXOsqj_3B-T138C_YQ&utm_campaign
 =yelp_api_v3&utm_medium=api_v3_business_search&utm_source=bd-gMXOsqj_3B-T138C_YQ',
 'review_count': 659,
 'categories': [{'alias': 'newcanadian', 'title': 'Canadian (New)'},
  {'alias': 'french', 'title': 'French'}],
 'rating': 4.0,
 'coordinates': {'latitude': 46.81125, 'longitude': -71.20789},
 'transactions': [],
 'price': '$$',
 'location': {'address1': '49 Rue Saint-Louis',
  'address2': '',
  'address3': '',
  'city': 'Quebec City',
  'zip_code': 'G1R 3Z2',
  'country': 'CA',
  'state': 'QC',
  'display_address': ['49 Rue Saint-Louis',
   'Quebec City, QC G1R 3Z2',
   'Canada']},
 'phone': '+14186947272',
 'display_phone': '+1 418-694-7272',
 'distance': 2714.4387733704843},
```


Annexe B : Exemples de données de l'API CKAN


```
var value = {
    "type": "Feature",
    "geometry": {
        "type": "LineString",
        "coordinates": [
            [
                -71.24003988088926,
                46.83234451991697
            ],
            [
                -71.24102482723642,
                46.83198038857159
            ]
        ]
    },
    "properties": {
        "ID": 100094,
        "NOM_TOPOGRAPHIE": "Boulevard des Alliés",
        "TYPE": "Chaussée désignée",
        "SENS_UNIQUE": "N",
        "LONGUEUR": 85.35
    }
}
```

```
var value =         {
    "type": "Feature",
    "geometry": {
        "type": "LineString",
        "coordinates": [
            [
                -71.24102482723642,
                46.83198038857159
            ],
            [
                -71.24198236392772,
                46.83163532779201
            ],
            [
                -71.24206581139462,
                46.83160525282929
            ]
        ]
    },
    "properties": {
        "ID": 100095,
        "NOM_TOPOGRAPHIE": "Boulevard des Alliés",
        "TYPE": "Chaussée désignée",
        "SENS_UNIQUE": "N",
        "LONGUEUR": 89.7
    }
}
```

Annexe C : Diagramme de la pipeline d'ETL de l'application



Annexe D : Base de données Neo4J (Nouvelle version du livrable 3)