

Équipe de travail - Équipe06- :

Hiba Habri - 537066308

Hugo Mazéas - 537004098

Yassine Nouri - 537231375

## **Remise et présentation écrite du produit minimum viable**

Travail présenté à Eloïse Prévôt

Base de données avancés

Faculté science et génie

Université Laval

20 Décembre 2024

# Table des matières

1. Introduction.....	1
1.1 Les produits alimentaires de base et d'Open Food Facts .....	1
1.2 Problématiques .....	1
1.3 Sommaire de la solution.....	1
1.4 Présentation du rapport .....	2
1.5 Public cible du rapport .....	2
2. Stratégie d'acquisition des données .....	3
2.1 Source et méthode d'extraction .....	3
2.2 Exemples de données .....	4
3. Technologies utilisées .....	5
3.1 Langage de programmation.....	5
3.1.1 NodeJS.....	5
3.1.2 ExpressJS .....	5
3.2 Bases de données.....	5
3.2.1 MySQL .....	5
3.2.2 MongoDB .....	6
4. Les détails du processus d'extraction, de transformation et de conversion (ETL) .....	6
4.1 Processus d'acquisition initiale des données .....	6
4.2 Processus d'acquisition incrémental des données .....	7
4.3 Processus de transformation des données .....	9
4.3.1 MySQL : .....	9
4.3.2 MongoDB .....	9
4.4 Schéma du pipeline d'ETL .....	10
5. Les détails du pipeline de données .....	10
5.1 Création de recommandations de produits .....	10
5.1.1 Explication visuelle et textuelle .....	10
5.1.2 Algorithme.....	10
5.1.3 Diagramme de séquence.....	11
5.2 Obtention de recommandations de produits .....	11
5.2.1 Explication visuelle et textuelle .....	11
5.2.2 Algorithme :.....	12

5.2.3 Diagramme de séquence circulaire .....	12
6. Une explication du plan d'expansion .....	12
6.1 Description .....	12
6.1.1 Scénario d'expansion réaliste .....	12
6.2.2 Deux métriques de charge cohérentes avec le scénario .....	13
6.2.3 Impact attendu de l'expansion sur les métriques .....	13
6.2 Réplication .....	14
6.2.1 Type de réplication .....	14
6.2.2 Impact sur le volume de requêtes par seconde (RPS) : .....	14
6.3 Partitionnement .....	14
6.3.1 Fragmentation de la base MySQL : .....	14
6.3.2 Fragmentation de la base MongoDB : .....	14
6.3.3 Impact sur le volume de requêtes par seconde (RPS) : .....	15
6.4 Sécurité .....	15
6.4.1 : Cas d'utilisation 1 : Injection SQL dans la base MySQL .....	15
6.4.2 Cas d'utilisation 2 : Modification illégitime des données de produits alimentaires .....	15
7. Fonctionnalités additionnelles avancées .....	16
7.1 Interfaces web : .....	16
7.2 Authentification : .....	16
7.3 Script Bash pour l'ajout de recette : .....	16
7.4 Recherche .....	16
Annexe 0 : Résultats Remise 2 .....	18
Annexe A: Extrait de données Open Food Facts .....	19
Annexe B: Extrait de USDA Global Branded Food Products Database .....	20
Annexe C: Extrait de données Recettes .....	21
Annexe D: Fonctionnement NodeJS .....	22
Annexe E : Fonctionnement ExpressJS .....	23
Annexe F : UML filtrage JSON .....	24
Annexe G: UML Scraping site web des recettes .....	25
Annexe H : Script Bash nettoyage liste ingrédients .....	26
Annexe I : Transformation CSV-MySQL .....	27
Annexe J : Script Bash mise à jour liste ingrédients .....	28
Annexe K : Document OFF transformé .....	29
Annexe L : Script de transformation de la BD USDA Branded .....	30

Annexe M : BD USDA Branded transformée .....	31
Annexe N : ETL général .....	32
Annexe O-1 : Diagramme de séquence création de recommandations.....	33
Annexe O-2 : Diagramme de séquence obtention de recommandations.....	33
Annexe P : Fragmentation BD MySQL .....	34
Annexe Q : Script Bash ajout recette BD MySQL .....	35
Annexe R : Page d'authentification.....	36
Annexe S : Page de recherche en action .....	37
Annexe T : Interfaces Web .....	38
Bibliographie .....	41

# 1. Introduction

## 1.1 Les produits alimentaires de base et d'Open Food Facts

Open Food Facts est une base de données collaborative qui répertorie des informations détaillées sur des milliers de produits alimentaires. Fondée en 2012, elle permet aux utilisateurs d'accéder à des données telles que la composition des produits, leur valeur nutritionnelle et leur impact écologique. Ces informations contribuent à sensibiliser les consommateurs aux choix alimentaires durables et sains.

Les produits alimentaires de base incluent des aliments essentiels à la préparation de recettes courantes, comme les fruits, légumes, céréales, et produits laitiers. Leur identification et classification dans une base de données permettent de mieux répondre aux besoins des consommateurs et des cuisiniers.

## 1.2 Problématiques

Dans un monde où les consommateurs recherchent des choix alimentaires éclairés et personnalisés, il est difficile de :

- Accéder à des informations fiables sur les produits alimentaires en temps réel.
- Identifier des produits correspondant précisément aux besoins culinaires ou nutritionnels.
- Intégrer efficacement des données provenant de sources multiples pour proposer des recommandations pertinentes.

L'application vise à résoudre ces problématiques en fusionnant des bases de données variées (Open Food Facts, USDA, recettes) et en offrant des algorithmes de recommandation précis.

## 1.3 Sommaire de la solution

La solution proposée repose sur un pipeline de données ETL (Extract, Transform, Load) bien défini, capable de gérer des sources multiples et de répondre aux exigences d'un système polyglotte. Les principales étapes incluent :

1. Extraction des données depuis des sources hétérogènes.

2. Transformation des données pour les nettoyer, les normaliser et les enrichir.
3. Chargement des données transformées dans une base MySQL pour les recettes et une base MongoDB pour les produits alimentaires. De plus, des algorithmes dédiés sont conçus pour proposer des produits adaptés en fonction des ingrédients de recettes ou des besoins spécifiques des utilisateurs.

## 1.4 Présentation du rapport

Ce rapport présente les détails de l'implémentation de la solution. Les sections suivantes décrivent :

- ❖ Les stratégies d'acquisition des données.
  - ❖ Les technologies utilisées pour gérer et intégrer les données.
  - ❖ Les détails du pipeline ETL, incluant les processus incrémentaux et les transformations appliquées.
  - ❖ Les algorithmes permettant de générer des recommandations basées sur les données collectées.
- Chaque section est accompagnée d'exemples de code pour faciliter la compréhension.

## 1.5 Public cible du rapport

Ce rapport s'adresse :

- ✓ Aux développeurs et ingénieurs souhaitant comprendre les aspects techniques du projet, notamment l'intégration de bases de données polyglottes et l'implémentation des algorithmes de recommandation.
- ✓ Aux décideurs et responsables produits intéressés par les applications pratiques des données alimentaires pour améliorer l'expérience utilisateur et promouvoir des choix alimentaires sains et durables.
- ✓ Aux enseignants et évaluateurs évaluant l'approche méthodologique et technique adoptée pour résoudre une problématique réelle.

## 2. Stratégie d'acquisition des données

### 2.1 Source et méthode d'extraction

Pour notre projet l'extraction des données s'est faite à partir de trois sources principales : Open Food Facts [1] , USDA Global Branded Food Products Database [2] et le guide alimentaire canadien pour les recettes chacune ayant apporté des informations complémentaires pour répondre aux objectifs d'analyse des produits alimentaires au Québec.

- Open Food Fact “OFF”: est une base de données collaborative et libre qui recense des informations sur des milliers de produits alimentaires à travers le monde. Fondée en 2012, elle permet aux consommateurs d'accéder à des données détaillées sur la composition des produits, leurs ingrédients, leur valeur nutritionnelle. Les utilisateurs peuvent contribuer en scannant les codes-barres des produits ou en ajoutant des informations manuellement, favorisant ainsi une transparence accrue dans l'industrie alimentaire. OFF s'engage également à offrir des outils pour évaluer la qualité nutritionnelle des produits à l'aide de scores comme le Nutri-Score, tout en soutenant des initiatives de sensibilisation aux choix alimentaires plus sains et durables. Grâce à sa nature ouverte et collaborative, OpenFoodFacts participe à l'amélioration de l'accès à des informations fiables, aidant ainsi les consommateurs à faire des choix éclairés pour leur santé et pour la planète. OFF viens avec plusieurs méthodes de téléchargements, parmi ces méthodes nous avons choisi de télécharger le fichier JSONL disponible [ici](#).
- Le Guide alimentaire canadien, mis à jour régulièrement par Santé Canada, est un outil essentiel pour promouvoir une alimentation saine et équilibrée. Il recommande de privilégier la diversité alimentaire en incluant une grande variété d'aliments dans chaque repas, notamment des fruits et légumes, des protéines de qualité (comme les légumineuses, les viandes maigres et les poissons), ainsi que des grains entiers. Le Guide encourage également à limiter les aliments ultra-transformés, riches en sucres ajoutés, en sel et en graisses saturées. Lors de l'élaboration de recettes. Pour extraire ces recettes il va falloir scraper le site de guide alimentaire pour récupérer la liste de recette.

- Le USDA Global Branded Food Products Database est une initiative de partenariat public-privé visant à améliorer la santé publique et le partage de données ouvertes. Ce projet vise à enrichir le USDA National Nutrient Database avec des informations sur la composition des nutriments et les ingrédients des produits alimentaires de marque et de marque privée fournis par l'industrie alimentaire. Le but est de fournir un accès public à ces données, permettant une évaluation précise de l'étendue et de la fluidité du système alimentaire. La base de données inclut des informations telles que le nom du produit, la taille des portions, les nutriments indiqués sur l'étiquette Nutrition Facts, la liste des ingrédients, et une date associée à la formulation la plus récente du produit. [10]

Cette source a été recommandée par notre point de contact avec OFF. Cette base de données existe sous deux formats CSV ou JSON, nous avons opté pour le téléchargement du format JSON pour sa facilité d'importation avec MongoDB puisque nous avons déjà une collection pour Open Food Facts dans MongoDB.

## 2.2 Exemples de données

Un exemple extrait de Open Food Facts est présenté dans la part [annexe A](#).

Un exemple extrait de guide alimentaire de recette est présenté par [l'Annex C](#). Les données n'existent pas sous forme de fichiers prêt à utiliser, on a dû utiliser la méthode de **scraping** pour obtenir un fichier CSV dont on a présenté une partie.

Un exemple extrait du USDA Global Branded Food Products Database est présenté par [l'annexe B](#).

La base de données OFF ainsi que celle USDA Global Branded Food Products Database comprennent plusieurs informations non utiles pour notre contexte de projet, ainsi que plusieurs aliments ont plus de deux scores qui sont à nul ce qui rend l'existence de ces aliments dans notre base de données moins pertinentes. Nous allons essayer à appliquer des filtres pour réduire ces informations inutiles.



### 3. Technologies utilisées

Pour réaliser ce travail, nous utilisons un environnement Docker avec VSCode.

#### 3.1 Langage de programmation

##### 3.1.1 NodeJS

Node.js est un environnement d'exécution JavaScript **open-source et multiplateforme**. [3]

Node.js est une bibliothèque permettant d'exécuter des applications web en dehors du navigateur du client. Ryan Dahl l'a développé en 2009. Les développeurs utilisent Node.js pour créer des applications web côté serveur, et il est parfait pour les applications gourmandes en données puisqu'il utilise un modèle asynchrone, piloté par les événements. [4]

La figure de l'annexe 1 montre comment fonctionne NodeJS. [5]

##### 3.1.2 ExpressJS

Express.js est un framework web rapide, flexible et minimaliste pour Node.js. Il s'agit en fait d'un outil qui simplifie la création d'applications web et d'API en utilisant JavaScript côté serveur. Express est un logiciel libre développé et maintenu par la fondation Node.js.

Express.js offre un ensemble de fonctionnalités robustes qui améliorent la productivité et rationalisent les applications web. Il facilite l'organisation des fonctionnalités de l'application à l'aide d'intergiciels et de routages. Il ajoute des utilitaires utiles aux objets HTTP Node et facilite le rendu des objets HTTP dynamiques. [6]

L'annexe 2 montre l'utilité de l'utilisation de Express JS avec NodeJS. [7]

#### 3.2 Bases de données

##### 3.2.1 MySQL

Pour la base de données des recettes, nous avons choisi MySQL.

MySQL offre une **haute fiabilité** des données, minimisant les risques de corruption et garantissant la cohérence des informations en respectant les propriétés ACID. En termes de **maintenabilité**, MySQL propose des outils performants et une documentation exhaustive,

facilitant ainsi l'administration, la gestion des bases de données et le dépannage. Enfin, son architecture modulaire et sa compatibilité avec diverses extensions et plug-ins permettent une **grande extensibilité**, répondant aux besoins croissants de notre application. [8]

### 3.2.2 MongoDB

Pour notre base principale, celle d'Open Food Fact, nous avons opté pour l'utilisation de MongoDB. Tout d'abord, il s'agit du type de base de données déjà utilisé par OFF, bien qu'ils proposent divers formats de fichiers tels que CSV, mongodump ou JSONL. De plus, la base de données OFF est **très volumineuse**, et MongoDB **facilite la gestion de grandes quantités de données** sous format BSON, ce qui le rend à la fois **extensible et maintenable**.

Par ailleurs, notre objectif est de fusionner au moins deux sources différentes, impliquant des **structures de données variées**. MongoDB, étant une base de données **documentaire**, offre une **flexibilité accrue** dans ce contexte. C'est pourquoi les deux sources autres que les recettes seront transformées et importées dans MongoDB.

En outre, MongoDB propose **nativement la réplication et le partitionnement** des données, ce qui renforce les principes **d'extensibilité et de fiabilité**.

## 4. Les détails du processus d'extraction, de transformation et de conversion (ETL)

### 4.1 Processus d'acquisition initiale des données

L'acquisition initiale de données a été faite à partir des trois sites déjà mentionnés dans la section 2.

Pour l'OFF, nous avons téléchargé le fichier JSONL qui contient des données partout dans le monde c'est une base énorme de 10g. Chaque aliment est présenté par plusieurs clefs comme son `ecosore_score` et `nutriscore_grade`.

Pour l'USDA Global Branded Food Products Database, on a téléchargé un fichier JSON avec comme information : `description`, `foodNutritients`, `foodNutrientSource`, `amount`, `dataType`, etc...

Et enfin pour les recettes, on a gardé le nom, les ingrédients et l'image de chaque recette en utilisant le scraping.

On a présenté les 2 processus par un diagramme UML en [annexe F](#) et [annexe G](#).

## 4.2 Processus d'acquisition incrémental des données

L'acquisition incrémentale, ou développement incrémental, est une approche où un système ou un produit est construit et amélioré par étapes successives, appelées incréments. Chaque incrément ajoute de nouvelles fonctionnalités ou améliore les fonctionnalités existantes sans interruption du service.

Pour notre application, nous proposons 2 méthodes pour chaque BD :

### MySQL : Détection des Changements

#### Méthode 1 : Utilisation de la colonne `last_updated`

- En ajoutant une colonne `last_updated`, elle peut être utilisée pour identifier les enregistrements ajoutés ou modifiés depuis le dernier import.
- **Processus :**
  1. Stocker la date/heure du dernier import dans une variable (ex. : `dernier_import`).
  2. Exécuter une requête SQL pour extraire uniquement les lignes mises à jour après cette date :

```
SELECT * FROM recettes WHERE last_updated > 'dernier_import';
```

3. Charger ces données dans le pipeline de synchronisation pour traitement.

#### Méthode 2 : Utilisation des journaux binaires (Binlogs)

- Les binlogs enregistrent toutes les modifications apportées aux tables MySQL (INSERT, UPDATE, DELETE).
- Avantages :
  - Permet un suivi des changements plus précis.
  - Réduit le besoin de colonnes spécifiques dans la base de données.

- **Processus :**

1. Configurer MySQL pour activer les binlogs.
2. Utiliser un outil comme Debezium ou MySQL Binary Log Client pour lire ces journaux en temps réel.
3. Filtrer les modifications spécifiques à notre table de recettes.

## **MongoDB : Détection des Changements**

### **Méthode 1 : Utilisation des Timestamps**

- Nous ajouterons aux documents MongoDB un champ `date_modified` ou similaire, il peut être utilisé pour détecter les changements.
- **Processus :**
  1. Stocker la date/heure du dernier import dans une variable (ex. : `lastImportDate`).
  2. Exécuter une requête pour récupérer les documents modifiés après cette date :  
`db.aliments.find({ date_modified: { $gt: lastImportDate } });`
  3. Traiter les documents récupérés pour les synchroniser.

### **Méthode 2 : Utilisation de Change Streams**

- MongoDB propose un mécanisme natif pour suivre les changements en temps réel via des Change Streams.
- **Avantages :**
  - Idéal pour des besoins en temps réel.
  - Permet de détecter les INSERT, UPDATE et DELETE.
- **Processus :**
  1. Activez les Change Streams sur votre collection MongoDB.
  2. Configurez un script pour écouter les événements et capturer les changements.

```
const changeStream = db.aliments.watch();  
changeStream.on('change', (change) => {  
  console.log(change); });
```

## 4.3 Processus de transformation des données

### 4.3.1 MySQL :

Pour la base de données des recettes après avoir faire le scraping, le résultat était un fichier CSV. Nous avons commencé par appliquer **un script Bash (annexe H)** pour nettoyer la liste des ingrédients pour chaque recette pour les préparer à la fonction de recommandation. Le script supprime toutes les unités de mesure telle que ml, l, boîte etc. et les numéros. Il supprime aussi les espaces vides, les propositions non utile comme « d' » etc..

On a utilisé ce fichier transformé et le converti en SQL en utilisant le site web « <https://convertcsv.com/csv-to-sql.htm> » pour créer la BD via ce site web. Le résultat est illustré dans [l'annexe I](#).

Après avoir créer la BD mysql, au fil de temps, nous avons constaté des ingrédients qui ne sont pas bien nettoyé, nous avons fait le nettoyage manuellement car se sont des petits changements uniques sans pattern global, mais nous avons automatiser la mise à jour de la BD par le script Bash de [l'annexe J](#).

### 4.3.2 MongoDB

Pour les 2 base de données OFF et USDA Global Branded Food Products Database, nous avons appliqué quelques commandes « **jq** » (json query) pour filtrer les données.

Tout d'abord pour OFF, un premier filtre pour faire sortir les données concernant le Canada :

```
zcat openfoodfacts-products.jsonl.gz | jq -c '. | select(.countries_tags[]? == "en:canada")' >
canada2.jsonl
```

Puis, garder les aliments qu'ont de Nutriscore. Et enfin garder quelques attributs utiles pour cette phase comme le nom de l'aliment, ses scores et sa catégorie :

```
cat canada2.jsonl | jq -c '. | select(.misc_tags[]? == "en:nutriscore-computed") |
[.code,.product_name,.ecoscore_score,.nutriscore_grade,.nova_group,.categories,.brands] ' >
canada-filtered.jsonl
```

Le résultat final est présenté dans [l'annexe K](#).

Pour l'USDA Global Branded Food Products Database, nous avons gardé pour chaque aliment : sa description, et pour chaque foodNutrient, son nom, son rank, et sa valeur en utilisant le script de [l'annexe L](#). Le résultat est illustré dans [l'annexe M](#).

#### *4.4 Schéma du pipeline d'ETL*

ETL, pour Extract Transform Load, son but est de gérer et intégrer des données provenant de diverses sources pour les rendre exploitables dans un système cible. Dans notre projet, nous allons modifier deux fichiers JSON et scraper un site web pour avoir nos 3 bases de données.

Les deux fichiers JSON transformés avec la commande « **jq** » et un script JS puis chargés dans mongodb avec la commande **mongoimport**.

Pour la base de données recettes, nous avons l'importé directement à MySQL avec la commande **source** après avoir faire le web scraping et exécuter un script Bash.

Le schéma général est disponible en [annexe N](#).

## **5. Les détails du pipeline de données**

### *5.1 Création de recommandations de produits*

#### *5.1.1 Explication visuelle et textuelle*

Les recommandations de produits sont générées à partir des ingrédients d'une recette. Le processus est divisé en deux étapes principales : la correspondance des ingrédients avec des aliments dans la base de données et la sélection des produits en fonction des préférences de l'utilisateur.

#### *5.1.2 Algorithme*

##### **Étape 1 :** Correspondance des ingrédients

1. Entrée : Liste d'ingrédients extraits d'une recette (issus de la base MySQL).
2. Traitement :

Chaque ingrédient est recherché dans la base MongoDB en utilisant une requête basée sur des expressions régulières.

Les correspondances incluent des aliments associés à chaque ingrédient dans la base Open Food Facts.

Une fois les correspondances trouvées, des métadonnées (comme le nutriscore, les catégories bio/locales) sont récupérées.

3. Sortie : Liste de correspondances entre les ingrédients et les aliments.

## **Étape 2** : Sélection des meilleurs produits

1. Entrée : Liste des aliments correspondants pour chaque ingrédient.
2. Traitement :

Les aliments sont filtrés selon les préférences de l'utilisateur (par exemple, bio, faible en gras, local).

Chaque aliment est trié en fonction de critères tels que le nutriscore, le prix ou les avis utilisateurs.

3. Sortie : Liste des produits recommandés pour la recette.

### *5.1.3 Diagramme de séquence*

Veuillez trouver le diagramme de séquence dans [l'annexe O-1](#).

Le diagramme illustre le flux de données suivant :

1. Extraction des ingrédients d'une recette depuis la base MySQL.
2. Recherche d'aliments correspondants dans MongoDB via des requêtes.
3. Filtrage et tri des aliments selon les critères utilisateur.

## *5.2 Obtention de recommandations de produits*

### *5.2.1 Explication visuelle et textuelle*

Les recommandations sont obtenues après la création des produits recommandés. Elle inclut une explication textuelle et un diagramme de séquence pour détailler les interactions entre les bases de données et l'application.

### 5.2.2 Algorithme :

#### **Étape 1 :** Recherche des aliments nécessaires pour une recette

1. Entrée : Identifiant de la recette.
2. Traitement :

Les ingrédients de la recette sont extraits de MySQL.

Les aliments correspondants sont recherchés dans MongoDB.

3. Sortie : Liste des aliments nécessaires pour réaliser la recette.

#### **Étape 2 :** Retour des produits optimaux

1. Entrée : Liste des aliments nécessaires.
2. Traitement :

Les aliments sont filtrés et triés pour ne conserver que les produits respectant les préférences utilisateur.

La liste finale est structurée et renvoyée au front-end.

3. Sortie : Liste des recommandations affichées à l'utilisateur.

### 5.2.3 Diagramme de séquence circulaire

Le diagramme de [l'annexe O-2](#) détaille les étapes suivantes :

1. Interaction utilisateur : sélection d'une recette.
2. Extraction des données depuis MySQL.
3. Correspondance et filtrage des aliments via MongoDB.
4. Renvoi des recommandations structurées au front-end.

## 6. Une explication du plan d'expansion

### 6.1 Description

#### 6.1.1 Scénario d'expansion réaliste

Dans une optique d'expansion, l'application pourrait être déployée à l'échelle internationale pour intégrer les données d'OFF pour d'autres pays (nous avons commencé par les données de



l'USA) et d'autres données provenant de nouvelles sources alimentaires (bases locales ou régionales) et répondre à des besoins spécifiques des consommateurs dans différentes zones géographiques. Cette expansion inclurait l'ajout de nouvelles recettes (exemple **Ciqual** de la France), de produits locaux, et de critères de personnalisation plus avancés, comme les préférences culturelles ou les restrictions alimentaires.

### 6.2.2 Deux métriques de charge cohérentes avec le scénario

#### 1. Volume de requêtes par seconde (RPS) :

- Avec une expansion internationale, le nombre de requêtes simultanées augmentera considérablement.

- Cette métrique permettra de suivre la capacité du système à gérer un flux de demandes accru.

#### 2. Temps moyen de réponse (RTT) :

- Le temps nécessaire pour générer une recommandation, incluant l'extraction des données, leur transformation, et la livraison des résultats.

- Une expansion nécessitera de maintenir un temps de réponse faible malgré l'augmentation du volume de données.

### 6.2.3 Impact attendu de l'expansion sur les métriques

#### 1. Volume de requêtes par seconde (RPS) :

- Augmentation significative due à un public plus large et des zones géographiques supplémentaires.

- Des solutions comme le scaling horizontal (ajout de serveurs) et la mise en cache des résultats fréquents seront nécessaires.

#### 2. Temps moyen de réponse (RTT) :

- Risque d'augmentation si les optimisations ne sont pas effectuées, comme le pré-chargement des données critiques ou l'utilisation de bases de données distribuées.

- Des optimisations d'algorithmes, comme la réduction du nombre de requêtes par cycle, seront mises en œuvre pour limiter cet impact.

## 6.2 Réplication

### 6.2.1 Type de réplication

Pour assurer une disponibilité élevée et une tolérance aux pannes, l'application utiliserait une réplication « maître-esclave » / « single leader », la plus répandue pour les bases de données MySQL et pour MongoDB et vu que nous n'avons pas trop d'écritures fréquentes. Dans ce modèle, un serveur principal (maître) gère toutes les écritures, tandis que les serveurs secondaires (esclaves) reçoivent une copie des données pour gérer les lectures.

### 6.2.2 Impact sur le volume de requêtes par seconde (RPS) :

- La réplication maître-esclave permettra de répartir la charge des requêtes en lecture sur les serveurs esclaves, augmentant ainsi la capacité du système à gérer un volume accru de requêtes simultanées.

- En cas de panne du serveur maître, un basculement rapide vers un esclave minimisera l'interruption du service, maintenant une expérience utilisateur fluide même en cas de défaillance.

## 6.3 Partitionnement

### 6.3.1 Fragmentation de la base MySQL :

Pour la BD des recettes, puisque les catégories sont assez réduites, nous optons pour une « fragmentation par Liste ». Cela permet de séparer les recettes en partitions distinctes basées sur leurs catégories. L'exemple de la création de la table avec fragmentation est démontré par l'[annexe P](#).

### 6.3.2 Fragmentation de la base MongoDB :

Pour les produits alimentaires de MongoDB, chaque fragment contiendra des produits basés sur des critères géographiques ou catégoriels (exemple : produits locaux, bio, ou par région) qu'on va ajouter avec l'expansion de notre application, exemple :

```
{ name: "Carotte", category: "bio", region: "Ontario", nutriscore: "B" },
```

Cela sera fait avec l'activation de l'option « **Sharding** » du mongoddb :

```
sh.enableSharding("off_db")
```

Et l'affecter par exemple par région :

```
sh.shardCollection("off_db.aliments", { region: 1 })
```

### 6.3.3 Impact sur le volume de requêtes par seconde (RPS) :

- Le partitionnement permet de réduire la charge sur chaque nœud de base de données en dirigeant les requêtes vers les fragments pertinents uniquement.
- Cela améliore l'évolutivité en permettant des écritures et des lectures parallèles sur plusieurs fragments.
- En conséquence, le système pourra gérer un nombre beaucoup plus élevé de requêtes simultanées, avec un impact positif direct sur le RPS.

## 6.4 Sécurité

### 6.4.1 : Cas d'utilisation 1 : Injection SQL dans la base MySQL

- **Risque** : Un attaquant pourrait exploiter des failles dans les requêtes SQL pour exécuter des commandes malveillantes, compromettant ainsi la confidentialité et l'intégrité des données.

- **Stratégie de mitigation** :

- Utilisation de requêtes préparées pour éviter l'injection de commandes SQL malveillantes.
- Validation stricte des entrées utilisateur pour s'assurer qu'elles respectent les formats attendus.
- Implémentation d'un pare-feu d'applications web (WAF) pour détecter et bloquer les tentatives d'injection SQL.

### 6.4.2 Cas d'utilisation 2 : Modification illégitime des données de produits alimentaires

- **Risque** : Un attaquant pourrait altérer les informations des produits dans MongoDB, compromettant ainsi l'intégrité des recommandations.

### - Stratégie de mitigation :

- Mise en place d'un système d'audit pour enregistrer toutes les modifications effectuées sur les données.

- Activation de la fonctionnalité `readOnly` pour les utilisateurs qui n'ont pas besoin d'accéder aux fonctionnalités d'écriture dans MongoDB.

## 7. Fonctionnalités additionnelles avancées

### 7.1 Interfaces web :

L'interface web permet de rechercher des recettes, explorer des ingrédients et accéder aux produits par marque via une navigation intuitive. Elle simplifie l'exploration des données transformées en proposant des ingrédients répertoriés pour chacune des recettes. L'interface peut être consulté à [l'annexe T](#).

### 7.2 Authentification :

L'authentification est requise uniquement pour accéder aux pages de l'interface utilisateur. L'utilisateur unique (username : prof password : prof) permettent une connexion simple, idéale pour limiter l'accès aux fonctionnalités avancées. L'utilisateur, bien qu'étant unique, est entreposé dans la base de données MySQL. La page d'authentification à l'application est montrée à [l'annexe R](#).

### 7.3 Script Bash pour l'ajout de recette :

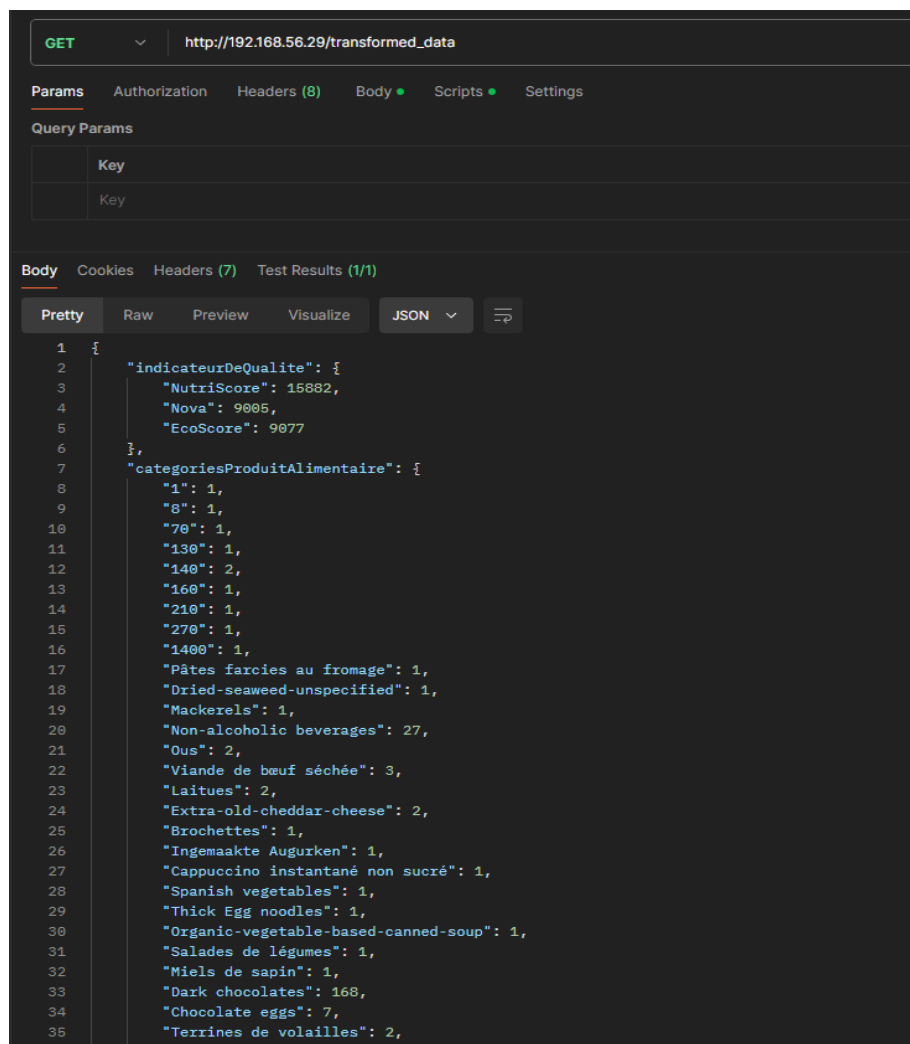
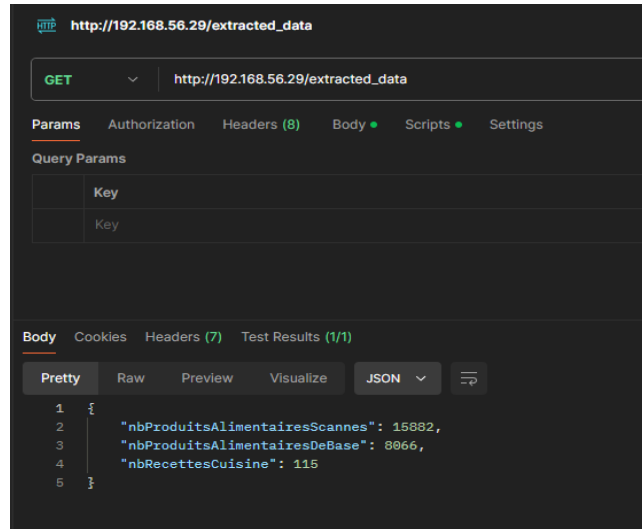
Nous avons préparé le script Bash précédemment mentionnée et l'adopter pour ajouter des recettes par la suite sans perturber le fonctionnement de notre application. Une partie du script est démontrée par [l'annexe Q](#).

### 7.4 Recherche

La fonctionnalité de recherche dans les tableaux est assurée par une bibliothèque JavaScript nommée DataTables. Elle permet une recherche instantanée à travers toutes les colonnes, avec la

prise en charge des mots partiels, offrant une méthode efficace pour interagir avec les données présentées puis avoir un large éventail de résultat plus tôt que de risquer d'afficher aucun résultat. Une démonstration de la page recherche en action est présentée à [l'annexe S.](#)

## Annexe 0 : Résultats Remise 2



## Annexe A: Extrait de données Open Food Facts

```
{
  "_id": "0008577002786",
  "ecoscore_score": 64,
  "food_groups_tags": [],
  "data_quality_bugs_tags": [],
  "data_sources_imported": "Databases, database-usda",
  "last_image_t": 1660213050,
  "ingredients_analysis_tags": [
    "en:palm-oil-free",
    "en:vegan",
    "en:vegetarian"
  ],
  "lc_imported": "en",
  "correctors_tags": [
    "usda-ndb-import",
    "teolemon",
    "fix-missing-lang-bot",
    "org-database-usda",
    "foodvisor",
    "org-label-non-gmo-project",
    "fighter-food-facts"
  ],
  "allergens_from_ingredients": "",
  "serving_quantity_unit": "ml",
  "allergens_tags": [],
  "product_name_en_imported": "100% pure vermont organic maple syrup, dark color robust taste",
  "manufacturing_places": "",
  "nutrition_grades_tags": [
    "d"
  ],
  "nutriscore_grade": "d",
  "cities_tags": [],
  "created_t": 1460234918,
  "last_editor": "fighter-food-facts",
  "removed_countries_tags": [],
  "ingredients_non_nutritive_sweeteners_n": 0,
  "ingredients_text": "Pure organic maple syrup",
  "emb_codes": "",
}
```

## Annexe B: Extrait de USDA Global Branded Food Products Database

```
{
  "foodClass": "Branded",
  "description": "MANGO CHIA POD, MANGO",
  "foodNutrients": [
    {
      "type": "FoodNutrient",
      "id": 13796296,
      "nutrient": {
        "id": 1003,
        "number": "203",
        "name": "Protein",
        "rank": 600,
        "unitName": "g"
      },
      "foodNutrientDerivation": {
        "code": "LCCS",
        "description": "Calculated from value per serving size measure",
        "foodNutrientSource": {
          "id": 9,
          "code": "12",
          "description": "Manufacturer's analytical; partial documentation"
        }
      },
      "amount": 1.76
    }
  ]
}
```



## Annexe C: Extrait de données Recettes

id	nomRecette	ingredientsRecette	imgUrl					
1	Salade de pois chiches et de carottes	<p>Ingrédients</p> <p>1 grand concombre, tranché</p> <p>15 ml (1 c. à table) d'aneth frais haché ou 2 ml (½ c. à thé) d'aneth séché</p> <p>2 carottes, épluchées et râpées</p> <p>1 boîte (540 ml/19 oz) de pois chiches, égouttés et rincés</p> <p>500 ml (1 tasse) de tomates cerise, coupées en deux dans le sens de la longueur</p> <p>60 ml (¼ tasse) de basilic frais haché</p> <p>45 ml (3 c. à table) de vinaigre balsamique</p> <p>15 ml (1 c. à table) de pesto de basilic</p> <p>10 ml (2 c. à thé) d'huile d'olive extra vierge</p> <p>1 gousse d'ail, émincée</p> <p>1 ml (¼ c. à thé) de poivre moulu</p>	<a href="https://guide-alimentaire.canada.ca/sites/default/files/styles/container_width/public/2020-07/chickpea_carrot_salad.jpg">https://guide-alimentaire.canada.ca/sites/default/files/styles/container_width/public/2020-07/chickpea_carrot_salad.jpg</a>					
2	Pain aux bananes et aux noix de Grenoble	<p>Ingrédients</p> <p>2 œufs</p> <p>85 ml (⅓ tasse) de sirop d'érable</p> <p>5 ml (1 c. à thé) d'extrait de vanille</p> <p>125 ml (½ tasse) d'huile végétale</p> <p>3 bananes, écrasées</p> <p>250 ml (1 tasse) de farine de blé entier</p> <p>190 ml (¾ tasse) de farine tout usage</p> <p>5 ml (1 c. à thé) de bicarbonate de soude</p> <p>5 ml (1 c. à thé) de cannelle</p> <p>1 ml (¼ c. à thé) de sel</p> <p>250 ml (1 tasse) de noix de Grenoble non salées, grillées et hachées</p>	<a href="https://guide-alimentaire.canada.ca/sites/default/files/styles/container_width/public/2022-05/Banana%20loaf%203.jpg">https://guide-alimentaire.canada.ca/sites/default/files/styles/container_width/public/2022-05/Banana%20loaf%203.jpg</a>					

## Annexe D: Fonctionnement NodeJS

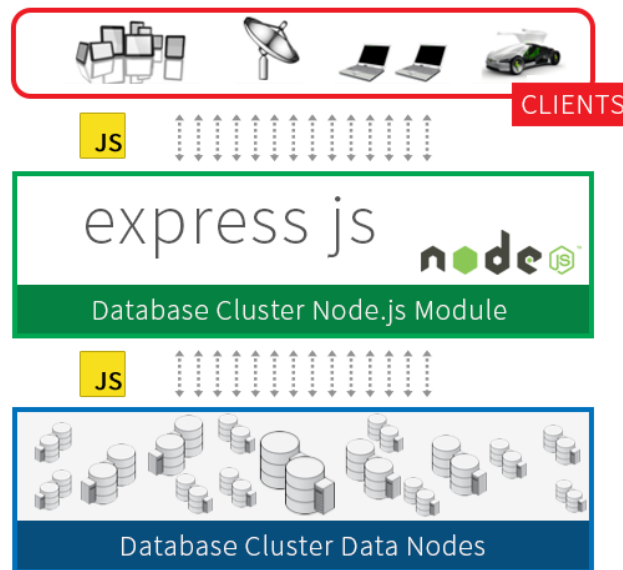


*Fonctionnement NodeJS*

Même s'il n'y a qu'une seule boucle d'événement, lorsqu'une demande est faite, la boucle transmet la demande à une fonction asynchrone qui effectue le travail. Lorsque cette fonction est terminée et qu'une réponse est renvoyée, elle peut alors être renvoyée à la boucle d'événement pour être exécutée par le callback et envoyée à l'utilisateur. Si les fonctions étaient synchrones, la boucle d'événements serait bloquée par la demande et la réponse d'un client, et tous les autres clients devraient attendre que ce client ait terminé. Grâce à la nature asynchrone de JavaScript, les applications utilisant Node peuvent gérer de nombreuses requêtes en même temps.

Cela signifie que lors de la programmation en Node.js, il est important de toujours garder à l'esprit que les fonctions écrites ne sont pas synchrones. Il est également très important d'attraper les erreurs sur le serveur avant qu'elles ne soient renvoyées au client. Cela permet d'éviter que des erreurs n'atteignent la boucle d'événements, ce qui pourrait faire planter le programme et tous les clients en souffriraient.

## Annexe E : Fonctionnement ExpressJS

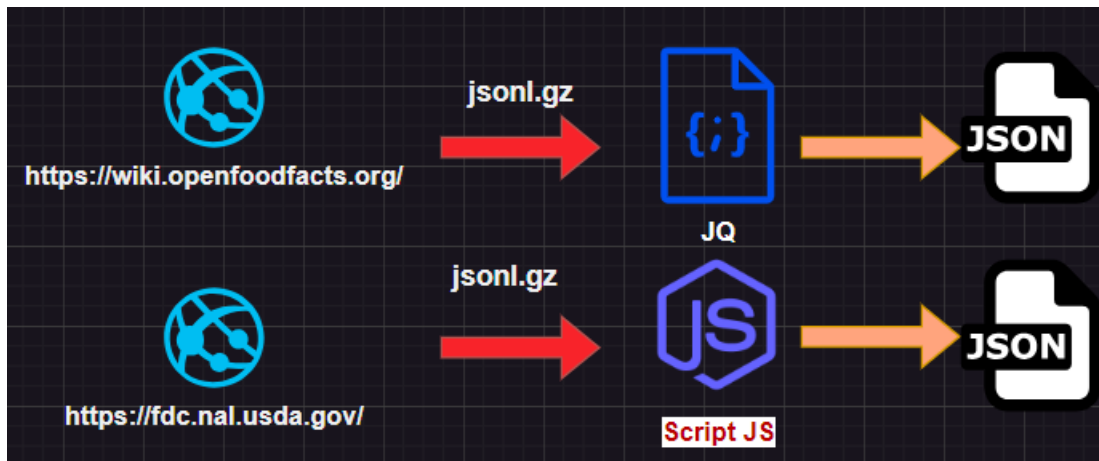


ExpressJS est un framework NodeJS préconstruit qui peut vous aider à créer des applications web côté serveur plus rapidement et plus intelligemment. Simplicité, minimalisme, flexibilité, évolutivité sont quelques-unes de ses caractéristiques et comme il est conçu en NodeJS, il a également hérité de ses performances.

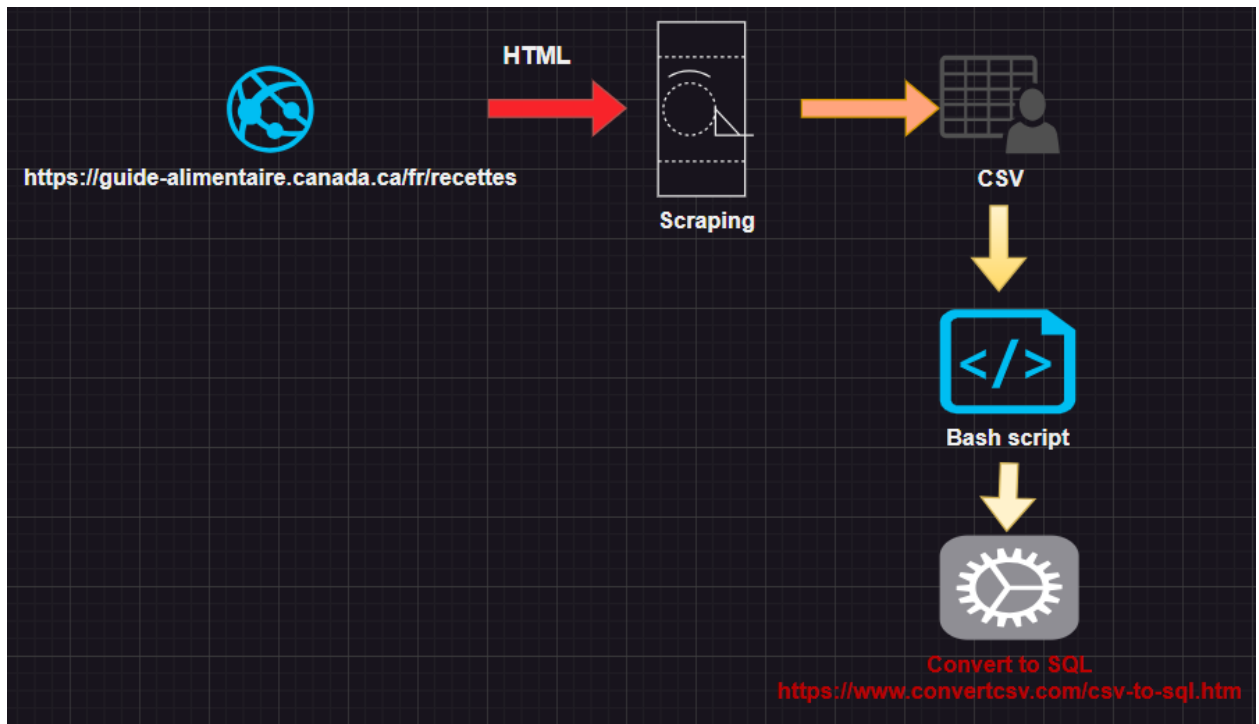
En bref, ExpressJS a fait pour NodeJS ce que Bootstrap a fait pour HTML/CSS et le responsive web design.

Il a fait du codage en NodeJS un jeu d'enfant et a donné aux programmeurs quelques fonctionnalités supplémentaires pour étendre leur codage côté serveur. ExpressJS est de loin le framework NodeJS le plus connu - à tel point que lorsque la plupart des gens parlent de NodeJS, ils veulent certainement dire NodeJS+ExpressJS.

## Annexe F : UML filtrage JSON



## Annexe G: UML Scraping site web des recettes



## Annexe H : Script Bash nettoyage liste ingrédients

```

1 $ script2:ingredients.sh
2
3 # File containing the list of ingredients
4 file="csv-ing.txt"
5
6 output_file="filtered_ingredients2.txt"
7
8 # Ensure the output file is empty or create it
9 > "$output_file"
10
11 # Read the input file line by line
12 while IFS= read -r line; do
13     # Remove text within parentheses
14     line=$(echo "$line" | sed 's/([^\s])*/g' | sed -E 's/[0-9]+//g' | sed -E 's/^[[:space:]],*/g' | sed -E 's/, +/, /g')
15     # Remove numbers and units (ml, L, oz, c. à thé, etc.)
16     line=$(echo "$line" | sed -E 's/[0-9]+([.][0-9]+)?//g' \
17 | sed -E 's/[[:space:]]*(ml|c\.\ à table|c\.\ à soupe|c\.\ à thé|tasse|boite|oz|X|Y|X)+[[:space:]]*/g' \
18 | sed -E 's/[[:space:]]*(\([0-9]+[[:space:]]*(ml|oz|g|t|c\.\ à table|c\.\ à soupe|c\.\ à thé|tasse|boite)[[:space:]]*)[[:space:]]*/g' \
19 | sed 's/^ */; s/ *$// ' | tr '\n' ',' | sed -E 's/,de/,/g' )
20     # Remove extra whitespace
21     line=$(echo "$line" | sed 's/[[:space:]]\+/ /; s/^ //; s/ $//; s/://g' | sed -E 's/,de/,/g' | sed -E "s/,d',/,/g" | awk ' !/^,/ { sub(/^de /, "", $0); print }')
22     # Append to the output file if the line is not empty
23     if [[ -n "$line" ]]; then
24         echo "$line" >> "$output_file"
25     fi
26 done < "$file"
27
28 echo "Filtered ingredients have been saved to $output_file"

```

## Annexe I : Transformation CSV-MySQL

```
CREATE TABLE recettes(  
  id                INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT  
  ,nomRecette       VARCHAR(64) NOT NULL  
  ,ingredientsRecette VARCHAR(998) NOT NULL  
  ,imgUrl           VARCHAR(159) NOT NULL  
);  
INSERT INTO recettes(id,nomRecette,ingredientsRecette,imgUrl) VALUES (1,'Salade de pois chiches et de carottes','Ingrédients  
1 grand concombre, tranché  
15 ml (1 c. à table) d''aneth frais haché ou 2 ml (¼ c. à thé) d''aneth séché  
2 carottes, épluchées et râpées  
1 boîte (540 ml/19 oz) de pois chiches, égouttés et rincés  
500 ml (1 tasse) de tomates cerise, coupées en deux dans le sens de la longueur  
60 ml (¼ tasse) de basilic frais haché  
45 ml (3 c. à table) de vinaigre balsamique  
15 ml (1 c. à table) de pesto de basilic  
10 ml (2 c. à thé) d''huile d''olive extra vierge
```

## Annexe J : Script Bash mise à jour liste ingrédients

```
$ mysql-inglist.sh
1  #!/bin/bash
2
3  # Paramètres de connexion MySQL
4  HOST="127.0.0.1"
5  USER="root"
6  PASSWORD="root_password"
7  DATABASE="recettes"
8  TABLE="recettes"
9
10 # Nom du fichier texte contenant les ingrédients
11 #FILE="filtered_ingredients2.txt"
12 FILE="ingredients-comma.txt"
13 # Initialiser l'ID à 1
14 id=1
15
16 # Lire le fichier ligne par ligne et mettre à jour la base de données
17 while IFS= read -r line
18 do
19     ingredients=$(echo "$line" | sed "s/'/\\'/g")
20     echo "Ingredients: $ingredients"
21     # Mise à jour de la colonne ingredientsList dans la table recettes
22     mysql -h "$HOST" -u "$USER" -p"$PASSWORD" "$DATABASE" -e "UPDATE $TABLE SET ingredientsRecette='$ingredients' WHERE id=$id;"
23
24     # Incrémenter l'ID
25     id=$((id + 1))
26 done < "$FILE"
```



## Annexe K : Document OFF transformé

```
{  
  "_id": "0055872025019",  
  "product_name": "Lait (2%)",  
  "ecoscore_score": 48,  
  "nutriscore_grade": "a",  
  "nova_groups": "1",  
  "categories": "Produits laitiers, Laits, Laits demi-écrémés",  
  "brands": "Québon"  
}
```

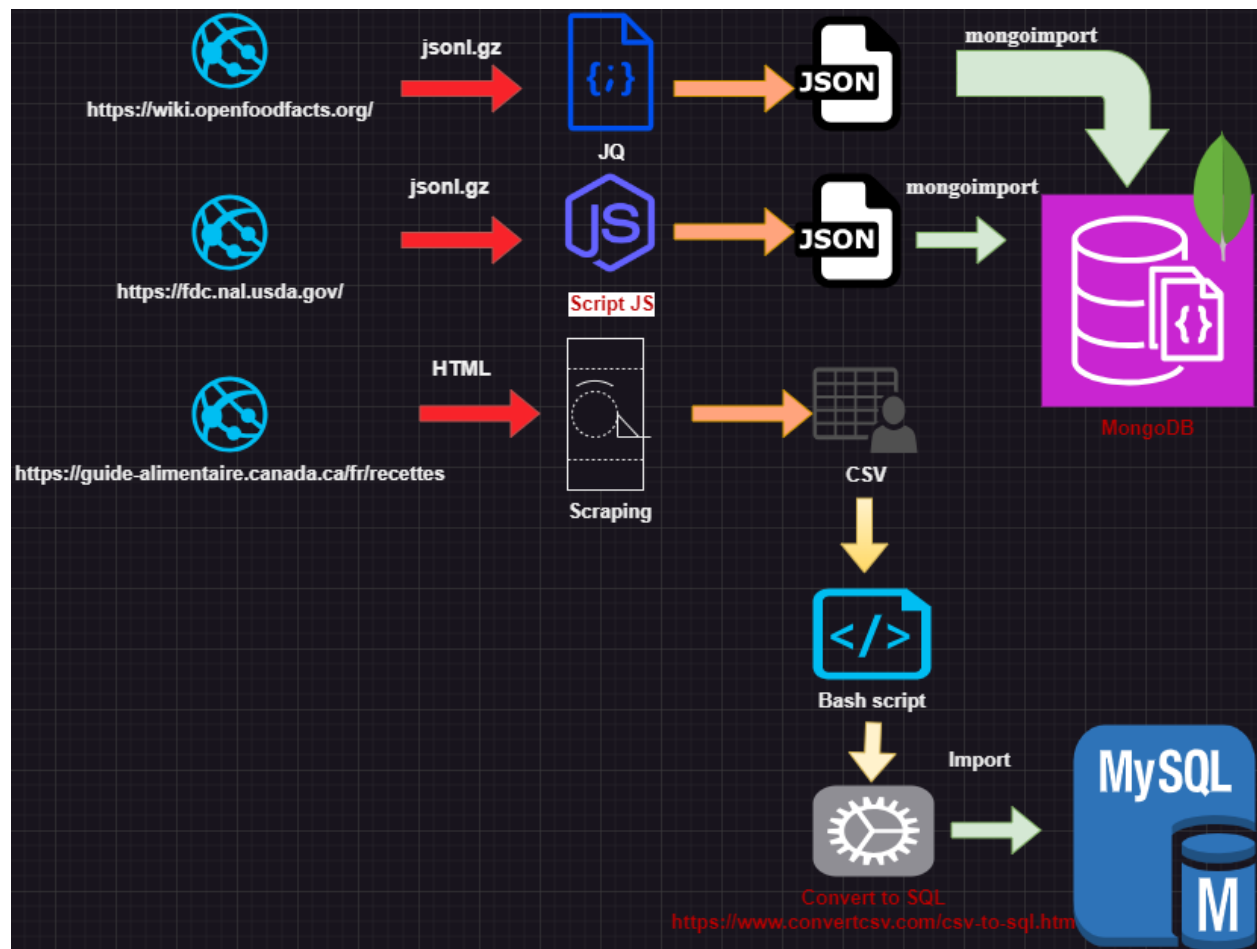
## Annexe L : Script de transformation de la BD USDA Branded.

```
app > JS transform_branded_ingredients.js > ...
1  const fs = require('fs');
2  const readline = require('readline');
3
4  // Paths for input and output files
5  const inputFile = 'data/mongodb/brandedDownload.jsonl';
6  const outputFile = 'data/mongodb/brandedDownload_transformed.jsonl';
7
8  // Create a write stream for the output file
9  const outputStream = fs.createWriteStream(outputFile);
10
11 // Create a readline interface to read the input file line by line
12 const rl = readline.createInterface({
13   input: fs.createReadStream(inputFile),
14   output: process.stdout,
15   terminal: false
16 });
17
18 // Process each line in the input file
19 rl.on('line', (line) => {
20   const data = JSON.parse(line);
21
22   // Extract only the fields you need
23   const transformedData = {
24     description: data.description,
25     foodNutrients: data.foodNutrients.map(nutrient => ({
26       name: nutrient.nutrient?.name,
27       rank: nutrient.nutrient?.rank,
28       amount: nutrient.amount
29     })))
30   };
31
32   // Write the transformed object as a JSON line to the output file
33   outputStream.write(JSON.stringify(transformedData) + '\n');
34 });
35
36 rl.on('close', () => {
37   outputStream.end();
38   console.log('Transformation complete. Output saved to', outputFile);
39 });
```

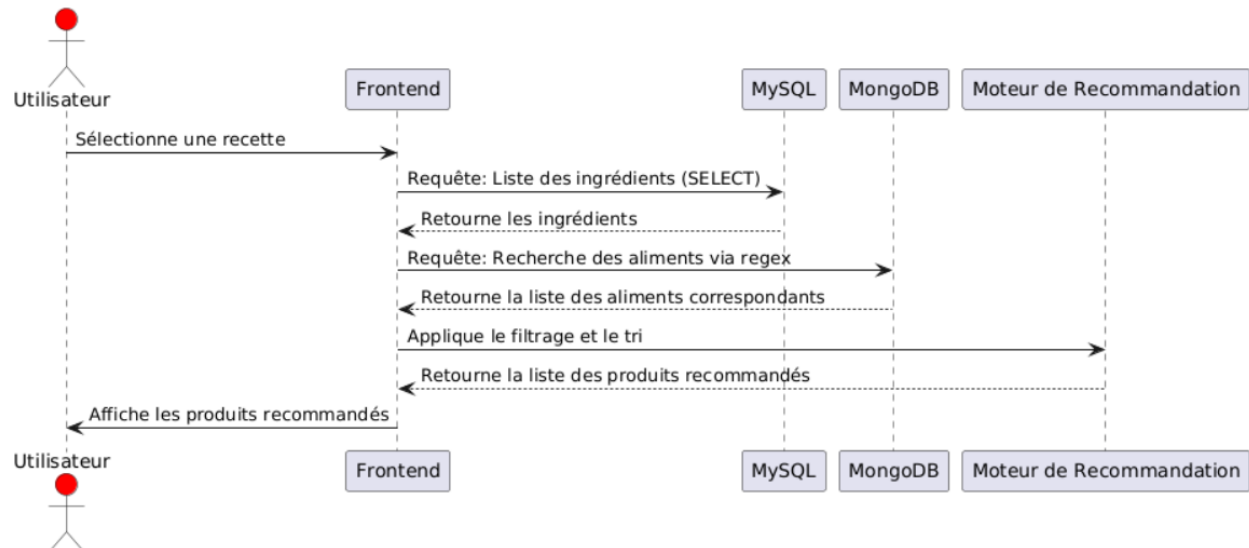
## Annexe M : BD USDA Branded transformée

```
{
  "foodClass": "Branded",
  "description": "MINI ANGEL FOOD CAKES",
  "foodNutrients": [
    {
      "type": "FoodNutrient",
      "id": 13696863,
      "nutrient": {
        "id": 1003,
        "number": "203",
        "name": "Protein",
        "rank": 600,
        "unitName": "g"
      }
    },
  ],
}
```

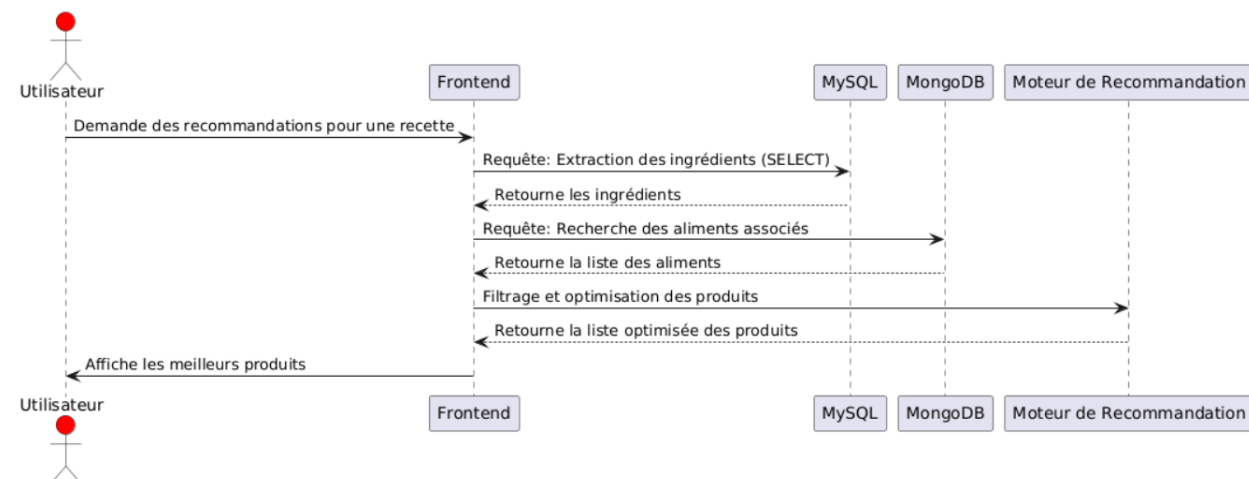
## Annexe N : ETL général



## Annexe O-1 : Diagramme de séquence création de recommandations



## Annexe O-2 : Diagramme de séquence obtention de recommandations



## Annexe P : Fragmentation BD MySQL

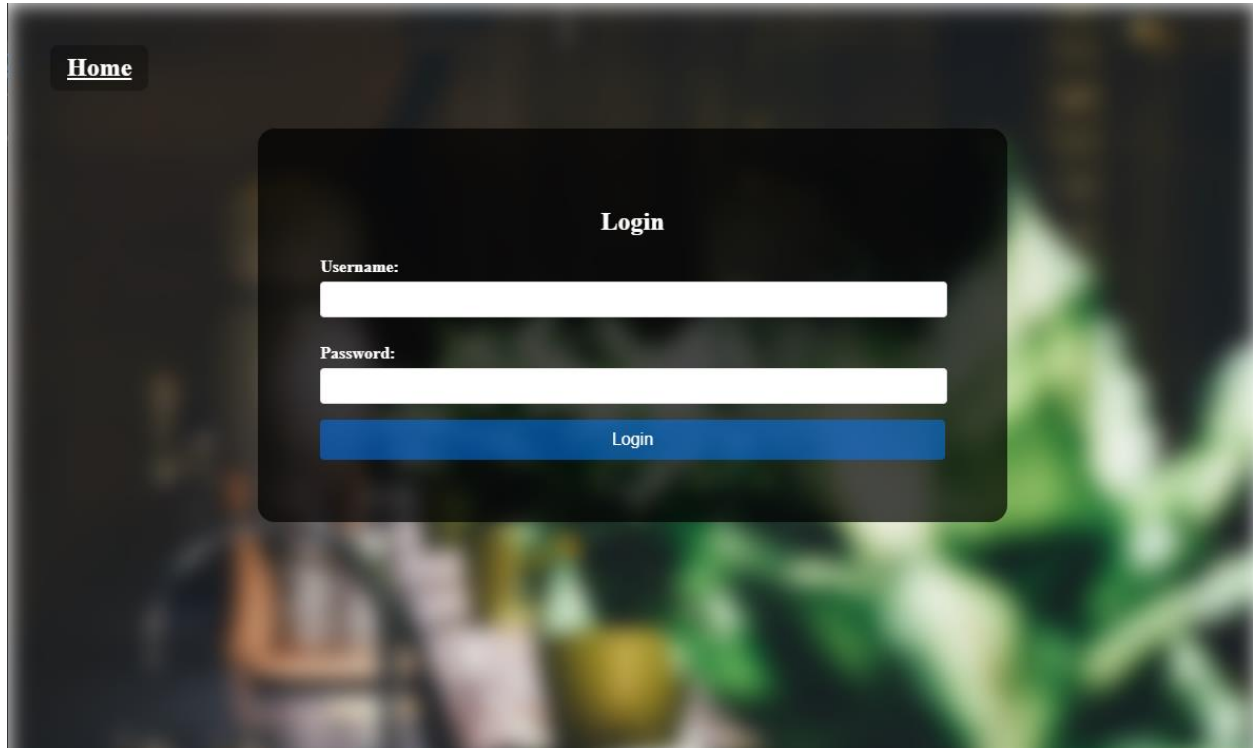
```
CREATE TABLE recettes (  
    id INTEGER NOT NULL PRIMARY KEY,  
    nomRecette VARCHAR(350) NOT NULL,  
    imgUrl VARCHAR(159) NOT NULL,  
    categoriesRecette VARCHAR(150),  
    tempsPrep VARCHAR(7) NOT NULL,  
    tempsCuisson VARCHAR(7) NOT NULL,  
    portions VARCHAR(50) NOT NULL,  
    ingredientsRecette VARCHAR(1500),  
    prepRecette VARCHAR(2500),  
    descrRecette VARCHAR(2500)  
)  
PARTITION BY LIST (categoriesRecette) (  
    PARTITION p_10ing VALUES IN ('10 ingrédients ou moins'),  
    PARTITION p_30min VALUES IN ('30 min ou moins'),  
    PARTITION p_enfants VALUES IN ('Pour enfants'),  
    PARTITION p_végétarien VALUES IN ('Végétarien'),  
    PARTITION p_sanscuisson VALUES IN ('Sans cuisson'),  
    PARTITION p_congelables VALUES IN ('Congelable'),  
    PARTITION p_autre VALUES IN ('autre', NULL)  
);
```

## Annexe Q : Script Bash ajout recette BD MySQL

```
# Lire le fichier ligne par ligne et insérer les données dans la base de données
while IFS="," read -r nomRecette imgUrl categoriesRecette tempsPrep tempsCuisson \
    portions ingredientsRecette prepRecette ingredientsList descRecette
do
    # Échapper les ' dans chaque colonne
    nomRecette=$(echo "$nomRecette" | sed "s/'/\\'/g")
    imgUrl=$(echo "$imgUrl" | sed "s/'/\\'/g")
    categoriesRecette=$(echo "$categoriesRecette" | sed "s/'/\\'/g")
    ingredientsRecette=$(echo "$ingredientsRecette" | sed "s/'/\\'/g")
    prepRecette=$(echo "$prepRecette" | sed "s/'/\\'/g")
    ingredientsList=$(echo "$ingredientsList" | sed "s/'/\\'/g")
    descRecette=$(echo "$descRecette" | sed "s/'/\\'/g")

    # Insertion dans la table recettes
    mysql -h "$HOST" -u "$USER" -p"$PASSWORD" "$DATABASE" -e "
    INSERT INTO $TABLE (
        nomRecette,
        imgUrl,
        categoriesRecette,
        tempsPrep,
        tempsCuisson,
        portions,
        ingredientsRecette,
        prepRecette,
        ingredientsList,
        descRecette
    ) VALUES (
        '$nomRecette',
        '$imgUrl',
        '$categoriesRecette',
        $tempsPrep,
        $tempsCuisson,
        $portions,
        '$ingredientsRecette',
        '$prepRecette',
        '$ingredientsList',
        '$descRecette'
    )"
```

## Annexe R : Page d'authentification



The image shows a login page with a dark, blurred background. In the top-left corner, there is a button labeled "Home". In the center, there is a dark rectangular box containing the "Login" form. The form has the title "Login" at the top, followed by a "Username:" label and a white input field. Below that is a "Password:" label and another white input field. At the bottom of the form is a blue button labeled "Login".

[Home](#)

**Login**

Username:

Password:

Login



## Annexe S : Page de recherche en action

Previous Page

# Search Branded Ingredients

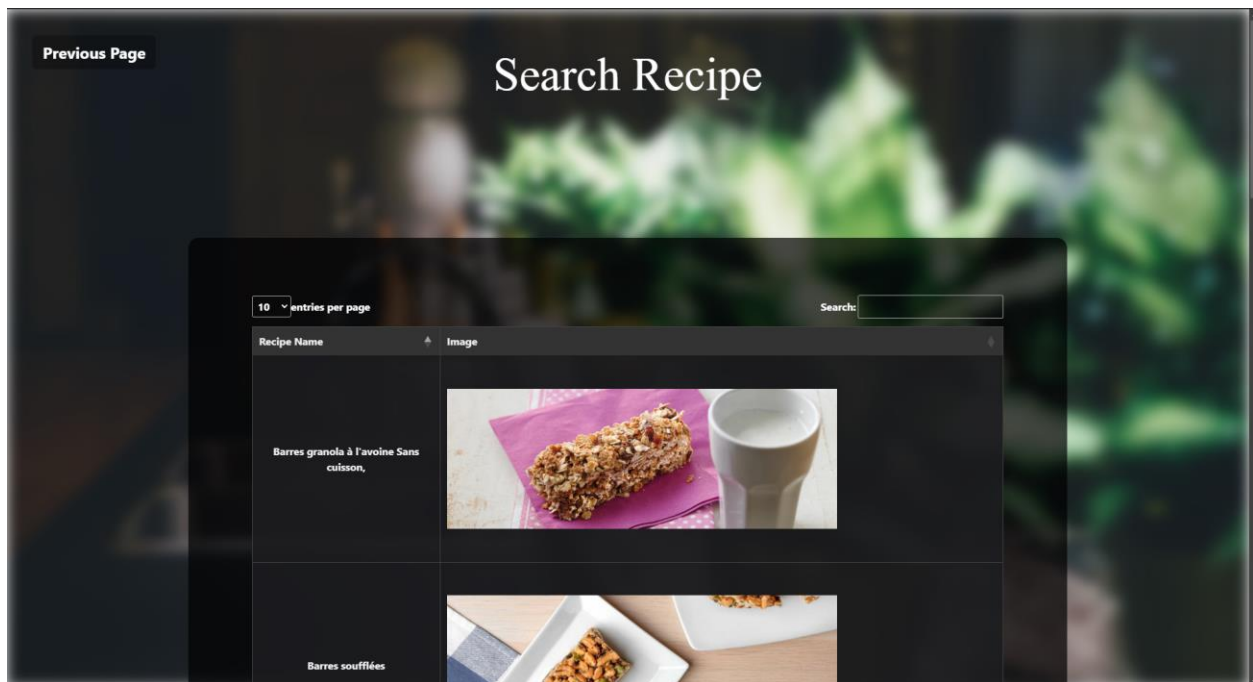
10 entries per page

Search: Fruit and vegetables

Brand Name	Categories	Description	Serving Size
BJ's Wholesale Club / Corporate Brands	Pre-Packaged Fruit & Vegetables	FRESH YELLOW ONION	148 g
BJ's Wholesale Club / Corporate Brands	Pre-Packaged Fruit & Vegetables	FRESH RED ONIONS	148 g
HEARTLAND	Pre-Packaged Fruit & Vegetables	PRIME SELECT RUSSET BAKING POTATOES	148 g
LITTLE SALAD BAR	Pre-Packaged Fruit & Vegetables	ROMAINE LETTUCE, CAESAR DRESSING, SHREDDED CHEESE AND CROUTONS CAESAR SALAD KIT	100 g
MOYER'S	Pre-Packaged Fruit & Vegetables	CHERRYCANDY APPLES WITH PEANUTS	150 g
NOT A BRANDED ITEM	Pre-Packaged Fruit & Vegetables	BABY LETTUCES, BABY GREENS, RADICCHIO, SPRING MIX	85 g
NOT A BRANDED ITEM	Pre-Packaged Fruit & Vegetables	ORGANIC MILDLY PEPPERY BABY ARUGULA LEAVES	142 g
NOT A BRANDED ITEM	Pre-Packaged Fruit & Vegetables	ORGANIC HALF & HALF BABY SPRING MIX + BABY	

Ici nous voyons que la librairie a par elle-même choisi la colonne « Categories » pour faire la recherche puis a aussi détecté l'équivalence du symbole « & » pour « and » en anglais.

## Annexe T : Interfaces Web



## Barres soufflées



Categories: 10 ingrédients ou moins, Pour enfants, Sans cuisson, Végétarien,

Portions: 12

Preparation Time: 40 min

Cooking Time: 0 min

### Ingredients:

1 l (4 tasses) de blé, de riz ou de kamut soufflé 45 ml (3 c. à table) de graines de chia 60 ml (¼ tasse) de graines de citrouille non salées 45 ml (3 c. à table) de pépites de cacao ou de mini pépites de chocolat 85 ml (½ tasse) de beurre d'arachide ou d'amande naturel ou une alternative sans noix 85 ml (½ tasse) de miel

### Preparation Instructions

1. Recouvrir un moule carré de 20x20 cm (8x8 pouces) de papier parchemin
2. Mettre de côté
3. Dans un grand bol, mélanger le blé, le riz ou le kamut soufflé avec les graines de chia, les graines de citrouille et les pépites de cacao
4. Dans un bol allant au four à micro-ondes, ajouter le beurre d'arachide et le miel
5. Faire chauffer au four à micro-ondes par intervalles de 20 secondes, en remuant entre chaque intervalle, jusqu'à ce que le mélange soit d'une consistance lisse et coulant
6. Verser le mélange chaud de beurre d'arachide sur le premier mélange
7. Bien mélanger
8. Presser le mélange dans le moule et mettre au congélateur pendant 30 minutes
9. Couper en 12 portions

Hide Healthy Alternatives

### Recommended Healthy Alternatives:

blé	Try: <a href="#">Villaggio - Blé entier</a> <b>Nutriscore: A</b>
riz	Try: <a href="#">Boisson De Riz Biologique Enrichie Vanille</a> <b>Nutriscore: B</b>
graines de chia	Try: <a href="#">Triscuit, graines de chia, saveur romarin et piments Jalapeños</a> <b>Nutriscore: D</b>
miel	Try: <a href="#">Biscuits Graham au miel</a> <b>Nutriscore: D</b>

[Previous Page](#)

### Triscuit, graines de chia, saveur romarin et piments Jalapeños

Categories: Snacks, Snacks salés, Amuse-gourmes, Biscuits apéritifs

Enrichies: 0%

Nutriscore: d

Nutra Group: No value yet

# Search Ingredients

10 ▾ entries per page

Search:

Product Name	↑	EcoScore	NutriScore	Nova Score
Ailes de poulet BBQ doux		22	c	4
Ailes de poulet Buffalo		22	c	
Ailes de poulet piquante		22	d	
Ailes de poulet poivre et sel marin		22	d	
Ailes de poulet ranch		22	c	4
Ailes de poulet roti au four, coupées assaisonnée et cuites		22	c	
Ail et herbes			c	
Ail forte		78	d	
Ail haché		83	c	
Ail Mariné Clic		78	c	

Showing 481 to 500 of 15,882 entries

« ‹ 1 ... 49 50 51 ... 1,589 › »

## BTY CRK FRT ROLLUPS FRT FLVRD SNCKS FRANKEN BRY STRWBRY SCREAM by

Chips/Crisps/Snack Mixes - Natural/Extruded (Shelf Stable)

Vitamin C, total ascorbic acid	42.9 mg (Rank: 6300)
Cholesterol	0 mg (Rank: 15700)
Protein	0 g (Rank: 600)
Total lipid (fat)	7.14 g (Rank: 800)
Carbohydrate, by difference	78.6 g (Rank: 1110)
Energy	357 kcal (Rank: 300)
Total Sugars	50 g (Rank: 1510)
Sodium, Na	357 mg (Rank: 5800)
Fatty acids, total trans	0 g (Rank: 15400)
Fatty acids, total saturated	3.57 g (Rank: 9700)

## Bibliographie

- [1] : <https://world.openfoodfacts.org/> , consulté le 24 septembre 2024
- [2] : <https://aliments-nutrition.canada.ca/cnf-fce/?lang=fre> , consulté le 24 septembre 2024
- [3] : <https://nodejs.org/en/about> , consulté le 21 septembre 2024
- [4] : [https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs#what\\_is\\_nodejs](https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs#what_is_nodejs) , consulté le 21 septembre 2024
- [5] : <https://medium.com/@lukepierotti/node-js-what-is-it-and-how-does-it-work-e3e83d054662> , consulté le 21 septembre 2024
- [6] : <https://www.geeksforgeeks.org/express-js/> , consulté le 21 septembre 2024
- [7] : <https://www.algoworks.com/blog/why-use-expressjs-over-nodejs-for-server-side-coding/> , consulté le 29 octobre 2024
- [8] : AI générative
- [9] : <https://www.mongodb.com/company/what-is-mongodb> , consulté le 29 septembre 2024
- [10] : [https://fdc.nal.usda.gov/GBFPD\\_Documentation.html](https://fdc.nal.usda.gov/GBFPD_Documentation.html) , consulté le 29 septembre 2024